Teaching a Powered Prosthetic Arm with an Intact Arm Using Reinforcement Learning

by

Gautham Vasan

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The idea of an amputee playing the piano with all the flair and grace of an able-handed person may seem like a futuristic fantasy. While many prosthetic limbs look lifelike, finding one that also moves naturally has proved more of a challenge for both researchers and amputees. Even though sophisticated upper extremity prostheses like the Modular Prosthetic Limb (MPL) are capable of effecting almost all of the movements as a human arm and hand, they can be useful only if robust systems of control are available. The fundamental issue is that there is a significant mismatch between the number of controllable functions available in modern prosthetic arms and the number of control signals that can be provided by an amputee at any given moment. In order to bridge the gap in control, we require a neural interface that can translate the physiological signals of the user into a large number of joint commands for simultaneous, coordinated control of the artificial limb.

In this thesis, we focus on a collaborative approach towards the control of powered prostheses. In our approach, the user shifts greater autonomy to the prosthetic device, thereby sharing the burden of control between the human and the machine. In essence, the prosthesis or rehabilitative device learns to "fill in the gaps" for the user. With this view in our mind, we developed a method that could allow someone with an amputation to use their non-amputated arm to teach their prosthetic arm how to move in a natural and coordinated way by simply showing the prosthetic arm the right way to move in response to inputs from the user. Such a paradigm could well exploit the muscle synergies already learned by the user. Consider cases where an amputee has a desired movement goal, e.g., "add sugar to my coffee," "button up my shirt," or "shake hands with an acquaintance". In these more complicated examples, it may be difficult for a user to frame their objectives in terms

of device control parameters or existing device gestures, but they may be able to execute these motions skillfully with their remaining biological limb.

As a first contribution of this thesis, we present results from our work on learning from demonstration using Actor-Critic Reinforcement Learning (ACRL), and show that able-bodied subjects ($n = 3$) are able to train a prosthetic arm to perform synergistic movements in three degrees of freedom(DOF) (wrist flexion, wrist rotation and hand open/close). The learning system uses only the joint position and velocity information from the prosthesis and above-elbow myoelectric signals from the user. We also assessed the performance of the system with an amputee participant and demonstrate that the learning-from-demonstration paradigm can be used to teach a prosthetic arm natural, coordinated movements with the intact arm.

For our second contribution, we describe a sensor fusion and artificial vision based control approach that could potentially give rise to context-aware control of a multi-DOF prosthesis. Our results indicate that the learning system can make use of the addition sensory and motor information to determine and context and differentiate between different movement synergies.

Our results suggest that this learning-from-demonstration paradigm may be well suited to use by both patients and clinicians with minimal technical knowledge, as it allows a user to personalize the control of his or her prosthesis without having to know the underlying mechanics of the prosthetic limb. This approach may extend in a straightforward way to next-generation prostheses with precise finger and wrist control, such that these devices may someday allow users to perform fluid and intuitive movements like playing the piano, catching a ball, and comfortably shaking hands.

Preface

A version of Chapter 3 has been accepted for publication and presentation as Gautham Vasan, Patrick M. Pilarski, *Learning from Demonstration: Teaching a Myoelectric Prosthesis with an intact limb via Reinforcement Learning*, for the Proc. of the 2017 IEEE International Conference on Rehabilitation Robotics (ICORR). London, United Kingdom, 2017. It was presented as a poster and also as a fast-forward session podium talk for 60 seconds. Our work was nominated for the best poster award (top 20 among 600+ posters) as a part of Rehab Week 2017.

An extended abstract of this paper was also presented as a poster and podium talk ($20mins$) at the Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM), Ann Arbor, Michigan, June 11-14, 2017. I was responsible for experimental design, implementation, execution, and analysis and for the bulk of manuscript composition. Pilarski was the supervisory author and contributed to the manuscript and provided implementation feedback and insights throughout.

This research received ethics approval for two protocols from the human research ethics board at the University of Alberta

Dedication

To my parents, Vasan and Vijayalakshmi. Thank you both for the love and support over the years and letting me live on my own terms.

To my brother, Srinath, thanks for putting up with all my random notions and long rants about AI, politics, etc. and countering my arguments with a great sense of humor

We have a brain for one reason one reason only:
to produce adaptable and complex movements

– Daniel Wolpert

Acknowledgements

I wish to express my gratitude to Patrick M. Pilarski. His enthusiasm and optimistic outlook towards research is highly contagious :) To Patrick: thanks a lot for your mentorship and constant encouragement. You've helped me grow as a scientist and provided numerous opportunities to be a part of amazing research. I couldn't have asked for a better supervisor.

I'd also like to thank Rich Sutton. His course was the major decisive factor in my choice to work on Reinforcement Learning. He's provided me with critical feedback which I believe has made me a better researcher. Many thanks to Martha White and Ming Chan for their thoughtful questions and suggestions as a part of my thesis committee.

I'm incredibly grateful Jaden Travnik, Vivek Veeriah, Craig Sherstan, Michael Rory Dawson and Kory Mathewson for all their help throughout. We've had several interesting discussions about AI, neuroscience and robotics. I am also hugely grateful to other colleagues in the BLINC lab (Alex, Nadia, Dylan, Katherine, Tarvo, McNiel, Jon, Sai, Riley, Quinn, Oggy and James) and the RLAI lab for their genuine affability and camaraderie.

Many thanks to Trevor Lashyn, Matthew Curran and Ming Chan for getting me involved in the Cerebral Palsy and Spasticity trials. It was a great learning experience.

To my long time friends Naresh Balaji Ravichandran and Siddharthan Rajasekaran, I am extremely thankful for all the absurd, wacky and intellectually inspiring conversations.

Table of Contents

List of Figures

Glossary

action Choice made by the agent from some set, based on current state, which in general affects the next state and reward received. 13, 16

action potential The change in electrical potential associated with the passage of an impulse along the membrane of a muscle cell or nerve cell. 6

afferent Neurons that receive information from our sensory organs (e.g. eye, skin) and transmit this input to the central nervous system are called afferent neurons. 77

agent In the Reinforcement Learning setting, the learner and decision-maker is called the agent. 11

DOC Degrees of Control. The number of independent functions a user can control. This can be a subset of the degrees of freedom, or may be aggregate functions, which control more than one degree of freedom. 7–9, 63

DOF Degrees of Freedom. The number of independent motions a system can produce. iii, ix, 1, 7, 8, 11, 26, 29, 55, 56, 58, 59, 63–66, 70, 76, 77, 79, 80

EEG Electroencephalogram. The measurement of electrical activity in the brain as measured through the scalp. 63

efferent Neurons that send impulses from the central nervous system to your limbs and organs are called efferent neurons. 77

EMG Electromyography. The recording of electrical signals generated by muscle contraction. 1, 5–7, 9, 36, 63

environment Everything the agent interacts with (outside the agent) is called the environment. 11, 27

exteroception The perception of environmental stimuli acting on the body. The stimuli are perceived by special, sometimes highly complex, structures called exteroceptors. An example of exteroception is the perception of light, sound, or heat. 54, 55, 77

GUI Graphical User Interface. 18, 93

i.i.d Independent and Identically Distributed. 12

MPL Modular Prosthetic Limb. It is a sophisticated upper extremity prostheses that is capable of effectuating almost all of the movements as a human arm and hand. It has more than 100 sensors in the hand and upper arm (26 DoF and 17 degrees of control). In addition, it has unique fingerprints. 2, 7, 8

policy Probability distribution defining probability of taking each action from each state. 13

prehension The use of the hands and fingers to grasp, pick up objects, or pinch. 55

proprioception The sense of position and movement of the limbs and the sense of muscular tension. The awareness of the orientation of the body in space and the direction, extent, and rate of movement of the limbs depend in part upon information derived from sensory receptors in the joints, tendons, and muscles. Information from these receptors, called proprioceptors, is normally integrated with that arising from vestibular receptors (which signal gravitational acceleration and changes in velocity of movements of the head), as well as from visual, auditory, and tactile receptors. Sensory information from certain proprioceptors, particularly those in muscles and tendons, need not reach consciousness, but can be used by the motor system as feedback to guide postural adjustments and control of well-practiced or semiautomatic movements such as those involved in walking. 54, 55, 68, 77

Reinforcement Learning A form of artificial intelligence in which an agent learns directly from experience how to behave, so as to maximize long-term reward. 6, 74

return Some specific function of the reward sequence (for e.g., $R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$). 13–15, 19, 73

reward Special numerical/scalar value supplied by the environment after each action. The agent tries to maximize cumulative reward. 11, 13

saccade A rapid movement of the eyes that changes fixation from one point to another. 67

state Immediate information available to the agent at the current time-step. 12, 15, 16

TMR Targeted Muscle Reinnervation. A surgery whereby motor nerve endings are transplanted into new host muscle. 6

TSR Targeted Sensory Reinnervation. A surgery whereby the sensory nerve endings are transplanted to new subcutaneous locations to provide a way of restoring sensory feedback for amputees. 6

value Expected sum of rewards received by the agent if they start from state S and then follow policy π. 13

Chapter 1

Introduction

Humans often exploit the dynamics of their complex musculoskeletal system in ingenious ways to generate efficient and coordinated movement. When the central nervous system (CNS) produces voluntary movement, various muscles, each comprising thousands of motor units, are simultaneously activated and coordinated. Computationally, this is a daunting task since the CNS needs to handle the large number of degrees of freedom (DOF) that must be continually adjusted and controlled (i.e., the *degrees-of-freedom problem* described in Turvey et al. (1982)). The CNS also needs to consider the highly complex, nonlinear relationship between alterations in the settings of the degrees of freedom and the effects of those alterations. However, according to Bernstein (1967), humans do not control elementary degrees of freedom, but instead use *muscle synergies*—the coordinated activation of a group of muscles—to handle their degrees-of-freedom problem. Recent findings of d'Avella et al. (2006) suggest that the CNS encodes a set of muscle synergies, and that it combines them in a task-dependent fashion in order to generate the muscle contractions that lead to desired movements. When someone loses an body part, it can be replaced with an artificial device known as a *prosthesis*. While modern hand prostheses now include a limited number of predefined synergistic grasping patterns, synergistic actuation of the kind described by d'Avella et al. is largely missing from most if not all commercial prosthetic devices.

Since the 1960s, the most common way of controlling powered prostheses has been through surface electromyography (sEMG), termed *myoelectric control*, which involves measuring the electrical manifestation of muscle contraction. Despite sig-

nificant technological advancements, a large proportion of amputees stop using myoelectric prostheses due to non-intuitive control, lack of sufficient feedback, and insufficient functionality (Peerdeman et al. (2011)). Even though sophisticated upper extremity prostheses like the Modular Prosthetic Limb (MPL) are capable of effectuating almost all of the movements as a human arm and hand and with more than 100 sensors in the hand and upper arm (26 DoF and 17 degrees of control) (Bridges et al. (2011)), they can be useful only if robust systems of control are available.

1.1 What Should the Ideal Prosthetic Control System Do?

There is a significant mismatch between the number of controllable functions available in modern prosthetic arms (e.g., MPL) and the number of control signals that can be provided by amputee at any given moment. This is a fundamental open problem that is relevant to prosthesis as well as Human Computer Interaction (HCI) domains. One of our goals at the Bionic Limbs for Improved Natural Control (BLINC) lab at the University of Alberta is to create intelligent systems that bridge this gap in control; we want to develop assistive/rehabilitative devices (predominantly prosthetic arms) that understand a user's needs, anticipate them and take appropriate actions to assist the user. Such devices must adapt to changes in the user's behavior, in their environment, and their own capabilities (Sherstan (2015)). In this thesis work, we are working towards developing a highly intelligent robotic arm that could be worn as a prosthetic arm. Such an intelligent prosthetic arm should have the following capabilities from Pilarski et al. (2011):

- Translate physiological signals (e.g., brain/muscle signals) into usable control commands for a powered prosthetic/robotic limb.
- Automatically tailor the control system to the needs and specific physical conditions of individual patients, without constant manual intervention, retraining, and periods of frustration or reduced function for the patient.
- Improve limb control based on (sparse) patient feedback.

1.2 Our Approach to Addressing These Challenges

State of the art approaches in myoelectric control are trying to address a fundamental issue—how to overcome the significant mismatch between the number of functions available in a modern powered prosthesis and the number of functions an amputee can attend to at any moment. With this goal in mind, in the present work we develop a method that could allow someone with an amputation to use their non-amputated arm to teach their prosthetic arm how to move in a natural and coordinated way. Such a paradigm could well exploit the muscle synergies already learned by the user. Consider cases where an amputee has a desired movement goal, e.g., "add sugar to my coffee," "button up my shirt," or "shake hands with an acquaintance." In these more complicated examples, it may be difficult for a user to frame their objectives in terms of device control parameters or existing device gestures, but they may be able to execute these motions skillfully with their remaining biological limb.

One approach that has been shown to reduce barriers for humans specifying a complex control policy (i.e., a desired behavior) is *learning from demonstration* (LfD) (Argall et al. (2009)). In LfD, a policy that map states to actions is learned from the examples or demonstrations provided by the teacher. The examples are defined as a sequence of state-action pairs or trajectories that are recorded during the teacher's demonstration of the recorded behavior (Argall et al. (2009)). By formulating prosthetic limb training as a LfD task, we present a new scenario wherein an amputee could teach their prosthesis how to move by showing desired movements via the movement of his or her non-amputated limb.

1.3 Outline

We have attempted to use an informal tone to ensure that the work is accessible to a multi-disciplinary audience. However, that is largely limited to using illustrative examples and terminology to explain certain aspects of our experiments or to describe Reinforcement Learning (RL) to researchers outside the Computing Sciences. All the equations and parameters presented in this thesis use the standard RL notation

as described in *Reinforcement Learning: An Introduction* by Sutton and Barto.

Chapter 2 summarizes information about state of the art prosthetic devices, current approaches for myoelectric control and their associated challenges. In addition, it also introduces the RL techniques used in Chapters 3, 4, 5 and 6. We also look at some of the challenges in applying RL to real-world robotics domains.

Chapter 3 introduces a novel learning-from-demonstration paradigm via Actor-Critic Reinforcement Learning (ACRL). We describe an intuitive approach to training a prosthetic control system that involves the participant using their intact arm to teach the prosthetic arm various movement synergies corresponding to their inputs. This chapter should be considered the most significant contribution of this thesis.

In Chapter 4, we evaluate the learning-from-demonstration paradigm on an amputee subject. We discuss the experiments used to test the control algorithm described in the previous chapter and the results of these experiments.

Chapter 5 presents the preliminary results of context-dependent predictions based on contextually relevant representations. We use artificial vision and other sensory information to distinguish between different context-dependent motor behaviors.

Lastly in Chapter 6, we talk about the future directions of research, some of which were discussed in the previous chapters. We examine its significance, limitations and potential applications. The concluding chapter includes final remarks and a summary of this work.

1.4 Key Contributions

The contributions of this thesis are summarized as follows:

1.4.1 Extending Learning from Demonstration to Powered Prosthesis Domains

As a first contribution of this thesis, we present an actor-critic reinforcement learning method that for the first time promises to allow someone with an amputation to use their non-amputated arm to teach their prosthetic arm how to move through a wide range of coordinated motions and grasp patterns. We evaluate our method during the myoelectric control of a multi-joint robot arm by amputee and non-amputee

users, and demonstrate that by using our approach a user can train their arm to perform simultaneous gestures and movements in all three degrees of freedom in the robot's hand and wrist based only on information sampled from the robot and the user's above-elbow myoelectric signals.

1.4.2 Context-Aware Learning from Demonstration

For our second contribution, we describe a method to achieve situation-dependent movement based on muscle excitation. We hypothesize that the control system should be given the relevant contextual information and meta-data about the user, the robotic limb and its environment in order to achieve situation-dependent motion. We provide camera data and additional sensory information from the socket of a prosthesis to allow an ACRL system to produce varied motor synergies in response to similar EMG signals from the user. Our results indicate that a system can use additional sensor and state information to help manage the user's degree-of-freedom problem, generating synergies that artfully align to different situations in the user's daily life.

1.4.3 Development of Delsys Trigno, CyberTouch II, Thalmic Myo and Bento Arm Control and Software Interface

A significant part of my research involved working with multiple sensory systems to obtain both physiological and non-physiological signals from the user and state information from the robotic arm. While this is not the main part of my thesis, the software developed for interfacing the EMG system (Delsys Trigno Wireless Lab), the motion capture glove (CyberTouch II), a wearable gesture recognition device (Thalmic myo) and the Bento Arm was a significant measurable output of the work of my MSc degree. This is described in Appendices (A-E).

Chapter 2

Background

Recently, there has been a lot of coverage on sophisticated upper-limb prostheses often dubbed as "mind-controlled bionic arm" (Barbour (2012)). Several advancements in signal processing of EMG signals and surgical innovations (e.g., targeted muscle reinnervation TMR, targeted sensory reinnervation TSR) have paved the way for these achievements. Nevertheless, there exist significant challenges that should be addressed before these advancements can be beneficial to the amputee population. In this chapter, I will discuss the basic idea of myoelectric control and the state of the art control approaches.

This chapter then introduces Reinforcement Learning (RL) and a subclass of control algorithms known as Actor-Critic methods. I also briefly discuss a predictive RL algorithm, known as general value functions (GVFs), and a method of constructing features from real-valued signals, known as tile coding.

2.1 Myoelectric Control

The electrical activity in muscle tissue can be recorded with electrodes placed over the skin. The electrical manifestation of the muscle contractions is called electromyography (EMG), and it contains information about the neural signals sent from the spinal cord to control the muscles [1]. In the case of a powered upper-limb

[1]The central nervous system (CNS) controls the force generated by a muscle with recruitment of motor units and adjustments in their discharge rates (Adrian and Bronk (1929)). These motor units discharge action potentials which innervate the muscle and give rise to electrical activity. The number of action potentials discharged by the motor neurons innervating the muscle accounts for the neural drive from the spinal cord to the muscle.

prosthesis user, the electrodes are placed directly above the remnant muscles that provide strong, repeatable signals. Myoelectric signal as control input has several advantages over other inputs (Parker et al. (2006)):

- The user is freed of straps and harnesses
- The signal is non-invasively detected on the surface of the skin
- The muscle activity required to provide control signals is relatively small and can resemble the effort required of an intact limb,
- It can be adapted to proportional control with relative ease
- The required electronic circuits (whether analog or digital) can be continuously improved, miniaturized and have better long-term reliability.

Although control of prosthetic functions (e.g., hand open/close, etc.) is possible through physiological signals other than EMG (e.g., brain or nerve signals), surface EMG has virtually been the only control signal for commercial, everyday use of upper limb prosthesis since the 1950s (Jiang et al. (2012)). Despite decades of research and development, however, myoelectric control of upper limb prostheses still generates relatively limited clinical (and commercial) impact, as only one out of four upper limb amputees chose to use myoelectric-controlled prostheses (Wright et al. (1995)).

2.1.1 Degrees of Control

Degrees of Freedom (DOF) of a system refer to the number of functions that can be operated independently. In physical systems, this is the number of independent movements a system can make. *Degrees of control* (DOC) are the number of controllable functions actually available to the user, which may be a subset of a systems DOF or aggregations of multiple DOF (Sherstan (2015)). That is, the DOC can be less than, equal to, or greater than the DOF of a system. Take, for example, the MPL, which has 26 DOF and 17 DOC. This essentially means that the MPL user can send 17 independent control commands that could result in 26 discrete joint movements.

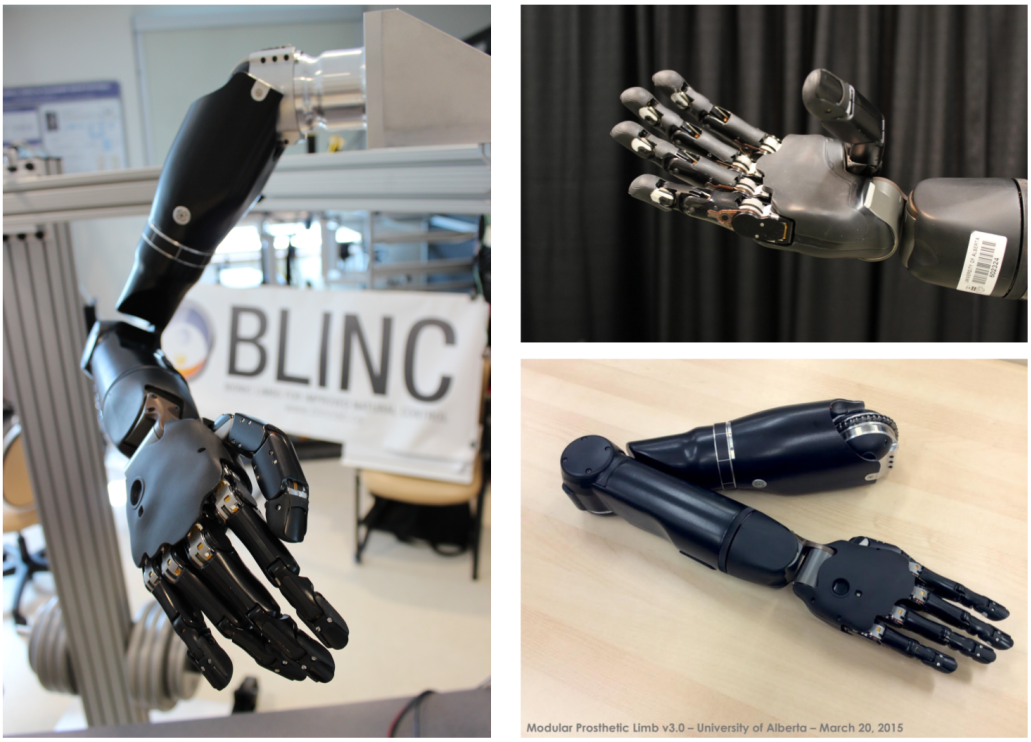


Figure 2.1: The Modular Prosthetic Limb (MPL). It is arguably the most advanced prosthetic arm till date. It has 26 DOF, 17 DOC, and 100+ sensory percepts. The MPL can effect almost all the movements of a biological arm. The MPL therefore promises to provide patients with improved, dexterous control of a prosthetic arm and hand, including a sense of touch.

In the prosthesis domain, DOC can vastly outnumber the number of input channels the user has available, with disparity increasing as the level of amputation increases (e.g., a person with hand amputation can provide more control signals compared to those with transhumeral amputations). For example, in the case of a transhumeral amputee (i.e., amputation at the level of elbow disarticulation), the user could provide signals from the biceps, triceps, deltoid and pectorals. Unfortunately, this user cannot simultaneously control all the joints of the MPL because the number of controllable functions vastly outnumber the number of input signals from the user.

2.1.2 Conventional Myoelectric Control

The most widely used approach to myoelectric control is still direct proportional control (Pilarski and Hebert (2017)). These schemes use an amplitude measure at each electrode site (e.g., root mean square or mean absolute value of the EMG) to quantify the intensity of contraction in the underlying muscles. In direct control, the magnitude of muscle contraction is used to move a degree of control (DOC, involving one or more prosthetic joints) of the prosthesis using a proportional mapping (Parker et al. (2005)). This allows the selection of control muscles based on physiological functions but has the disadvantage of typically requiring two control muscles for each prosthetic DoC (Parker et al. (2005)). If physiologically appropriate muscles are available to restore lost function, the EMG signals obtained from the user can be used intuitively. In the absence of physiologically appropriate muscles, we substitute them with alternate signals that can control the desired joint. In a scenario with multiple DOCs and limited physiological control signals from the user, a mode switching strategy (using a hardware switch or co-contraction of muscle pairs) is often employed. Adaptive switching methods use RL methods to sequentially transition between different DOCs (c.f., Edwards et al. (2016); Edwards (2016)). These simplistic methods provide reliable control, but lack the functionality to smoothly operate multiple DOCs.

2.1.3 Pattern Recognition

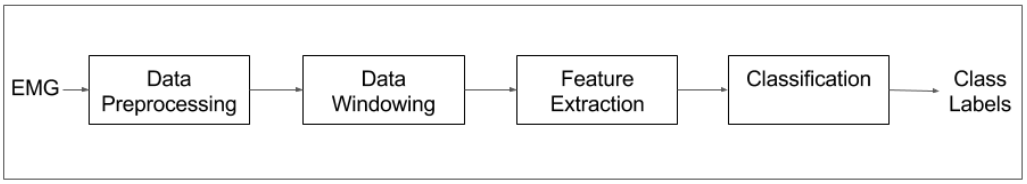


Figure 2.2: States of signal processing for EMG pattern recognition adapted from Scheme and Englehart (2011).

More recently, pattern recognition methods have started to see commercialization and clinical use (Scheme and Englehart (2011); Castellini et al. (2014)). Pattern Recognition methods use classification (Scheme and Englehart (2011)) and regres-

sion (Hahne et al. (2014)) techniques to translate EMG signals into usable control commands for a multi-function prosthetic limb. All EMG pattern recognition approaches have the fundamental processing stages shown in Fig 2.2. The EMG signals are filtered to remove unwanted interference (e.g., motion artifacts due to electrode movement, power line harmonics, etc.). *Data windowing* (also known as segmentation) is usually performed in the intervals of $150 - 200ms$ to obtain an acceptable control delay. A *feature extraction* stage is added to increase the information density of the EMG signals. Ideally, contraction discrimination information should be retained while other irrelevant information is discarded (Scheme and Englehart (2011)). EMG feature extraction methods including time-domain (TD), autoregressive (AR), and cepstral features have been extensively investigated (Scheme and Englehart (2011)). Joint time-frequency methods have been shown to effectively represent transient EMG patterns resulting from dynamic contractions. A comparison of the different feature extraction methods has shown that for slowly varying EMG patterns, a concatenated TD/AR (TDAR) feature set outperforms all others (Huang et al. (2005); Hargrove et al. (2007)). But the TDAR feature set incurs considerable processing overhead for a slight improvement in performance over simple TD features. (Scheme and Englehart (2011)) have reported that linear discriminant analysis (LDA), support vector machines (SVM) and hidden markov models (HMM) are the most popular choices of classifiers in recent work based on marginal advantages in classification performance.

Myoelectric classification for prosthetic control is not only possible but also highly accurate, even with a large number of functions (> 10) (Gijsberts and Caputo (2013); Atzori et al. (2013)). Currently, the only commercially available example of pattern recognition is Complete Control (Coapt LLC). In the Coapt system, all electrodes in a prosthesis socket are fed into a single pattern recognition module, which then provides output signals to a set of actuators.

The main problem with pattern classification is that it inherently leads to a control scheme that is substantially different from natural control. While natural movements are continuous and require simultaneous, coordinated articulations of the multiple DoFs, pattern classification provides only a discrete approximation of

the continuous parameter space. Current methods can typically generate reliable activation in only one class. Additionally, proportional control is not directly obtained from the classification, but instead requires additional processing (Jiang et al. (2012)).

2.1.4 Motor Skill Learning

Few alternative methods employ ideas drawn from motor skill learning and brain plasticity to extend direct control principles to multiple DOFs. Pistohl et al. and Ison et al. showed that users adapt to controls within a single session regardless of their initial intuitiveness or relationship with kinematics and develop muscle synergies associated with enhanced control of the myoelectric interface (Pistohl et al. (2013); Ison et al. (2016)).

Even though this approach promises improved control for prosthetic users, it relies completely on the human user to adapt to his/her prosthetic device rather than vice versa. Great utility may arise from a bidirectional partnership between the prosthetic device and its human user—while the user improves his or her ability to communicate their intentions to the prosthesis, the prosthesis would learn to anticipate and adapt to that specific needs of the user and improve its own ability to satisfy them (Pilarski et al. (2015)).

2.2 Reinforcement Learning (RL)

2.2.1 The Agent-Environment Interface

Reinforcement Learning (RL) has been identified as a promising approach to learn from incremental experience and discover successful decision-making and control policies. The reinforcement learning problem is meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The agent learns from the environment and makes decisions. Everything the agent can interact with (outside the agent) is called the environment. These interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent. The environment also gives rise to rewards, special

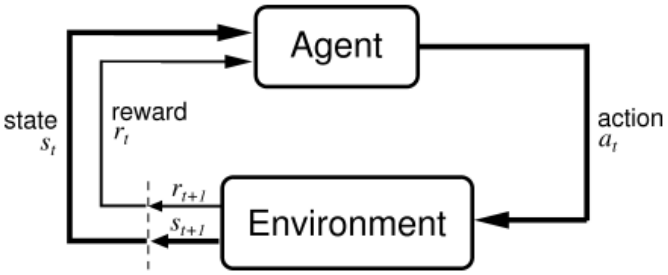


Figure 2.3: The agent-environment interaction in reinforcement learning adapted from Sutton and Barto (1998).

numerical/scalar values that the agent tries to maximize over time. A complete specification of an environment defines a task, one instance of the reinforcement learning problem (Sutton and Barto (1998)).

Characteristics of RL:

- The environment responds to the agent's actions and presents new situations.
- Actions may have long term consequences.
- There is no supervisor, only a scalar reward signal.
- Feedback (i.e., rewards) could be temporally delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)

2.2.2 RL Notation

Classical reinforcement learning approaches are based on the assumption that we have a Markov Decision Process (MDP) consisting of the set of states S , set of actions A, the rewards R and transition probabilities $P^a_{ss'}$ that capture the dynamics of a system. Markov decision processes, and the associated dynamic programming (DP) methodology (Sutton and Barto (1998)), provide a general framework for posing and analyzing problems of sequential decision making under uncertainty.

State s_t: Immediate information available to the agent at the current time-step t

Action a_t: Choice made by the agent from some set, based on current state, which in general affects the next state and reward received

Reward r_t: Scalar value supplied by the environment after each action, the agent tries to maximize the cumulative reward (also called return)

Policy π: Probability distribution defining probability of taking each action from each state $\pi(a|s)$

Value $V(s)$: Expected sum of rewards received by the agent if they start from state S and then follow policy π (i.e., "goodness" or "badness" of a state)

Return R_t: Some specific function of the reward sequence (for e.g., $R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$)

The agent and environment interact at each discrete time step (t = 0, 1, 2, 3...). At each time step the agent receives some representation of the environment's state, $s_t \epsilon \mathcal{S}$, where $\mathcal{S}$ is the set of possible states, and on that basis selects an action, $a_t \epsilon \mathcal{A}_t$, where $\mathcal{A}_t$ is the set of actions available in state s_t. One time step later, in part as a consequence of its action, the agent receives a numerical reward, $r_{t+1} \epsilon \mathcal{R}$, and finds itself in a new state s_{t+1}.

The purpose or goal of the agent is formalized in terms of a special reward signal passing from the environment to the agent. At each time step, the reward is a simple number, $r_t \epsilon \mathcal{R}$. Informally, the agent's goal is to maximize the total amount of reward it receives. This means maximizing not immediate reward, but cumulative reward in the long run. The field of reinforcement learning is primarily the study of methods for tackling this challenge.

A reinforcement learning agent chooses actions according to a policy $\pi(a|s)$ which is a probability distribution over all possible actions for the current state. The policy may be deterministic or stochastic.

A wide range of RL algorithms are based on estimating *value functions*—functions of states (or of state-action pairs) that estimate the *goodness or badness* of state. The notion of *goodness or badness* here is defined by the future rewards to be expected (i.e., expected return). The agent tries to choose actions such that the sum of the discounted rewards it receives over the future is maximized (i.e., it chooses action a_t to maximize the expected discounted return).

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

The rewards the agent can expect to receive in the future depend on the policy it follows (the action/sequence of actions) it will take. Accordingly, value functions are defined with respect to particular policies.

$$V_\pi(s) = \mathop{\mathbb{E}}_{\pi}\{R_t | S_t = s\} = \mathop{\mathbb{E}}_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

where $\mathbb{E}_\pi$ denotes the expected value given that the agent follows policy π, and t is any time step, γ is a discount factor indicating how much more to credit immediate reward than long term reward [2]. The value of the terminal state, if any, is always zero. We call the function $V_\pi(s)$the *state-value function* for policy π . For a given problem we define the optimal policy π^* as that which produces the highest value in every state. [3]

Similarly, we define the value of taking action a in state s under a policy π, denoted $Q_\pi(s, a)$, as the expected return starting from s, taking the action a, and thereafter following policy π:

$$Q_\pi(s, a) = \mathop{\mathbb{E}}_{\pi}\{R_t | S_t = s, a_t = a\} = \mathop{\mathbb{E}}_{\pi}\{\sum_{k=0}^{\inf} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

We call $Q_\pi(s, a)$ the *action-value function* for policy π.

2.2.3 Value Function Approaches (Critic-Only Methods)

One of the main challenges in RL is to make predictions about the (discounted) sum of future rewards (i.e., return), in an initially unknown environment. With temporal difference (TD) learning it is possible to learn good estimates of the expected return quickly by bootstrapping from other expected-return estimates.

[2]This is generally necessary to ensure that reward is finite if the agent-environment interaction continues indefinitely. However it may be omitted if the interaction ends in bounded time (e.g., in a game of chess)

[3]Note that neither $V_\pi(s)$ or $V^*(s)$ are computationally tractable to compute in general, however it is the task of a wide variety of RL algorithms to estimate them from the agent's experience.

Temporal difference (TD) learning uses changes or differences in predictions over successive time steps to drive the learning process. It learns how to predict a quantity that depends on future values of a given signal. The prediction at any given time step is updated to bring it closer to the prediction of the same quantity at the next time step. It can be thought of as a supervised learning process in which the training signal for a prediction is a future prediction. TD algorithms are often used in reinforcement learning to predict the expected return or any other relevant measure.

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.1}$$

The TD error (shown in equation 2.1) is the difference between the old estimate and a new estimate of the value function (taking into account the reward received in the current sample). These updates are done iteratively and, in contrast to dynamic programming (DP) methods, only take into account the sampled successor states rather than the complete distributions over successor states. Like the Monte Carlo methods, these methods are model-free, as they do not use a model of the transition function to determine the value function. Like DP, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome (they bootstrap). The relationship between TD, DP, and Monte Carlo methods is a recurring theme in the theory of reinforcement learning (Sutton and Barto (1998)).

Bootstrapping is an efficient way to update predictions, however, it can have high bias due to errors from inaccurate estimates. On the other hand, Monte Carlo methods update their estimates only when the final / terminal state has been reached (i.e., updates are only performed at the end of an episode). Monte Carlo methods can have high variance in stochastic domains since they are sensitive to variations in the return. The TD(λ) algorithm described in 1 uses the parameter $0 \leq \lambda \leq 1$ to take an intermediate approach that lies between full bootstrapping ($\lambda = 0$) and Monte Carlo ($\lambda = 1$) updates. This results in a trade-off between bias and variance. The λ refers to the use of an *eligibility trace*. An *eligibility trace* is a temporary record of the occurrence of an event, such as visiting of a state or taking an action. The trace marks additional memory variables associated with each event as eligible

for learning updates. When a TD error occurs, only the eligible states or actions are assigned credit or blame for the error as shown in equation 2.2 (Sutton and Barto (1998)).

$$e_t(s) = \begin{cases} \lambda e_{t-1}(s) & if\ s \neq s_t \\ \lambda e_{t-1}(s) + 1 & if\ s = s_t \end{cases} \tag{2.2}$$

Function approximation is critical in every RL problem, especially in continuous state-action spaces. In large discrete spaces it is often impractical to visit or even represent all states and actions. In these settings, function approximation can be used as a means to generalize to neighboring states and actions. In this thesis, we only look at parametric function approximators [4] like linear basis functions and neural networks. Linear approximators are computationally efficient and I will solely using only linear approximators throughout the thesis.

$$V(s) = w^T \phi(s) \tag{2.3}$$

The key idea of the TD(λ) algorithm (see Alg. 1) is to bootstrap the value of a state from the subsequent values many steps into the future. The value function is parameterized by a column vector with fixed number of real valued components w as shown in equation 2.3. The value function for all states is updated in proportion to both the TD-error and the eligibility of the state:

$$w_{t+1} = w_t + \alpha \delta_t e_t(s) \tag{2.4}$$

Gradient-descent methods are the most popular of all the function approximation methods and are particularly well suited to RL. Here, we try to minimize the mean squared TD error on the observed examples. This yields the general gradient-descent methods for state-value prediction:

$$w_{t+1} = w_t + \alpha[r_{t+1} + V_t(s_{t+1}) - V_t(s_t)]\nabla_{w_t} V_t(s_t) \tag{2.5}$$

[4] A parametric function approximator uses a finite set of parameters or arguments where the goal is to find parameters that make this approximation fit the observed data as closely as possible.

where α is the usual step size parameter used in gradient descent and hence is subject to the same convergence constraints. That is, for a deterministic environment, a decaying α is required in order to converge on the solution with the smallest error. In practice, a small fixed value for α is commonly used, which allows predictions to adapt to changing conditions.

Algorithm 1 Temporal Difference Learning: $TD(\lambda)$

procedure $TD(\lambda)$ ▷ Learn to predict values
  Initialize: s, $\phi(s)$ and weights w, e
  for *each step* **do**
    $a \leftarrow$ action given by π for s
    Take action a in state s and observe the next state s' and reward r
    $\phi(s') \leftarrow tilecode(s')$ ▷ Tile Coding (Sutton and Barto (1998))
    $\delta \leftarrow r + \gamma w^T \phi(s') - w^T \phi(s)$
    $e \leftarrow \gamma \lambda e + \phi(s)$
    $w \leftarrow w + \alpha \delta e$
    $\phi(s) \leftarrow \phi(s')$

Value function approximation methods may succeed in constructing a "good" approximation of the value function yet lack reliable guarantees in terms of near-optimality of the resulting policy (Konda and Tsitsiklis (2003)).

Let's consider an example grid world (as shown in Fig. 2.4). In this grid world, each state has four actions namely up, down, left and right. The world has two terminal states, a goal (cheese) and a pit (shock), which return rewards of $+1$ and -1 respectively. Actions that do not lead to the terminal state result in a reward = 0. A wall was added to the grid which blocks the movement of the agent into that location. If the agent took an invalid action, such as moving into the boundary or wall, the agent did not move and simply remained in the same position. We used TD(0) for value estimation with $\gamma = 1$ and $\alpha = 0.1$ over a random action policy and the results can be seen in Fig. 2.4.

2.2.4 Policy Gradient Framework

In policy gradient reinforcement learning, we update the parameters of the agent's policy $\pi_\theta(s, a)$by gradient ascent and adjust the policy to maximize the expected reward. Policy gradient methods are typically higher variance and therefore less

Figure 2.4: This figure shows the learned values over multiple iterations using Temporal Difference (TD) learning without eligibility traces in this grid world. The grid world GUI was adapted from the Berkeley Pacman Project.

efficient compared to value-based approaches, but they have a few significant advantages.

Why Approximate Policies Instead of Values?

The value-function approximation approach has worked well in numerous applications, but has several limitations. The formulation of the problem in terms of policy rather then value offers many features relevant to robotics. It allows for a natural integration of expert knowledge, e.g., through both structure and initialization of the policy. It allows the transfer of domain-specific knowledge by pre-structuring/initialization of the policy in an approximate form without changing the original problem. Optimal policies often have many fewer parameters than optimal value functions. For example, in linear quadratic control, the value function has quadratically many parameters in the dimensionality of the state-variables while the policy requires only linearly many parameters (Kober and Peters (2012)).

Value function methods are indirect in the sense that they do not optimize directly over a policy space. They have an implicit policy (e.g., ϵ-greedy policy) which finds deterministic policies, whereas the optimal policy is often stochastic, selecting different actions with specific probabilities (e.g., Rock-Paper-Scissors, bluffing, POMDPs, etc.).

An arbitrarily small change in the estimated value of an action can cause it to be, or not be, selected. Such discontinuous changes have been identified as a key obstacle to establishing convergence assurances for algorithms following the value-

function approach (Sutton et al. (1999)).

Policy based RL is an optimization problem where we aim to find the best possible policy parameters θ that maximizes the average reward obtained per timestep. One approach is to use derivative-free optimization methods, such as the cross-entropy method (CEM) and covariance matrix adaptation (CMA), which treat the return as a black box function to be optimized in terms of the policy parameter (Szita and Lörincz (2006)). However, greater efficiency is often possible using gradient-based methods (e.g., gradient descent, conjugate gradient, quasi-newton, etc.). In the policy gradient approach, the policy parameters are updated approximately proportional to the gradient:

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta}$$

where α is a positive-definite step size. If the above can be achieved, then θ can usually be assured to converge to a locally optimal policy in the performance measure ρ. Unlike the value-function approach, here small changes in θ can cause only small changes in the policy and in the state-visitation distribution (Sutton et al. (1999)).

The REINFORCE algorithm (Williams (1992)) updates the parameters of the policy by stochastic gradient ascent. Given a differentiable policy $\pi_\theta(s, a)$ that is parameterized by a vector of adjustable weights θ, the REINFORCE algorithm adjusts the weights at each timestep as follows:

$$\Delta\theta = \alpha(R_t - b(s_t))log\nabla_\theta \pi_\theta(s, a) \tag{2.6}$$

where α is the step-size parameter and b is the *reinforcement baseline* that does not depend on the current action a_t.

Sutton et al. (1999) extended the this approach to use the action-value function is place of the actual return.

$$\Delta\theta = \alpha(Q_\pi(s_t, a_t) - b(s_t))log\nabla_\theta \pi_\theta(s, a) \tag{2.7}$$

2.2.5 Value Function Methods vs Policy Search

Some methods attempt to find a value function or policy which eventually can be used without significant further computation, whereas others (e.g., roll-out methods) perform the same amount of computation each time. If the optimal value function is known/learned, a globally optimal solution follows simply by greedily choosing actions to optimize it. However, the translation of value-function based approaches to high dimensional robotics have thus far been difficult since they require function approximation for the value function. Most theoretical guarantees no longer hold for this approximation (Kober and Peters (2012)). R.S. Sutton suggests that the risk of divergence arises whenever we try to combine three things: (i) Function Approximation (ii) Bootstrapping [5] and (iii) Off-policy learning [6]. Given that we have limited experience or high cost of exploration in the real world, we might want to try combining the above three things in order to fully utilize the state information we obtain for learning.

In principle, a value function requires a thorough sweep through the entire state space and the largest local error determines the quality of the resulting policy. A particularly significant problem is the error propagation in value functions. A small change in the policy may cause a large change in the value function, which again causes a large change in the policy. While this may lead more quickly to good, possibly globally optimal solutions, such learning processes often prove unstable under function approximation (Kakade and Langford (2002)) and are considerably more dangerous when applied to real systems where large/extreme policy deviations may lead to dangerous decisions (Kober and Peters (2012)).

In contrast, policy search methods usually only consider the current policy and its neighborhood in order to gradually improve performance. The policy gradient approach is only guaranteed to converge to a locally optimal policy (Sutton et al. (1999)). However, these methods have worked well in conjunction with continuous

[5] Learning value estimates from other value estimates, as in Dynamic Programming or Temporal Difference learning

[6] An off-policy learner learns the value of the optimal policy independently of the agent's actions (For e.g., Q-learning is an off-policy learner). An on-policy learner learns the value of the policy being carried out by the agent, including the exploration steps.

features. The local coverage (i.e.,explored parts of explored state space) and local errors results into improved scale-ability in robotics (Peters and Schaal (2006)).

2.2.6 Actor-Critic Methods

Actor-critic algorithms combine the advantages of policy gradient methods with the efficiency of value-based reinforcement learning. They consist of two components: an actor that updates the agent's policy, and a critic that updates the action value function. When value function approximation is used, care must be taken to ensure that the critic's parameters w are compatible with the actor's parameters θ. The compatibility requirement is that $\nabla_w Q_w(s,a) = \nabla_\theta log \pi_\theta(s,a)$.

Policy search methods are sometimes called actor-only methods; value function methods are sometimes called critic-only methods. The idea of a critic is to first observe and estimate/measure the performance of taking an action on a given state (i.e., the value function), then learn a policy based on the knowledge gained. In contrast, the actor directly tries to deduce the policy.

Actor-only methods work with a parameterized family of policies.

- The gradient of the performance, with respect to the actor parameters, is estimated and the parameters are updated in a direction of improvement .
- Gradient estimators may have a large variance.
- As the policy changes, a new gradient is estimated independently of past estimates.

Critic-only methods rely exclusively on value function approximation.

- Aim at learning an approximate solution to the Bellman equation, which will then hopefully prescribe a near-optimal policy.
- They are indirect in the sense that they do not try to optimize directly over a policy space.
- They may succeed in constructing a "good" approximation of the value function yet lack reliable guarantees in terms of near-optimality of the resulting policy.

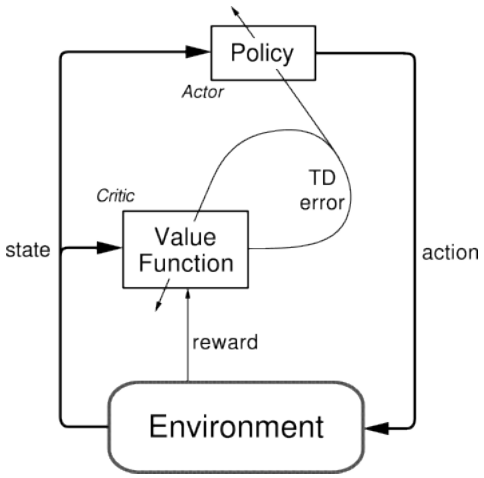


Figure 2.5: Actor-Critic Architecture.

Actor-critic methods aim at combining the strong points of Policy Gradient and Value Function Approximation methods. Actor-critic methods have a separate memory structure to explicitly represent the policy independent of the value function. The policy structure is known as the actor, because it is used to select actions, and the estimated value function is known as the critic, because it criticizes the actions made by the actor. The critique takes the form of a Temporal Difference (TD) error. This scalar signal is the sole output of the critic and drives all learning in both actor and critic (Sutton and Barto (1998)). The resulting update step features the local convergence properties of policy gradient algorithms while reducing update variance (Konda and Tsitsiklis (2003)). There is a trade-off between the benefit of reducing the variance of the updates and having to learn a value function as the samples required to estimate the value function could also be employed to obtain better gradient estimates for the update step. The algorithm is explained in detail in Section 5.2.

Konda and Tsitsiklis (2003) proposed some actor-critic algorithms in which the critic uses linearly parameterized approximations of the value function, and provide a convergence proof for the same. The algorithms are based on the following important observation: Since the number of parameters that the actor has to update is relatively small (compared to the number of states), the critic need not attempt to compute or approximate the exact value function, which is a high-dimensional object. In fact, it is shown that the critic should ideally compute a certain "projection"

of the value function onto a low-dimensional subspace spanned by a set of "basis functions," which are completely determined by the parameterization of the actor. This key insight was also derived in simultaneous and independent work (Sutton et al. (1999)) that also included a discussion of certain actor-critic algorithms. 11

The analysis by Tsitsiklis and Van Roy (1997) shows that temporal difference (TD) learning algorithms try to compute the projection of an exact value function onto a subspace spanned by feature vectors. Actor-Critic methods developed by Konda and Tsitsiklis (2003) use TD learning for the critic. [7]

2.2.7 Actor Critic Reinforcement Learning (ACRL)

The policy gradient algorithms mentioned above are independent of the structure of the policy distribution used. For discrete actions, the Gibbs distribution is often used. In this thesis, for continuous actions, we define the policy such that actions are taken according to a normal distribution, as suggested by Williams (1992), with probability density function defined as

$$\mathcal{N}(s,a) = \frac{1}{\sqrt{2\pi\sigma^2(s)}} exp(-\frac{(a-\mu(s))^2}{2\sigma^2(s)})$$

where $\mu(s)$ and $\sigma(s)$ are the mean and standard deviation respectively of the policy distribution $\pi(.|s)$.

As suggested by Degris et al. (2012), our policy is parameterized by the scalars $\mu(s) = u_\mu^T x_\mu(s)$ and $\sigma(s) = exp(u_\sigma^T x_\sigma(s))$. These scalars are parameterized using linear combinations where $u = (u_\mu^T, u_\sigma^T)^T$ and the feature vector in state s is $x_u(s) = (x_\mu(s)^T, x_\sigma^T)^T$.

The compatible features $\frac{\nabla_u \pi(a|s)}{\pi(a|s)}$ depend on the structure of the probability den-

[7] **Note:** The actor takes decisions about what to do in the world. The critic evaluates the actions that are taken by the policy (i.e., the actor in this case). In this sense, the actor can be considered as Policy improvement step and the critic as a form of policy evaluation. Together the Actor-Critic methods could be considered as a form of Generalized Policy Iteration as described by Sutton and Barto (1998).

sity of the policy. Given that we use a Normal Distribution [8],

$$\frac{\nabla_{u_\mu}\pi(a|s)}{\pi(a|s)} = \frac{1}{\sigma(s)^2}(a-\mu(s))x_\mu(s) \tag{2.8}$$

$$\frac{\nabla_{u_\sigma}\pi(a|s)}{\pi(a|s)} = \Big(\frac{(a-\mu(s))^2)}{\sigma(s)^2} - 1\Big)x_\sigma(s) \tag{2.9}$$

where $\frac{\nabla_u \pi(a|s)}{\pi(a|s)} = \big(\frac{\nabla_{u_\mu}\pi(a|s)}{\pi(a|s)}^T, \frac{\nabla_{u_\sigma}\pi(a|s)}{\pi(a|s)}^T\big)^T$.

The compatible feature in equation 2.8, used to update the parameters u_μ of the policy, has a $\frac{1}{\sigma(s)^2}$ factor: a smaller estimate of the standard deviation results in a larger norm of $\frac{\nabla_{u_\mu}\pi(a|s)}{\pi(a|s)}$ and vice versa. We observed that such an effect can cause instability, particularly because $\lim_{\sigma\to 0}\frac{(a-\mu(s)}{\sigma(s)^2} = \infty$

As suggested by Degris et al. (2012) and Williams (1992) we use a step size of the form $\alpha_u\sigma^2$ for the gaussian distribution, scaling the gradient with respect to the variance of the distribution.

Algorithm 2 Actor Critic Reinforcement Learning

procedure ACRL ▷ Learn a control policy for joint j
  Initialize: s, $x(s)$ and weights e_v, v, e_μ, w_μ, e_σ, w_σ
  for *each step* **do**
    $a_j \leftarrow \mathcal{N}(\mu, \sigma^2)$
    Take action a in state s and observe the next state s' and reward r_j
    $x(s') \leftarrow tilecode(s')$ ▷ Tile Coding (Sutton and Barto (1998))
    $\delta \leftarrow r_j + \gamma v^T x(s') - v^T x(s)$
    $e_v \leftarrow \gamma\lambda e_v + x(s)$
    $v \leftarrow v + \alpha_v \delta e_v$
    $e_\mu \leftarrow \gamma\lambda e_\mu + (a-\mu)x(s)$
    $w_\mu \leftarrow w_\mu + \alpha_\mu \delta e_\mu$
    $e_\sigma \leftarrow \gamma\lambda e_\sigma + ((a-\mu)^2 - \sigma^2)x(s)$
    $w_\sigma \leftarrow w_\sigma + \alpha_\sigma \delta e_\sigma$
    $x(s) \leftarrow x(s')$

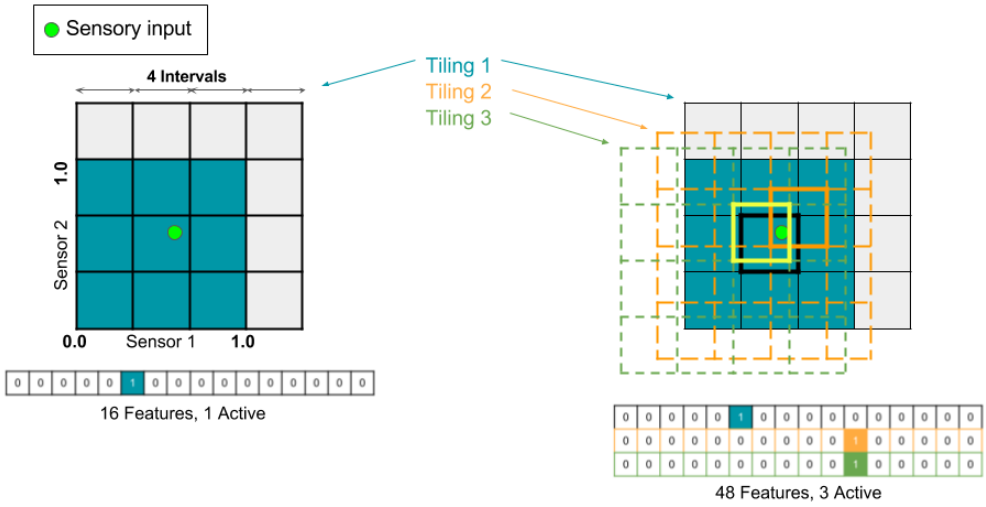


Figure 2.6: Tile coding - In the left, a normalized continuous 2D sensor space is tiled using 4 tiles per dimension. Each tile has a width of 0.33 in each dimension. In the right, three offset tilings are placed over the same sensor space, producing 48 features with 3 active at any given instant.

2.2.8 Tilecoding

As the number of dimensions grows, exponentially more data and computation are needed to cover the complete state-action space. Bellman (1957) termed this exponential explosion of states and actions as "Curse of Dimensionality". Decreasing the dimensionality of the state-action space can significantly ease most RL problems, particularly in the context of robotics.

While continuous signals can be simply discretized and used as state inputs, a lot of the structural and overlapping feature information is lost in such a crude discretization. Tile coding is an approach to address learning in continuous domains which diminishes the aliasing effects that discretization can incur. It is used for creating sparse binary features from real-valued signals and is well suited to the online learning paradigm. While tile coding helps us make our learning task tractable, it still suffers from the curse of dimensionality, in terms of the number of features

[8] The easiest way to derive the compatible features is to exploit its mathematical structure, i.e., $\frac{\nabla_{u_\mu}\pi(a|s)}{\pi(a|s)} = \nabla_{u_\mu} log(\pi(a|s))$. Using the logarithmic form leads to an easier calculation of the derivative.

(rather than say in terms of the number of states).

In tile coding (Sutton and Barto (1998)), a piecewise-constant approximation of the optimal value function is represented by a set of exhaustive partitions of the state-space called tilings. Typically, the tilings are all partitioned in the same way but are slightly offset from each other. Each element of a tiling, called a tile, is a binary feature activated if and only if the given state falls in the region delineated by that tile. Figure 2.6 illustrates a tile-coding scheme with three tilings.

Tilings need not be square in shape and need not have the same resolution across all sensors as shown in Fig 2.6. They can be shaped and distributed to promote particular kinds of generalization. Also, they can be used be used for any number of dimensions. In a sense, tile coding is a form of hand-crafted discretization which incorporates human-domain knowledge in its representation.

When multiple tilings are used, each is offset by a different amount, so that each cuts the state-space in a different way. The width and shape of the tiles should be chosen to match the width of generalization that one requires or deems necessary. The number of tilings should be chosen to influence the density of tiles. The number of tilings directly influence the resolution of the final function approximation. Using a large number of tilings implies that the desired function can be approximated more accurately, but with greater computational costs.

2.3 Prosthetic Control as a Robot Control Problem

A wide variety of control schemes that are currently being used for robotics applications can be extended to the control of prosthetic devices. This results from the fact that both robotic and prosthetic systems contain several independent DOFs that must be coordinated to perform a particular task.

The developments of control systems for robots have advanced relatively faster than that for robotic prostheses. This is perhaps a result of the inherent difficulties of interfacing with the human in prosthetic systems to generate reference command/control signals and to derive feedback from various sensory resources. Also, constraints on the system due to stringent size, power and cosmetic specifications

can cause additional design problems. Some of the characteristics that are common to both robotics and prosthetic domains follow (Orin (1980)):

- Force, position and compliance must be controlled
- Feedback of multiple sensory information is essential in complex tasks
- The trajectory or desired movement can be specified as Cartesian coordinates while actuation occurs in relative joint coordinates
- Good performance over wide range of motion and loads is difficult since it is dependent upon complex, time-varying, non-linear dynamics.

We can think of prosthetic arms as assistive robots that directly and physically interact with humans in order to help them achieve a particular goal. In this thesis, I focus on the control of prosthetic arms using RL techniques. It is important to note that in such a setting, the user becomes a part of the environment.

While this introduces a different set of problems, I believe that in the long run these methods could give rise to robust, generalizable solutions.

Chapter 3

Learning from Demonstration: Teaching a Prosthetic Arm Using an Intact Arm via Reinforcement Learning

3.1 Overview

Prosthetic arms should restore and extend the capabilities of someone with an amputation. In other words, that will restore near-natural motor and sensory capability to upper-extremity amputee patients. They should move naturally and be able to perform elegant, coordinated movements that approximate those of a biological arm. Despite these objectives, the control of modern-day prostheses is often non-intuitive and taxing. Existing devices and control approaches do not yet give users the ability to effect highly synergistic movements during their daily-life control of a prosthetic device. As a step towards improving the control of prosthetic arms and hands, we introduce an intuitive approach to training a prosthetic control system that helps a user achieve hard-to-engineer control behaviours. Specifically, we present an actor-critic reinforcement learning method that for the first time promises to allow someone with an amputation to use their non-amputated arm to teach their prosthetic arm how to move through a wide range of coordinated motions and grasp patterns. We evaluate our method during the myoelectric control of a multi-joint robot arm by non-amputee users, and demonstrate that by using our approach a user can train their arm to perform simultaneous gestures and movements in all three de-

grees of freedom in the robot's hand and wrist based only on information sampled from the robot and the user's above-elbow myoelectric signals. Our results indicate that this learning-from-demonstration paradigm may be well suited to use by both patients and clinicians with minimal technical knowledge, as it allows a user to personalize the control of his or her prosthesis without having to know the underlying mechanics of the prosthetic limb. These preliminary results also suggest that our approach may extend in a straightforward way to next-generation prostheses with precise finger and wrist control, such that these devices may someday allow users to perform fluid and intuitive movements like playing the piano, catching a ball, and comfortably shaking hands.

3.2 Introduction

In order to achieve simultaneous, coordinated movement, every DOF of the prosthetic arm needs to be supplied with appropriate motor commands at every time step. The commands must be chosen such that they accomplish the desired task, but also such that they do not violate any safety or physical constraints of the system. Due to the numerous DOFs in complex movement systems and the almost infinite number of possibilities to use the DOFs over time, there actually exist an infinite number of possible trajectories for any given task. This redundancy is advantageous as it allows a system to avoid situations where, for instance, the range of motion of DOFs is highly restricted, or where obstacles need to be circumvented to reach a goal. But it also makes it quite complicated to learn good policies since the state spaces spanned by all possible trajectories is extremely large. We need to make learning tractable in such high dimensional systems using some constraints that can reduce the state-space in a reasonable way without eliminating good solutions.

Bernstein (1967) suggested that humans do not control elementary degrees of freedom but use *synergies*. Synergies are patterns of usage that are functionally advantageous. For example, in extending the hand to a target, the actor might tie the activities of certain muscles together to produce a particular synergy that would move the hand reliably in a particular direction. This simplifies control because

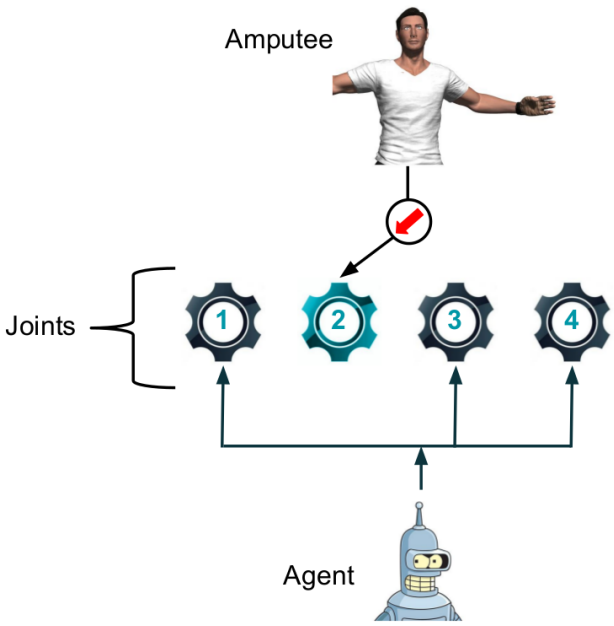


Figure 3.1: The collaborative control scheme used in this chapter. The user can control few selective joints simultaneously while the robot, in this case the RL agent, controls all other joints. The joints are represented by the gears and controlled using velocity commands.

the actor has to control only the set of synergies available and not the individual muscles to produce reliable movement (Berthier et al. (2005)).

The learning from demonstration (LfD) paradigm offers a nice framework to exploit the learned synergies of the human user. While RL provides a nice framework to achieve hard-to-engineer behavior, it is frequently hard to provide a good reward function manually. Consider, for example, the task of serving in tennis. When serving the ball, we optimize many different desiderata, such as the force with which hit the ball needs to be hit, angle of the joints depending on the ball's target location and hitting the ball without straining the arm. To define a reward function for this task is rather difficult. Furthermore, building the policy requires gathering information by visiting states to receive rewards, which is non-trivial for a robot learner executing actual actions in the real world. While it is hard to define the "tennis serve" as an RL task that learns only from trial and error, it is far easier to demonstrate the task to a robot arm and have it learn from expert demonstrations.

Demonstration based learning techniques have also been called as imitation

learning, apprenticeship learning, behavioral cloning and programming by demonstration. LfD formulations typically do not require expert knowledge of the domain dynamics, which removes performance brittleness resulting from model simplifications. In a clinical setting, this means that patients and clinicians can communicate their requirements to the robot without actually knowing the underlying mechanics of the system. This is increasingly useful in a world where robots become more commonplace. Furthermore, demonstration has the attractive feature of being an intuitive medium for communication from humans, who already use demonstration to teach other humans. Demonstration also has the practical feature of focusing the dataset to areas of the state–space actually encountered during task execution.

In this chapter, we are essentially looking at a collaborative control approach (see Fig. 3.1) where the robot learns a policy which is dependent on the control commands provided by the user and its own sensorimotor information.

3.3 Methods

A myoelectric prosthesis can be thought of as a wearable robot that responds to sEMG control signals. A myoelectric user is faced with the task of interacting with a robot to accomplish everyday tasks. It is reasonable to expect that most people with amputations may not be robotics experts, but they could have ideas of what their prosthesis should do, and therefore what types of synergies their prosthetic control algorithms should give rise to. A natural and practical extension of having this knowledge is to use it to develop their own desired control algorithms. However, unlike the usual practice of directly engineering a control approach, we suggest that desired behaviours could be learned by the prosthesis from demonstrations provided by the user.

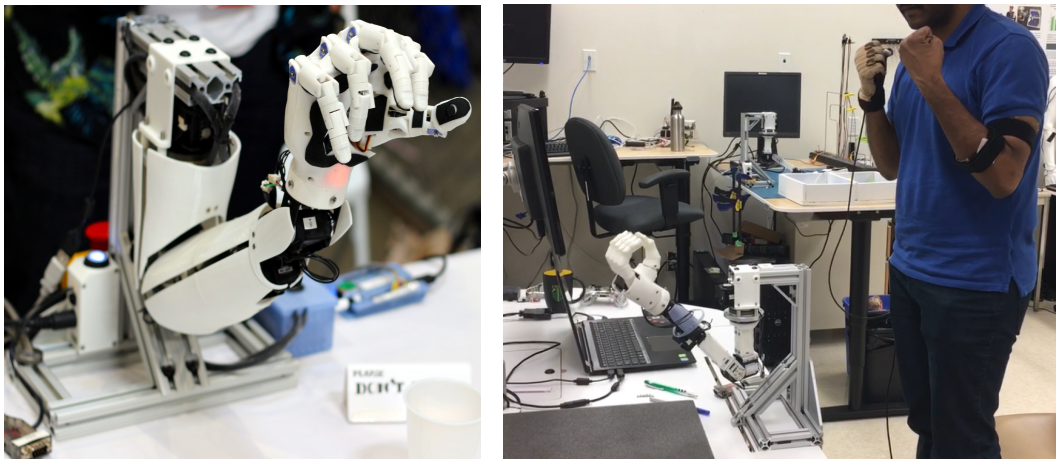
In the present work, we specifically address the common case of a user with a unilateral, transhumeral amputation—someone missing their hand, forearm, and elbow. In this setting, the user has one biological limb, and one robotic limb that they wish to train to appropriately respond to the commands being generated by the muscle tissue in the user's residual limb. We refer to the arm generating the

control signals as the user's *control arm*. For someone with an amputation, these control signals would come from the residual limb that is attached to their robotic prosthesis, where EMG signals from residual biceps and triceps may be already used in the direct control of their robotic elbow. We term the arm providing the training movements the *training arm*, or the contralateral, intact biological limb.

By asking the user to perform the *same motion* with both arms (or visualize performing, in the case of an amputated control limb, and as in pattern recognition training (Scheme and Englehart (2011)), we suggest that the motion of the training limb can provide training information for creating a prosthetic policy that maps the state of the control limb (e.g., gross robot limb position and control-limb EMG signals) to motor commands for the remaining joints of the prosthetic hands and wrist not controllable by the user. The robotic prosthesis can then use its learned, state-conditional policy to "fill in the gaps" for the user during ongoing, post-training use.

For this study, we first explore prosthetic LfD with able-bodied participants. In the case of these able-bodied subjects, the control arm is defined as the arm providing the control signals to a robot limb, where control channels are sampled in the same locations as they would be for someone with an amputation. The training arm is their contralateral limb, and represents what would be the user's non-amputated arm.

Robotic Arm: Our experiments were done via an open-source robot platform known as the Bento Arm, as shown in Fig. 1. The Bento Arm is a myoelectric training tool to assess and train upper-limb amputees in how to use their muscle signals prior to being fit with a myoelectric prostheses (Dawson et al. (2014)). Although designed to be donned via a socket, for repeatability in this experiment the Bento Arm was rigidly fixed to a desk directly in front of the able-bodied subject (Fig. 1b), such that its arm position was aligned with the control-delivering arm of the subject. The myoelectric control system received the angular position and velocity of the following joints from the Bento Arm: elbow $\langle \theta_e, \dot{\theta}_e \rangle$, wrist flexion/extension $\langle \theta_{wf}, \dot{\theta}_{wf} \rangle$, wrist rotation $\langle \theta_{wp}, \dot{\theta}_{wp} \rangle$, and aperture angle of the gripper hand $\langle \theta_h, \dot{\theta}_h \rangle$.



(a) The Bento Arm (b) Experimental Setup

Figure 3.2: Experimental setup which includes the Bento Arm, Delsys Trigno Wireless Lab and CyberTouch II. The Bento Arm as used in our trials had 5 active DoFs including shoulder rotation, elbow flexion/extension, wrist pronation/supination, wrist flexion/extension and hand open/close.

EMG Data Acquisition: We used a 16-Channel Delsys Trigno Wireless Lab (Delsys, Inc.) to record EMG signals from our subjects. As shown in Fig. 1b, able-bodied subjects were fitted with four Delsys Trigno units that provided sEMG signals and inertial measurements from accelerometers, magnetometers and gyroscopes. The Trigno units were placed on the control arm of each subject as follows: two units on the biceps and two units on the triceps. We placed one additional inertial measurement unit (IMU) on the training arm's wrist to measure the desired wrist rotation angle (θ^*_{wp}).

Motion Capture Glove: The desired joint angle configurations for wrist flexion/extension (θ^*_{wf}) and hand open/close (θ^*_h) were defined by the subject using a CyberTouch II system (CyberGlove Systems LLC) worn on the hand of their training arm. The CyberTouch (shown in Fig. 1b) uses resistive bend-sensing technology to accurately transform hand and finger motions into real-time digital joint-angle data (18 high-accuracy joint-angle measurements). When this data was coupled with that from the single IMU on the training wrist, the subject was able to use the movement of their training arm to precisely specify their desired pose for the robot arm's hand and wrist.

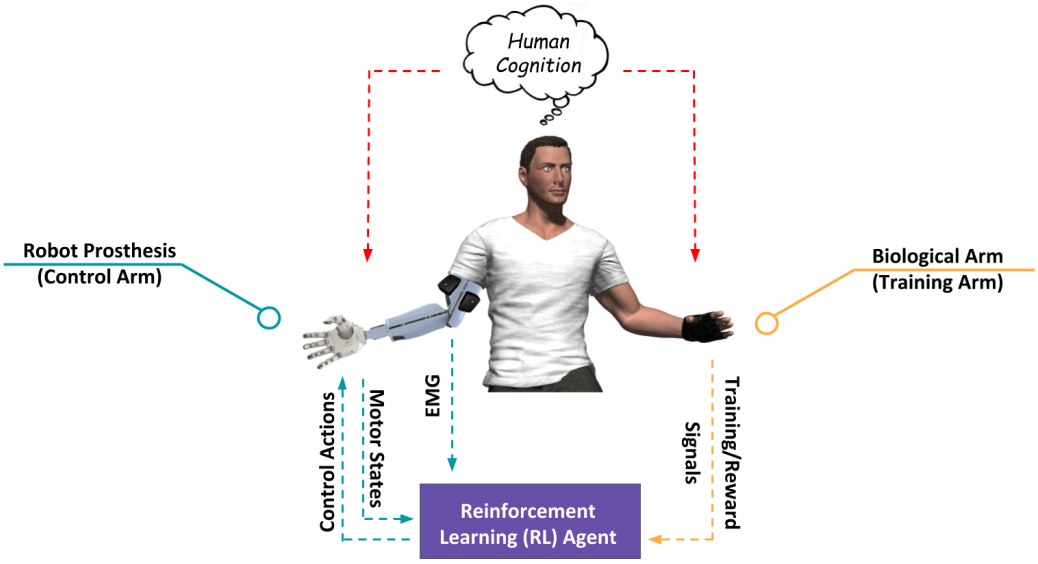


Figure 3.3: Schematic showing the flow of information through the experimental setup during the training period.

Phase I: Recording Training Data

In this phase, subjects were instructed to execute a repetitive sequence of simple reaching and grasping movements that were mirrored by both their control and training arms (for someone with an amputation, this would correspond to trying to perform identical movements using their non-amputated arm and the prosthetic arm). The training arm demonstrated the desired movement and grasp pattern to the prosthetic arm. During training, the elbow of the Bento Arm was actuated via proportional myoelectric control from the subject's control arm, while to wrist and hand of the Bento Arm were actuated via direct teleoperation—i.e., the Bento Arm copied the training arm's movements as reflected to the contralateral side. As shown in Fig. 2 and described above, we recorded desired angles from the subject's training arm (wrist and finger joints) using the motion capture glove and inertial measurement system.

For our experiment, we chose a simple, repeatable movement as the desired behavior—a bicep-curl motion involving the smooth alternation of 1) supinated hand-closed wrist flexion during elbow flexion, and 2) hand-open pronation with wrist extension during elbow extension. The position of the wrist and hand was correlated to the angular position of the elbow joint, such that any given elbow po-

sition could be uniquely mapped by a policy into a higher dimensional combination of joint motions. Using the specific approach described previously by Pilarski et al. (Pilarski et al. (2013b, 2011, 2013a)), mean-absolute-value signals recorded from antagonistic muscle groups (biceps/triceps) were mapped to joint velocity commands in order to control the elbow joint of the Bento Arm—i.e., the user controlled the elbow joint of the Bento Arm using EMG signals from their control arm using direct proportional control. The subject used their training arm to demonstrate the desired behavior for three joints: wrist flexion/extension, wrist rotation and opening/closing the hand using the motion capture glove (the CyberTouch system) and IMU signals. The CyberTouch was worn on the intact limb and used only during this phase. We recorded all the real-valued data signals we received from the Bento Arm, Delsys Trigno system and the CyberTouch II while the user repeatedly demonstrated the desired behavior to the prosthetic arm. In this phase, no machine learning takes place. We effectively asked the subjects to teleoperate the Bento Arm while demonstrating the desired behavior since seeing the motor outcomes on the Bento Arm was found to help subjects visualize what the prosthetic arm would actually do. We recorded data for $\geq 5mins$ for each user. All subjects ($n = 3$) gave informed consent in accordance with the study's authorization by the University of Alberta Health Research Ethics Board.

Mapping Contralateral Training Hand Demonstrations to Robot Hand Joint Angles: We fixed our frame of reference (3-dimensional euclidean space) relative to the wrist such that every hand movement could be represented as series of rotations along the x, y and z axis (i.e, roll, pitch and yaw respectively). The CyberTouch provided wrist pitch (i.e., flexion/extension) and yaw (i.e., radial/ulnar deviation) angles of the intact limb. We placed an additional Trigno unit on the wrist to capture pronation/suppination (i.e., roll) of the wrist. While the right wrist rotates in a clock-wise direction, the left wrist rotates in an anti-clockwise direction. How-

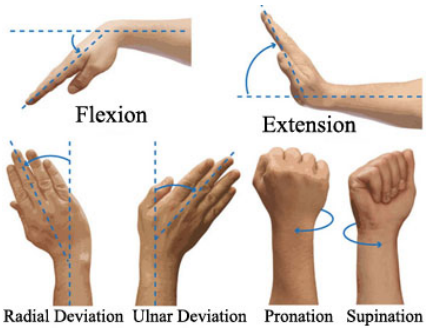


Figure 3.4: Anatomical terms of hand motion.

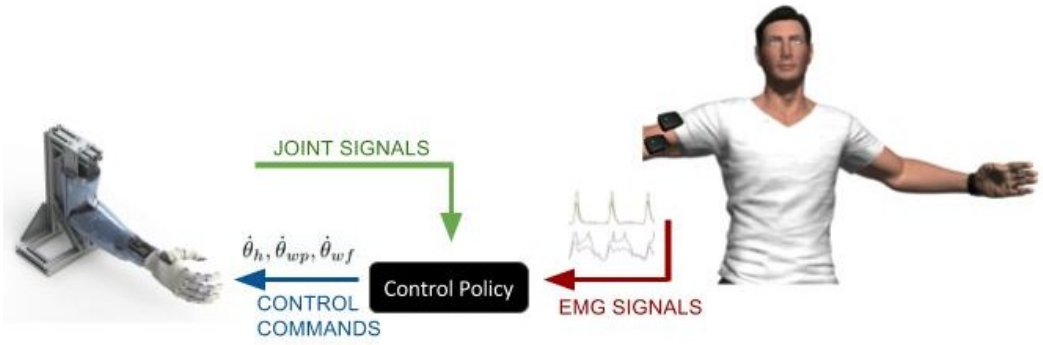


Figure 3.5: Schematic showing the flow of information during deployment. The controller generates new robot joint velocity commands using its learned policy, EMG data from the user, and sensor readings from the robot limb.

ever, the pitch and yaw rotation axes are similar for both arms (relative to our frame of reference). Hence, we used wrist pitch angle of the training arm as target for the Bento Arm. In the case of wrist rotation, the difference between 2π and the joint angle was used as the desired actuator angle. Yaw was not present on the robot and thus not used in this study.

Phase II: Learning a Robot Control Policy

The user was not involved in this phase. A reinforcement learning agent, described below, was tasked with learning and maintaining a control policy for the three target actuators on the robot arm: wrist flexion/extension, wrist rotation and opening/closing the hand. The learned control policy should pick actions such that the actuator's instantaneous position matches the joint configuration demonstrated by the subject using the motion capture glove. We used the pre-recorded data to build a simulator that plays back the user's EMG signals and motion capture signals. The learning agent is given the pre-recorded EMG and CyberTouch signals along with the Bento Arm's real-time joint information (position and velocity) to learn the desired behavior. In essence, the robotic arm trained itself to track the desired trajectory via joint velocity modulation (further details about robot learning are given in Sec. IV). In practice, learning could be conducted once a limb is doffed, e.g., overnight or periods of non-use.

Phase III: Testing the Learned Control Policy

During testing and deployment, a subject used his or her EMG signals from the control limb to move the prosthetic arm, again via conventional direct myoelectric control (as shown in Fig. 3). The subjects were asked to freely actuate the elbow joint of the Bento arm using conventional EMG-based linear-proportional control, and the system would use the learned control policies to move the remaining target joints (i.e., in response to user control choices the system would now effect the synergies learned during the training phase). The non-amputated arm was free to perform any movement. The controller selected appropriate velocity commands at each timestep depending on the EMG signals from the user and sensor readings from the Bento Arm (as shown in Fig. 3). For safety during initial user testing, all joint velocities were bounded by $-2 \leq \dot{\theta}_j \leq 2\ radians/sec.$

3.4 Learning a Control Policy in Real-Time Using Actor-Critic Reinforcement Learning

In order to learn from the demonstrations of the training (non-amputated) arm, we applied actor-critic reinforcement learning (ACRL) as our primary mechanism for LfD policy development. As shown in previous work by Pilarski et al. (2011, 2013b), ACRL is a flexible, online learning framework that can be easily adapted to different application domains and the needs of individual amputees. In particular, reinforcement learning (RL) enables a robot to autonomously discover optimal behavior through trial-and-error interactions with its environment. Instead of explicitly detailing the solution to a problem, in RL the user provides feedback about the performance of the robot in terms of a scalar reward signal. The goal of any RL agent is to maximize the expected cumulative reward (also known as the *return*) (Sutton and Barto (1998)).

ACRL methods in particular are well-suited for the LfD task in the present work since they are model-free, parameter-based incremental learning algorithms which allow fast computation (millisecond updates even over large state spaces) (Pilarski et al. (2011)). In the field of robotics, one of the earliest successes of ACRL was

shown by Benbrahim et al. for biped locomotion (Benbrahim and Franklin (1997)). Peters and Schaal have since applied Natural Actor Critic methods to teach a 7-DoF anthropomorphic robot arm to hit a ball as far as possible (Peters and Schaal (2008b)).

The RL agent chooses control actions denoted a based on the learned policy. The actions, in this case, were real-valued signals which indicate the desired joint velocity. At each time step, the continuous actions a_{wf}, a_{wp}, a_h were taken according to each joint's respective parameterized policy. The actions were drawn from a normal distribution with a probability density function defined as $\mathcal{N}(s,a) = \frac{1}{\sqrt{2\pi\sigma^2(s)}} \exp\Big(- \frac{(a-\mu(s))^2}{2\sigma^2(s)} \Big)$. The parameters of the normal distribution were functions of the system's learned weight vectors w_μ and w_σ as given by $\dot{\theta} \approx a \leftarrow \mathcal{N}\{\mu,\ \sigma\}$. The actions selected by the RL agent were allowed to persist for $\sim 75ms$ to give the robot enough time to execute the control commands and better explore the world. Learning updates occurred every $\sim 40ms$ of the training period.

In our policy parameterization, the scalars $\mu = {u_\mu}^T x(s)$ and $\sigma = \exp({u_\sigma}^T x(s) + \log(\sigma_c))$ were defined as a linear combination of the parameters of the policy and the feature vector of the state $x(s)$. Actor weights w_μ and w_σ were updated based on the compatible features for normal distribution (Degris et al. (2012)). We used accumulating eligibility traces for both the critic (e_v) and the actor (e_μ and e_σ) (Degris et al. (2012)).

The ACRL agent, implemented as described in Pilarski et al. (2013b), was given control of three continuous angular velocities $\dot{\theta}_{wf}, \dot{\theta}_{wp}$ and $\dot{\theta}_h$, where they denote the angular velocities of wrist flexion/extension, wrist pronation/suppination and the gripper hand. Raw EMG signals s were rectified and averaged as $\bar{s} = (1 - \tau)\bar{s} + \tau s$, with a time constant $\tau = 0.037$. Differential EMG was later computed for antagonistic muscle pairs (biceps/triceps), $\bar{s}_1 = \bar{s}_{BI} - \bar{s}_{TRI}$ to control the robot's elbow joint. The following signals were used to construct the state approximation vector for each joint *j* controlled by the learning system $x(s)$: $\langle \bar{s_1}, \theta_e, \dot{\theta}_e, \theta_j \rangle$; where $\theta_e, \dot{\theta}_e, \theta_j$ denote elbow joint angle, elbow joint velocity and current angle of the joint controlled by the robot.

We used tile coding (Sutton and Barto (1998)) to construct the state approxima-

tion vector $x(s)$ used in learning. Our state representation consisted of 25 incrementally offset tilings ($width = 1$) for better generalization. Each tiling had two resolution levels $N_R = [4, 8]$, along with a single baseline unit. This resulted in a binary feature vector of length 108,801 hashed down to a memory size of 2048, with m = 51 active features per step. The learning parameters were set as follows: $\sigma_c = 1$, $\alpha_v = 0.1/m$, $\alpha_\mu = 0.02/m$, $\alpha_\sigma = 0.25\alpha_\mu$, $\gamma = 0.96$ and $\lambda = 0.7$. Weight vectors e_v, v, e_μ, w_μ, e_σ, w_σ were initialized to zero and σ bounded by $\sigma \geq 0.01$.

The ACRL systems was trained incrementally using repeated cycles of the training data earlier recorded in Phase 1, as described in Sec. 2. Total training time was held constant at 45 min after which the learned control policy was tested on a different data set for accuracy. The control learner received negative rewards r on each step proportional to the difference between the target and current joint angles: $r_j = -|\theta_j{}^* - \theta_j|$, in radians. Each controlled joint had its own ACRL learner with its own reward function r_j.

Performance of the learning system was measured based on its ability to achieve desired joint angles. All learning algorithms were run on a Lenovo Flex-3 Laptop with Intel Core i7-6500U @2.50GHz x 4 and 8GB RAM. We used the Robot Operating System (ROS) Kinetic on Ubuntu 16.04 to send and receive information and commands from the Bento Arm, CyberTouch II and the Delsys Trigno Wireless Lab.

3.5 Results

In our experiments, the actuator targets $\theta_j{}^*$ were demonstrated by the user on a moment to moment basis during training. To serve as a baseline performance measure for post-training ACRL policies, we used a reactive control approach (Pilarski et al. (2013b)) as an offline equivalent to direct teleoperation. Since the set of joints targets are not known until the user demonstrates them, a simple baseline teleoperation policy would be to observe the desired joint angle $\theta_j{}^*$ and take an action a_j that moves current actuator angle θ_j towards the target angle as quickly as possible. This control approach assumes perfect knowledge of desired joint angles, (i.e,

the states and targets are fully observable), so is only applicable as a training-data baseline.

Figure 4 shows the quartile analysis of mean absolute angular error accumulated over two minutes of learning/testing for five independent runs for each subject. The ACRL learner showed continuous improvement in performance over learning. Importantly, performance at the end of the learning phase was consistent with performance during actual user control in the testing phase for all three subjects. The negative of the mean absolute angular error is the average reward received during the evaluation period.

Figure 5 shows examples of the joint control trajectories achieved by the ACRL learner during the early and late stages of learning, and during testing. As shown in Fig. 5, the joint angles remained within the target regions for the majority of the evaluation period during both testing and training scenarios. The trajectories achieved by the ACRL learner are compared with direct teleoperation (reactive control as described above). After 20 min of learning the ACRL learner started to achieve the desired joint trajectories, though the controller did visibly overshoot/oscillate around the desired trajectory. The controller started to tightly track the desired trajectory following 30 min of learning. However, as seen in Fig. 5, we observe occasional spikes in the joint angles whenever there is a sudden transition in the desired position. These spikes likely correspond to the fact that less time is spent in training for transitional motion.

3.6 Discussion

3.6.1 Comparison of Performance Between Subjects

Among the 3 subjects, Subject 1 had the most familiarity with the experimental setup. The rest of the subjects had minimal/no experience with the system. As can be seen from Fig. 4, Subject 1 was able to obtain slightly better control performance especially in terms of the variability of the ACRL system's selected control actions. Our observations suggest that improved performance could be achieved with more practice and familiarity with the system.

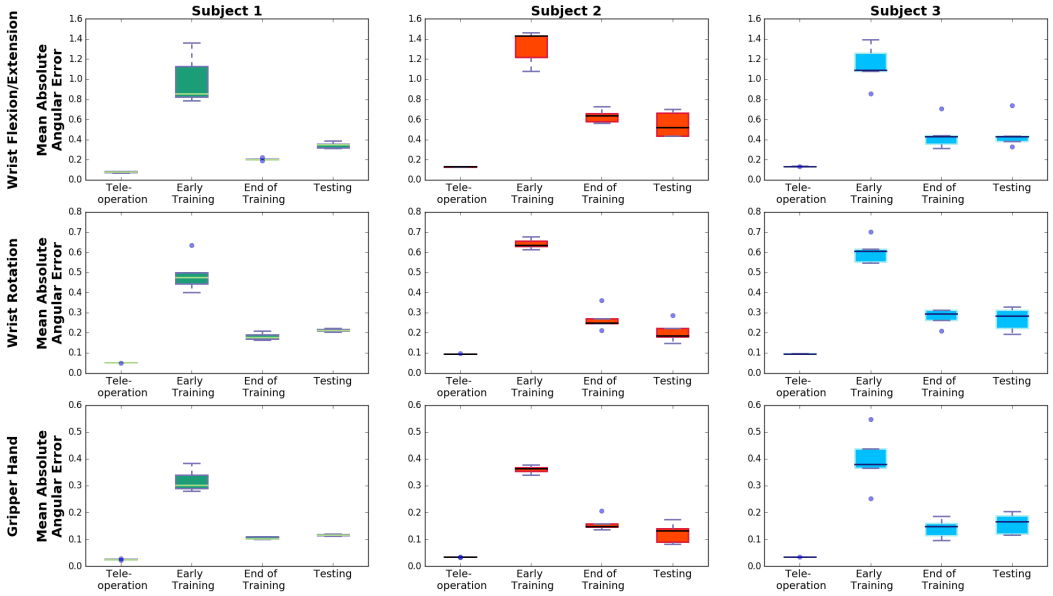


Figure 3.6: Comparison of mean absolute angular error accumulated over the course of $\sim$ 45min of learning. Quartile analysis of median values shown over 2 min of learning and testing as compared to direct reactive control for 5 independent runs for each subject. These plots are reflective of the performance of the ACRL learner on this particular task.

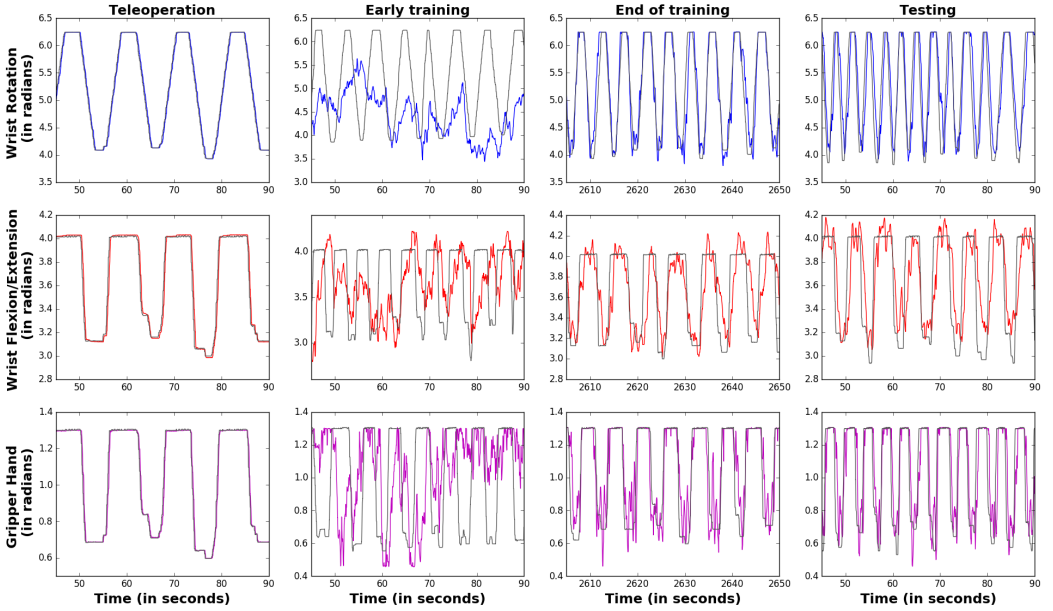
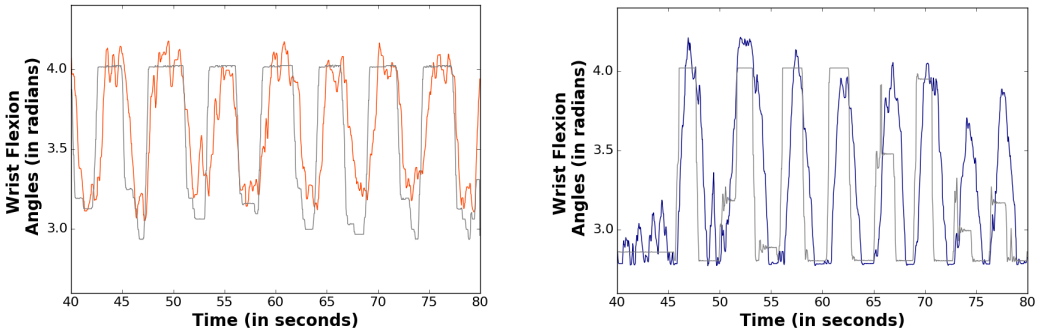


Figure 3.7: Comparison of target (grey line) and achieved (colored lines) actuator trajectories over training and testing periods. This plot shows the joint trajectories achieved by the ACRL learner for Subject 1 during training and testing as compared to the offline teleoperation baseline (reactive control).



(a) Testing on Day 1

(b) Testing after a week

Figure 3.8: Comparison of target (grey line, θ^*_{wf}) and learned (coloured lines, θ_{wf}) angles for control on different days.

3.6.2 Transferability of Results

In order to test the effectiveness of the learned control policy over prolonged use, the control policy learned from the initial training period was stored for future use. Another testing session was conducted after a week to evaluate the performance of the learned control policy on this new data. As can be seen in Fig. 6, the control policy managed to pick actions that achieve the desired trajectory, but it does visibly overshoot in a few regions. It can be seen from Fig. 6b that the actual trajectory overshoots considerably around 70 to 80 seconds. Though the system does capture the intent of the user, it doesn't encapsulate the finer movements of the user. We attribute this deterioration in performance in part to differences in EMG gains and sensor positions between visits. We believe that training the system with a larger dataset (training over multiple sessions, with and without muscle fatigue) could possibly alleviate some of these issues.

3.6.3 Why Reinforcement Learning?

Bernstein assumed that a large part of the development of motor control involved learning, and that such learning was accomplished through an active search involving gradient extrapolation by probabilistic sampling so that each attempt is informed by previously acquired information about how and where the next step must be taken (Bernstein (1967))(p. 161). RL has been identified as a promising

approach to learn from incremental experience and discover successful decision-making policies through the pursuit of reward.

By shaping these rewards, we can engineer behavior. Though we use a position-based similarity measure in our experiments, it is possible to use other non-trivial measures such as patient satisfaction (e.g., face-valuing methods (Veeriah et al. (2016)), torque or velocity matching, minimal power consumption, and others. We could imagine using a combination of these measures as an ACRL reward function to satisfy numerous goals. Mathewson et al. have further explored control learning methods using human-generated rewards and the robustness of these learning methods with stochastic reward signals (Mathewson and Pilarski (2016)). Optimizing multiple objectives is often challenging using other approaches like supervised learning.

3.6.4 Extending to Applications Beyond Upper-Limb Prostheses

While our approach was tested on the myoelectric control domain, it is widely applicable to other human-computer interaction tasks where the degrees of freedom problem exists. Researchers are currently looking at developing a robotic limbs attached to the human body (also known as supernumerary limbs) that can assist the human user during laborious tasks which cause discomfort and fatigue or when involved in tasks in dangerous environments. Bonilla and Asada (2014) developed supernumerary limbs that assist a human user while installing ceiling panels in an airplane. In our work we studied the scenario where an intact limb teaches a prosthetic limb different movement patterns. A user could also teach supernumerary limbs a desired behavior in the same way. Our approach is equally applicable to other domains like lower-limb gait training, lower-limb prostheses, powered orthotics, exoskeletons, and functional electrical stimulation systems.

3.6.5 Extending to Bimanual Tasks such as Playing the Guitar

When an amputee participant can demonstrate isolated control of all prosthetic functions and synergistic myoelectric control of multiple joints, we can include functionally relevant tasks such as loading a dishwasher, unloading a dryer, sorting

mail, or using form boards to test our one-handed prosthetic control. But a large number of everyday activities require the use of both the hands—tying shoelaces or a necktie, folding or hanging clothes, and removing money or credit cards from a purse or wallet. Higher levels of motor control are required for accurate rendition using a guitar. Subtle manipulation of timing and dynamics and coordination of finer finger movements is extremely important for playing the guitar. Using the learning-from-demonstration paradigm, the prosthetic arm can be given expert demonstrations of multiple renditions of a songs (for instance, fret and chord patterns for the guitar) and the agent can be tasked with learning a state-conditional policy that depends on the strumming and picking patterns exhibited by the intact arm.

3.7 Conclusion

This work presented an ACRL LfD framework that will potentially allow an amputee to use their non-amputated arm to teach their prosthetic arm how to move in a natural and coordinated fashion. To our knowledge, this study is the first demonstration of the training an upper-limb myoelectric prosthesis with a user's contralateral limb. We show that an ACRL learner can observe patterns of movement provided by a user and use these demonstrations in learning so as to generate accurate hand and wrist synergies during testing and free-form control by a user. Though our experiments were limited to motions involving three DoFs, our approach could be easily extended to incorporate more DoFs and finer motions. Ideally, we imagine someone with an amputation could use a LfD approach to continue to train a powered prostheses at home on an ongoing basis. While our approach is designed for upper-limb prosthetic control, we expect that it can be easily extended to other human-computer interaction tasks where the degrees of freedom problem exists. In the long run, we expect these methods to improve the quality of life for people with amputations by providing them better ways of communicating their intentions and goals to their myoelectric prosthesis. While we have shown results for coarse movements involving three DoFs, there is no algorithmic barrier to adding addi-

tional DoFs or DoCs such that a dexterous manipulator could be capable of finer movements and sophisticated multi-actuator grasp synergies. Using an intact limb to train a prosthetic hand for context-specific manipulation with multiple digits is the subject of ongoing work.

Chapter 4

Case Study: Learning from Demonstration with an Amputee Participant

4.1 Introduction

An amputee subject's capacity to generate control commands at will is often a major constraint in developing robust control algorithms for prostheses. There are significant differences in the EMG signals obtained from the residual limb of amputees and a healthy biological limb. Moreover, the learning-from-demonstration paradigm described in the earlier chapter relies only on the ability of the subject to control a single-joint using EMG signals (from biceps and triceps) and therefore transferable to an amputee. It has straightforward extensions to multi-joint pattern recognition and any user capable of using current clinical myoelectric control solutions could potentially benefit from the ACRL LfD approach presented here. Prior work by our group has also shown that temporal-difference-learning-based meth-

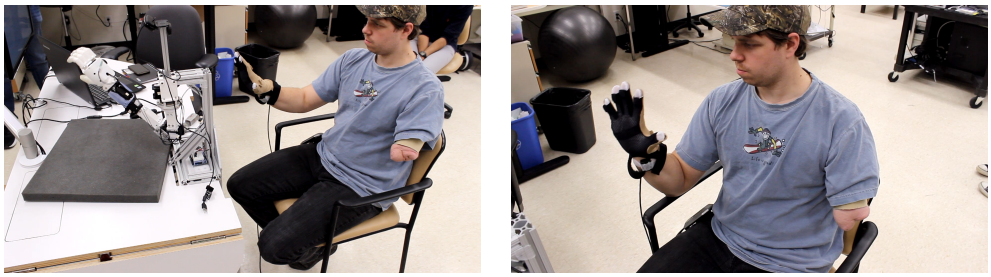


Figure 4.1: Learning from Demonstration - Study with an Amputee Participant [Learning from Demonstration - study with an amputee participant]

ods as used here operate in similar ways between subjects with and without amputations, especially when primary EMG control is performed via direct proportional mappings (Edwards (2016); Edwards et al. (2016)). In our earlier experiments (with able-bodied and amputee participants), the RL agent was able to learn an exact sequence of movements without knowing the underlying mechanics of the system or the relationship between EMG signals and desired motion. In part because of this generality, and the ability of RL approaches to optimize to case-by-base prosthetic situations (Pilarski et al. (2011); Mathewson and Pilarski (2016)), we expect our method to also transfer well to real-world use by amputees with transradial and shoulder disarticulation amputations, assuming reliable physiological control signals can be recorded from the user (i.e., users that could be prescribed myoelectric prostheses).

This chapter represents an addition to the work presented in the previous chapter. In the experiment described below, we assess the performance of the learning-from-demonstration paradigm with an amputee subject.

4.2 Methods

This experiment is similar to the experiment described in Chapter 3. Our subject had undergone a unilateral, transhumeral amputation (someone missing their hand, forearm, and elbow). In this setting, the subject has one biological hand (right hand), and one robotic arm (bento arm with a left hand) that they wish to train to appropriately respond to the commands being generated by the muscle tissue in the user's residual limb. We use the same experimental hardware setup described in Chapter 3.

As shown in Fig. 4.2, the control signals were sampled from the amputated arm that is attached to his robotic prosthesis. EMG signals from residual biceps and triceps were used in the direct control of their robotic elbow. The *training arm*, or the contralateral, intact biological arm was used to provide the training movements. We used the training movements to obtain the reward signals.

The user visualized performing the *same motion* with both arms (similar to the

Figure 4.2: We obtain control signals and state information from the control arm. The training arm provides reward signals for the RL agent.

pattern recognition training in (Scheme and Englehart (2011)). We used the motion of the training limb to provide training information for learning a prosthetic control policy that maps the state of the control limb (e.g., gross robot limb position and control-limb EMG signals) to motor commands for the remaining joints of the prosthetic hands and wrist not controllable by the user. During testing/deployment, the robotic prosthesis used its learned, state-conditional policy to "fill in the gaps" for the user.

In the data recording phase, our subject was instructed to execute a repetitive sequence of simple reaching and grasping movements that were mirrored by both their control and training arms. To make it easier for the subject to visualize the desired movement, we had our subject teleoperate the prosthetic arm. During this data recording phase, the elbow of the Bento Arm was actuated via proportional myoelectric control from the subject's control arm, while the wrist and hand of the Bento Arm were actuated via direct teleoperation—i.e., the Bento Arm copied the training arm's movements as reflected to the contralateral side. The subject used his training arm (i.e., intact arm) to demonstrate the desired behavior for three joints: wrist flexion/extension, wrist rotation and opening/closing the hand using the motion capture glove (the CyberTouch system) and IMU signals.

For our experiment, we chose a simple, repeatable movement as the desired behavior—a bicep-curl motion involving the smooth alternation of 1) supinated hand-closed wrist flexion during elbow flexion, and 2) hand-open pronation with wrist extension during elbow extension. For more details, please refer the supple-

mentary videos. [1]

The angular position of the hand and wrist flexor and wrist rotator joints (denoted by θ_h, θ_f and θ_r respectively) are correlated with the angular position of the elbow joint (denoted by θ_e) such that there exists a policy that could map any given elbow position uniquely into a combination of higher dimensional joint movements. We recorded data for $\sim 20mins$ for the user. The subject gave informed consent in accordance with the study's authorization by the University of Alberta Health Research Ethics Board.

4.2.1 Learning a Control Policy Using ACRL

The learning agent's action space consisted of 3 continuous angular velocity values—wrist flexor(a_{wf}), wrist rotator(a_{wp}) and the hand(a_h) actuators. Each of these three joints were assigned its own ACRL learner whose actions were drawn from a normal distribution $\mathcal{N}(s, a)$. The parameters of the normal distribution were functions of the system's learned weight vectors w_μ and w_σ as given by $\dot{\theta} \approx a \leftarrow \mathcal{N}\{\mu,\ \sigma\}$. Learning updates and action choices occurred every $\sim 40ms$ and $\sim 75ms$ of the training period respectively. If the actions selected by the learning agent were not allowed to persist for $\geq 75ms$, the actuators might not have enough time to execute the given velocity commands. Actions were computed on each timestep was described in Algorithm 2.

A continuous state space consisting of $s = \langle \theta_e, \dot{\theta}_e, \bar{s}, \theta_j \rangle$ was used, where $\theta_e, \dot{\theta}_e, \theta_j$ denote elbow joint angle, elbow joint velocity and current angle of the joint controlled by the robot. The difference between the mean-absolute-value of EMG was later computed for antagonistic muscle pairs (biceps/triceps), $\bar{s}_1 = \bar{s}_{BI} - \bar{s}_{TRI}$ to control the robot's elbow joint.

While there exists a direct correlation between the positions of the elbow joint and wrist rotator, the relationship between elbow joint angle and wrist flexion or hand angle isn't so straightforward. As seen in Fig. 4.3 (left), the gripper hands' trajectory can be thought of as a delayed response to the EMG control signals. But the

[1] `https://www.youtube.com/watch?v=UkZoY1JsnBA`
`https://www.youtube.com/watch?v=EtKQ7vSyYVk`

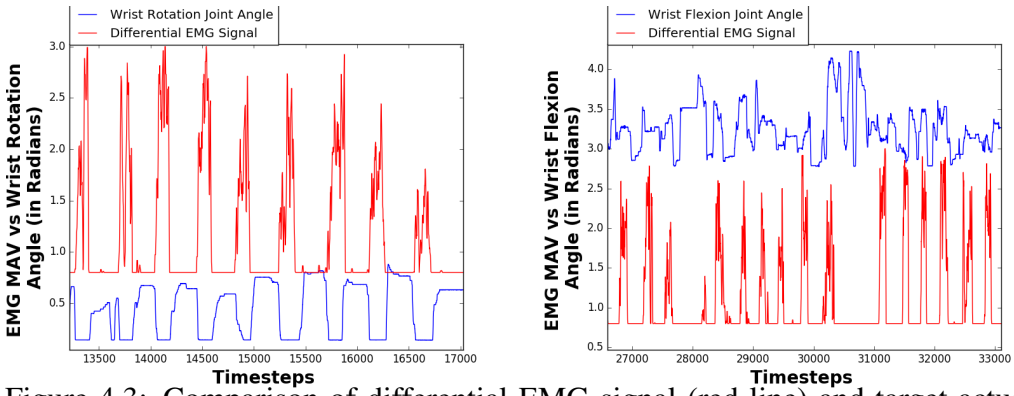


Figure 4.3: Comparison of differential EMG signal (red line) and target actuator trajectories (blue line) over training. There exists a clear correlation between wrist rotation joint angle and the processed EMG signals (left) whereas there is no distinct correlation between wrist flexion joint angle and the processed EMG signals (right).

wrist flexion trajectory is noisy and doesn't have consistently repeatable patterns as shown in 4.3 (right). In order to provide a richer state information to the RL agents, we included the last 5 observations as a part of the feature space. At each timestep t, we formed 64 tilings over $\langle {\theta_e}^t, \dot{\theta}_e^t, \bar{s}^t, {\theta_j}^t \rangle$, 64 tilings over $\langle {\theta_e}^{t-1}, \dot{\theta}_e^{t-1}, \bar{s}^{t-1}, {\theta_j}^{t-1} \rangle$ and so on up to $\langle {\theta_e}^{t-4}, \dot{\theta}_e^{t-4}, \bar{s}^{t-4}, {\theta_j}^{t-4} \rangle$, for a total of 320 tilings overall. Each tiling had a resolution of 10 x 4 x 8 x 12, for a total of $1,228,800$ features. Adding a single constant bias feature resulted in $1,228,801$ features, exactly $m = 321$ of which were active at any given timestep. The learning parameters were set as follows: $\sigma_c = 1$, $\alpha_v = 0.1/m$, $\alpha_\mu = 0.01/m$, $\alpha_\sigma = 0.5\alpha_\mu$, $\gamma = 0.99$ and $\lambda = 0.7$. Weight vectors e_v, v, e_μ, w_μ, e_σ, w_σ were initialized to zero and σ bounded by $\sigma \geq 0.01$.

The ACRL systems was trained incrementally using repeated cycles of the demonstration data recorded earlier, as described in Sec. 3. We had recorded data for $\sim 20mins$ of which $15mins$ was used for the training set and $5mins$ for testing set. Total training time was held constant at 45 min after which the learned control policy was tested on a different data set (i.e., hold out set) for accuracy. The control learner received negative rewards r on each step proportional to the difference between the target and current joint angles: $r_j = -|{\theta_j}^* - \theta_j|$, in radians. Each controlled joint had its own ACRL learner with its own reward function r_j.

Performance of the learning system was measured based on its ability to achieve desired joint angles. All learning algorithms were run on a Lenovo Flex-3 Laptop with Intel Core i7-6500U @2.50GHz x 4 and 8GB RAM. We used the Robot Oper-

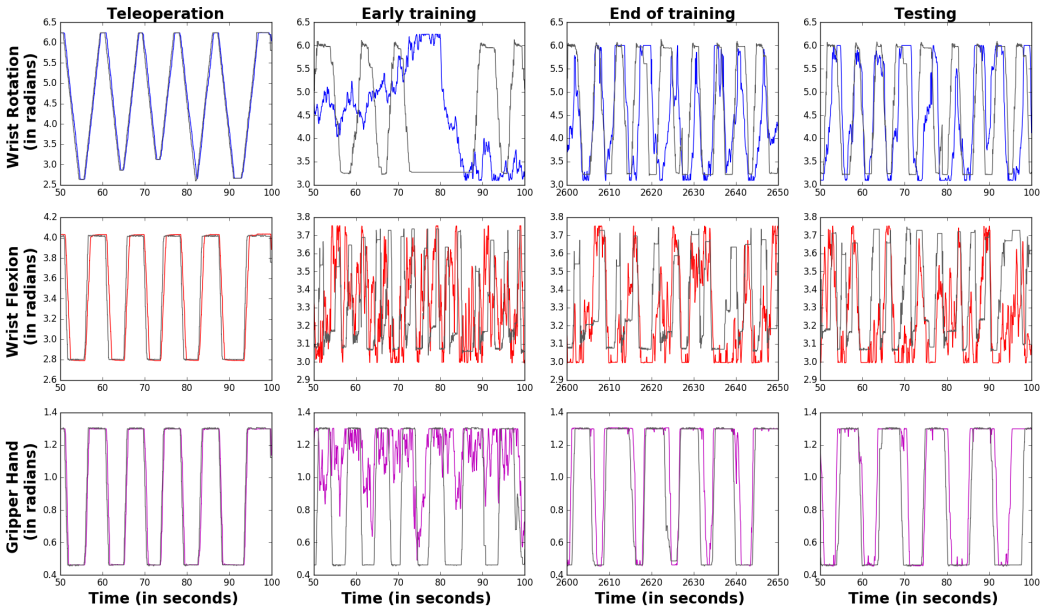


Figure 4.4: Comparison of target (grey line) and achieved (colored lines) actuator trajectories over training and testing periods. This plot shows the joint trajectories achieved by the ACRL learner for the amputee subject during training and testing as compared to the offline teleoperation baseline (reactive control).

ating System (ROS) Kinetic on Ubuntu 16.04 to send and receive information and commands from the Bento Arm, CyberTouch II and the Delsys Trigno Wireless Lab.

4.3 Results

We use reactive control (an offline equivalent of direct teleoperation) as our baseline performance measure for the learned ACRL policies. In this approach, the controller observes the desired joint angle ${\theta_j}^*$ and takes an action a_j that moves current actuator angle θ_j towards the target angle as quickly as possible.The reactive control approach assumes that an oracle provides the desired joint angles (i.e., states and targets are fully observable) and hence is applicable only as a training-data baseline.

Figure 4.4 shows examples of the joint control trajectories achieved by the ACRL learner during the early and late stages of learning, and during testing for 2 joints — wrist rotation and hand open/close. The agent managed to learn a control policy that ensures that the joint angles remain within the target regions for

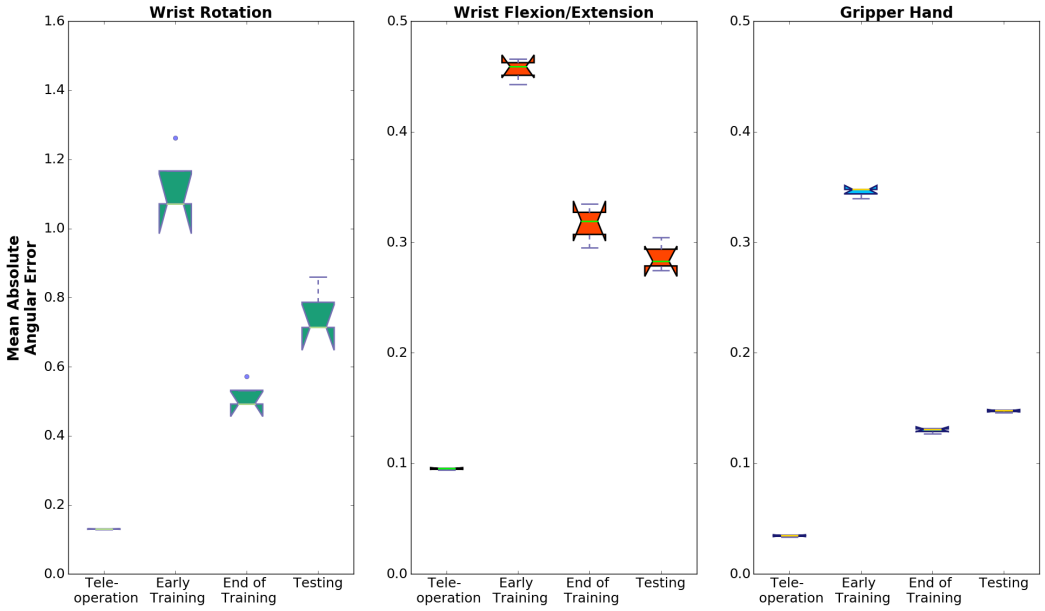


Figure 4.5: Comparison of mean absolute angular error accumulated over the course of $\sim$ 45min of learning. Quartile analysis of median values shown over 2 min of learning and testing as compared to direct reactive control for 3 independent runs for each subject. These plots are reflective of the performance of the ACRL learner on this particular task.

the majority of the evaluation period during both testing and training scenarios. The trajectories achieved by the reactive control approach and ACRl learner are compared to each other. The ACRL learner started to achieve the desired joint movement, though the controller did visibly overshoot/oscillate around the desired trajectory after 20 min of learning. The controller started to tightly track the desired trajectory following 30 min of learning. However, as seen in Fig. 4.4, we observe occasional spikes or delays in the joint angles whenever there is a sudden transition in the desired position. These spikes likely correspond to the fact that less time is spent in training for transitional motion. The ACRL learner was unable to learn a control policy for the wrist flexor joint. One major reason could be that there is no rhythmic movement involved (as seen in figure 4.3 (right)).

4.4 Conclusion

To the best of our knowledge, this trial is the first demonstration of training a prosthetic arm with an amputee user's non-amputated arm. The ACRL learner was

able to observe patterns of movement provided by an amputee user and learn from these demonstrations in order to generate reliable hand and wrist synergies during the testing period. While we considered the case of a unilateral amputation, we could also imagine an occupational therapist using our approach to demonstrate single-arm or bimanual training movements during occupational therapy for bilateral amputation and myoelectric control.

Chapter 5

Context-aware control of multi-DOF prosthesis

5.1 Introduction

Humans have an astounding ability to learn a variety of motor skills, ranging from tying shoelaces, threading a needle to shooting a basketball. A wide number of interacting elements are involved in learning such skills. In order to be proficient in multiple motor skills, we should learn to gather task-relevant sensory information efficiently and use them in decision making and selection of control strategies.

When any muscle or muscle group is activated, the resulting movement is dependent on the context. The relationship between muscle excitation and movement is variable and this variability is context conditioned (Turvey et al. (1982)). Exteroception [1] and proprioception [2] information can be used by the motor system as feedback to guide postural adjustments and control of semi-automatic or skilled movements such as those involved in writing, jogging, etc. Consider picking up a coffee mug — to accomplish this task, the hand is transported to an appropriate location in the vicinity of the mug and then oriented and pre-shaped conveniently to grasp the mug by coordinating the muscles of the hand and applying the right

[1] The perception of environmental stimuli acting on the body. The stimuli are perceived by special, sometimes highly complex, structures called exteroceptors. An example of exteroception is the perception of light, sound, or heat.

[2] The sense of position and movement of the limbs and the sense of muscular tension. The awareness of the orientation of the body in space and the direction, extent, and rate of movement of the limbs depend in part upon information derived from sensory receptors in the joints, tendons, and muscles.

amount of force.

Vision provides critical sensory information for the planning and execution of hand movement and prehension, since it allows the central nervous system to predict the extrinsic properties of the target object. The perception of the object's location, size and shape, and orientation with respect to the environment enables the brain to execute fast-reaching movements by combining muscle synergies. Vision also provides closed-loop feedback during the execution of body movements to allow for corrections, especially during the reaching phase.

Apart from EMG signals used for the myoelectric control, we can use many other physiological and non-physiological signals to assist in the control of multiple DOFs. We hypothesize that the prosthesis controller can emulate highly synergistic movements traditionally considered the responsibility of the user if equipped with such sensors. By combining sensory information from different sources, the controller should be able to detect and analyze the current context, plan the movement strategy, and simultaneously and proportionally control multiple DOFs available in the prosthesis.

In this chapter, we explore an approach that could produce context-dependent motion if presented with contextually relevant representations.

5.2 Related Work

Markovic et al. (2015) introduced a sensor fusion and computer vision based control approach for the context-aware control of a multi-DOF prosthesis. This work uses a combination of sensing units, comprising myoelectric recording, computer vision, inertial measurements and embedded prosthesis sensors (position and force), to develop a controller that could allow a multi-DOF prosthesis to perform simultaneous, coordinated movements. The method relies on sensor fusion which allows for the perception of the user (proprioception), the environment (exteroception) and their interaction, leading to simultaneous, proportional control of multiple DOFs through context-dependent behavior.

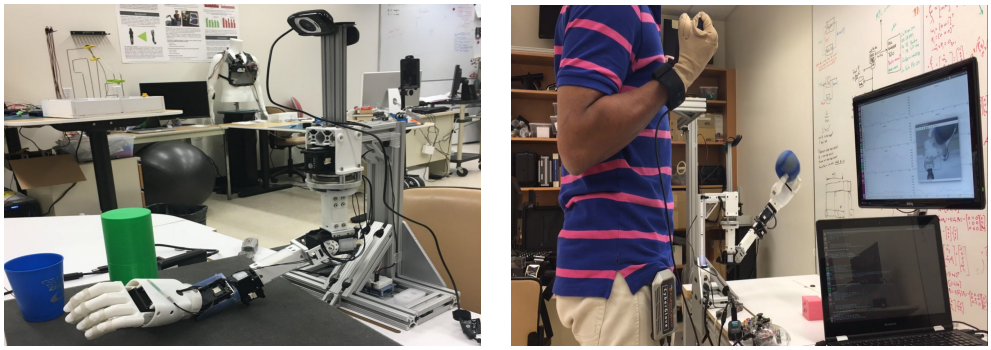


Figure 5.1: Experimental setup which includes a high-definition camera, Bento Arm, Thalmic Myo Armband and CyberTouch II. The Bento Arm as used in our trials had 5 active DoFs including shoulder rotation, elbow flexion/extension, wrist pronation/supination, wrist flexion/extension and hand open/close.

5.3 Methods

In order to achieve situation-dependent movement based on muscle excitation, the control system should be given the relevant contextual information and meta-data about the user, the robotic limb and its environment. In the present work, we explore how additional sensory information can be exploited to improve synergistic control of a multi-DOF prosthetic arm. More specifically, we equip the prosthetic arm with artificial vision to perceive the state of the user, prosthesis and the environment. Using this additional information, the controller should learn to distinguish between different grasp patterns according to the context and user's intentions (communicated using EMG signals).

Similar to the experiments described in Chapters 3 & 4, we assume that our subject had undergone a unilateral, transhumeral amputation (someone missing their hand, forearm, and elbow). In this setting, the subject has one biological hand (right hand), and one robotic arm (bento arm with a left hand) that they wish to train to appropriately respond to the commands being generated by the muscle tissue in the user's residual limb. There are two significant changes in our hardware setup—first, we mount a Logitech HD 1080p webcam on top of the arm (see Fig. 5.1 (left)) and second, we use the Thalmic myo armband instead of the Delsys Trigno Wireless system to obtain EMG signals.

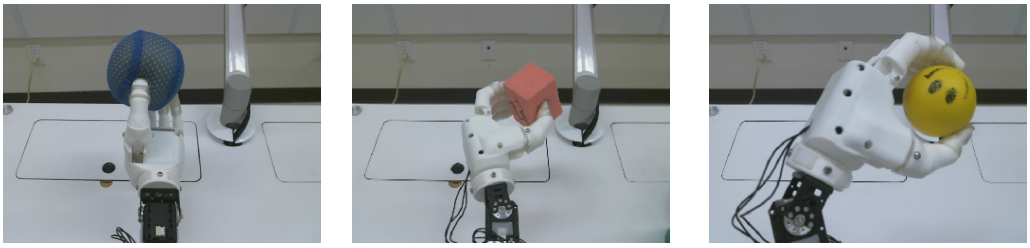


Figure 5.2: We used three different objects for our experiment - a large blue ball, a red sponge and a small yellow smiley ball. The input images were processed and provided to the learning system as input features.

5.3.1 Hardware

Bento Arm: The Bento Arm is a myoelectric training tool to assess and train upper-limb amputees in how to use their muscle signals prior to being fit with a myoelectric prostheses (Dawson et al. (2014)). We use a 5-DOF Bento Arm for the purposes of our experiments. It is a 3D printed prototype with Dynamixel MX Series servos.

EMG Data Acquisition: We used a 8-Channel Thalmic Myo armband to record EMG signals from our subjects. The Myo armband is fashioned as a wearable gesture control and motion control device that can be worn around the forearm/upper arm. It also provides inertial measurements which can be used to calculate rotation and translation with respect to a fixed frame of reference.

Motion Capture Glove: The desired joint angle configurations for wrist flexion/extension (θ^*_{wf}) and hand open/close (θ^*_h) were defined by the subject using a CyberTouch II system (CyberGlove Systems LLC) worn on the hand of their training arm.

Computer Vision: We used a Logitech HD 1080p webcam mounted on top of the Bento Arm. The camera captured images at 50 frames-per-seconds. Few examples of the images obtained from the camera are shown in Fig. 5.2.

We're following the same learning-from-demonstration protocol as outlined in Chapter 3. In the earlier chapters, the ACRL learner observed a single movement pattern provided by the user and used the demonstrations in order to learn and generate accurate hand and wrist synergies during testing and free-form control by an able-bodied user. Here, we are trying to learn 3 different muscle synergies

for grasping the following objects – a large blue ball, a red sponge and a smaller yellow smiley ball as shown in Fig. 5.2 (right).

This experiment was similar to the experiment described in Chapter 3, in which we had our participants demonstrate the desired movement and grasp patterns for 3 DOF to the ACRL learner. In this experiment, our subject demonstrated the wrist and grasp patterns for manipulating the desired objects.

The task can be divided into multiple segments as follows - grasp the object on the table, bring it up by flexing the elbow (as if the object is being examined closely), extend the arm and bring it down and finally drop it on the table. While the EMG control commands remain the same for flexing and extending the elbow, the corresponding wrist and hand trajectories manipulating each object is different. The hand, wrist flexor and wrist rotator joints (denoted by θ_h, θ_f and θ_r respectively) are correlated with the angular position of the elbow joint (denoted by θ_e) such that there exists a policy that could map any given elbow position uniquely into a combination of higher dimensional joint movements. The subject demonstrated each desired reaching and grasping movement for $\sim 15mins$.

5.3.2 Prediction in Adaptive Control

Highly skilled motor behavior relies on our brain learning both to control its body as well as predict the consequences of this control. Flanagan et al. (2003) studied the relation between predictions as control during motor learning. They found different time scales of learning for predictions and control, with predictions being learned much faster than control. Pilarski et al. (2013b) integrated learned anticipatory predictions into the control the actuators of a multi-joint prosthesis for use by amputees, especially amputees with limited signal recording sites on their amputated limbs. They were able to make accurate, anticipatory predictions at different timescales about various joint angles, EMG signals, etc. Their integration of real-time prediction and control learning promises to speed up control policy acquisition, allow unsupervised adaptation in myoelectric controllers, and facilitate coordinated, synergistic movements in a mutli-DOF prosthetic limbs. In this chapter, we test the ability of our system to learn accurate, temporally abstracted predictions about the

actuator positions of the joints controlled by the learning agent. In the future, we hope to implement a similar integration of real-time prediction learning and control to learn a larger repertoire of motor behaviors.

5.3.3 Making Predictions Using TD(λ)

In this experimental setting, the EMG control signals from the user are extremely similar across the three context-dependent control tasks. Artificial vision is the only distinguishing input feature that could help differentiate between the tasks. Learning accurate predictions about the desired target trajectories for each object shows the ability to ascertain context. We believe that integrating these learned predictions into a prosthetic control system can achieve context-aware control of a multi-DOF prosthesis.

We use the standard temporal difference learning (Sutton and Barto (1998)) to allow the system to make predictions about the desired joint angles. We created three General Value Functions (GVFs) (Sutton et al. (2011)) for predicting the three signals of interest ${\theta^*}_j$ in the robotic system. We make temporally extended predictions at a short time scale (1.0s) about three target joint angles—wrist rotation, wrist flexion and hand open/close. The learning agent was presented with a signal space consisting of the following:

- Elbow joint angle and velocity $\langle \theta_e, \dot{\theta}_e \rangle$
- Object specific features — We converted the input *rgb* images into *hsv* format and analyzed how often certain colors appear and in what proportions they are to be found in different types of images. We found the range of 'r', 'g' and 'b' values of the corresponding objects manually and used it to classify the target object. Based on the number of blue pixels within a particular threshold, we can classified the three target objects and assigned a distinct numerical value. For example, blue ball in the frame resulted in a signal value of one, the red sponge had a signal value of two and smiley ball had a signal value of three.
- EMG control signal — We used the difference between the mean-absolute-value of the EMG signals obtained from sensors 3 and 8 (placed directly over

the biceps and triceps respectively).

We used tile coding (Sutton and Barto (1998)) for linear function approximation. Our state representation consisted of 32 incrementally offset tilings (width=1) for better generalization. Each tiling had a resolution level $N_R = [10]$. The binary feature vector of length 5,000,000 was hashed down to a memory size of 8192 and we also added a bias unit which was always active. At every timestep, 4 continuous signals were provided to the tile coder and $m = 33$ features were active. The learning parameters were set as follows: $\alpha = 0.1/m, \gamma = 0.99, \lambda = 0.7$. Weight vectors w, e were initialized to zero. Each GVF received its target joint angle ${\theta^*}_j$ as the cumulant (i.e., reward signal).

Performance of the learning system was measured based on its ability to predict desired joint angles. All learning algorithms were run on a Lenovo Y700 Laptop with Intel Core i7-6700HQ @2.60GHz x 8 and 8GB RAM. We used the Robot Operating System (ROS) Kinetic on Ubuntu 16.04 to send and receive information and commands from the Bento Arm, CyberTouch II and the Thalmic Myo armband. All sensorimotor information were communicated between different systems using ROS topics. We recorded all the sensorimotor informations (from ROS topics) using rosbags. Rosbags avoid deserialization and reserialization of the messages. After recording, we can playback the data in a time synchronized fashion and simulate real-time sampling and learning conditions. While we used an offline approach for obtaining the results, it can easily be extended to an online setting.

5.4 Results

The prediction P_q is dependent on the timescale (i.e., time constant) of return predictions determined by γ (i.e., the discount factor). For normalized return predictions ($\bar{P}_q$), predictions are scaled according to the time constant. For example, $P_q = \bar{P}_q/(1 - \gamma_q)$.

As shown in the results, the system was able to successfully anticipate the joint trajectories initiated by the subject. Accurate predictions were observed after $5 - 6mins$ of real-time sampling (simulated by playing back recorded data and

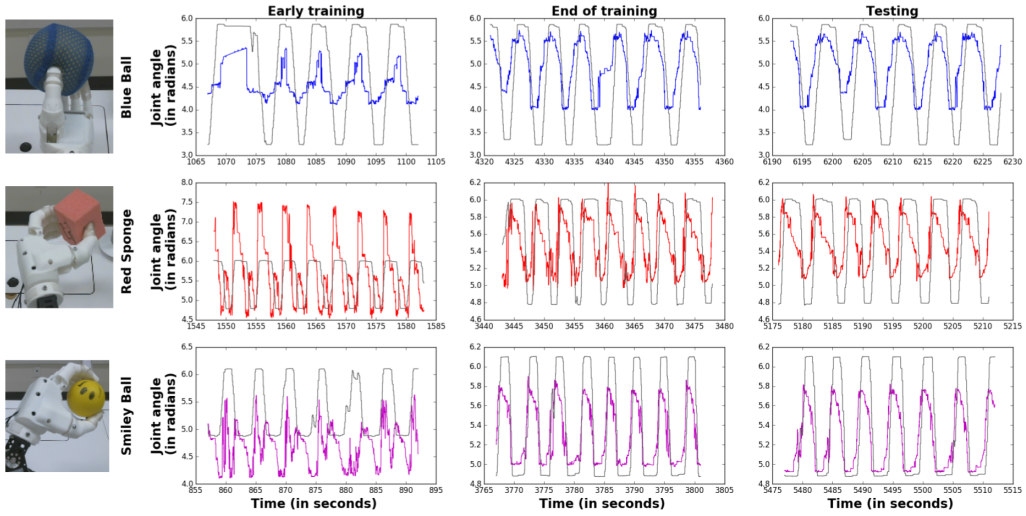


Figure 5.3: Comparison of target (grey line) and predictions (colored lines) of wrist rotation trajectories over training and testing periods. This plot shows the joint angle predicted by the TD learner for the able-bodied subject during training and testing.

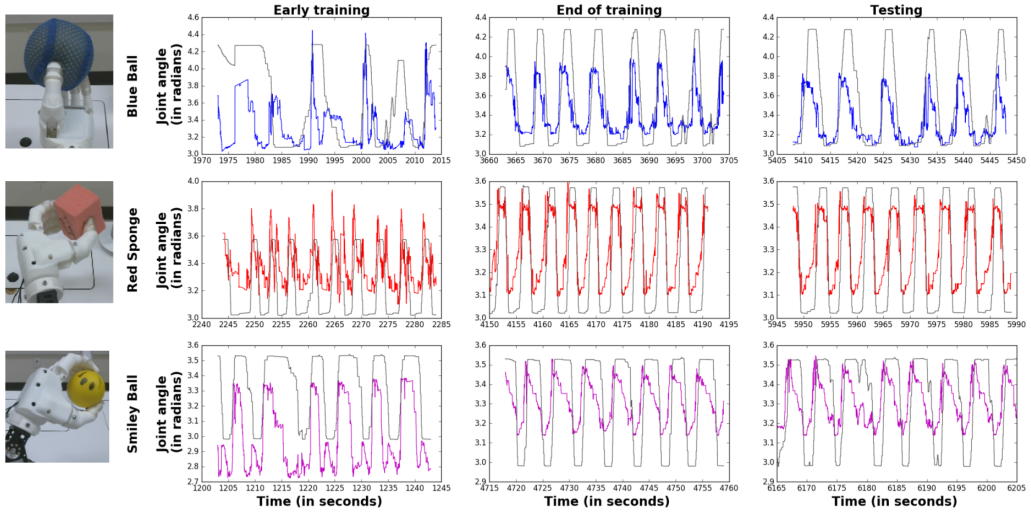


Figure 5.4: Comparison of target (grey line) and predictions (colored lines) of wrist flexion trajectories over training and testing periods. This plot shows the joint angle predicted by the TD learner for the able-bodied subject during training and testing.

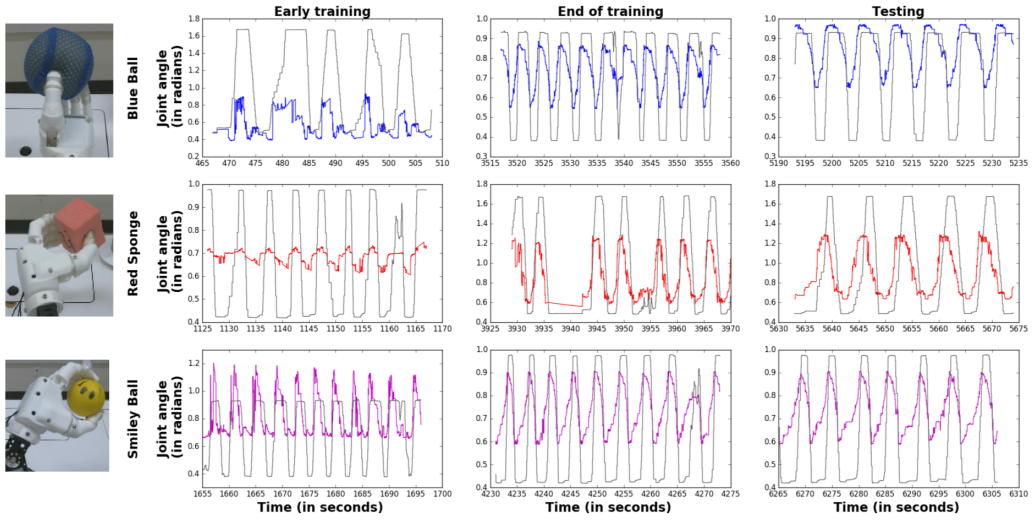


Figure 5.5: Comparison of target (grey line) and predictions (colored lines) of gripper hand trajectories over training and testing periods. This plot shows the joint angle predicted by the TD learner for the able-bodied subject during training and testing.

synchronized using timestamps) and learning. Here the normalized predictions $\bar{P}_q$ at a time scale of 1.0s (colored lines) are compared against corresponding target joint angles (grey lines). Fig. 5.3 shows an example of wrist rotation angle prediction after two offline learning passes through $10mins$ of recorded training data from the subject demonstrations. The agent was tasked with learning three target joint angles in parallel—wrist flexion, wrist rotation and hand open/close. The training phase lasted for $\sim 90mins$ (two offline passes through $15mins$ of recorded data for each demonstration) and the testing phase lasted for $\sim 15mins$ (one offline pass through $5mins$ of recorded data for each demonstration). Fig. 5.4 and 5.5 show examples for normalized joint angle predictions for wrist flexion and gripper hand respectively over training and testing periods.

5.5 Discussion

5.5.1 End-to-end RL for Context-Aware Control in a Real World Setting

In this chapter, we showed that it is feasible to learn to distinguish between the desired grip trajectories for 3 different objects. But we used heavily engineered

features in a simple setting. The predictions for the same task would start failing even under slightly altered conditions — for example, altered lighting, displacement of the object, etc. While the same features used for the prediction experiments can be used to learn a good control policy using ACRL, it'd still face the same limitations (i.e., it'd fail under slightly altered conditions). We need a robust control policy that can generalize to new situations and resilient to lighting conditions. We believe that RL can be used to learn predictive models which in turn can be used for control.

5.5.2 Sensor Fusion for Context-Aware Control of a multi-DOF Prosthesis

As discussed in Chapter 3, the learning-from-demonstration paradigm is not applicable only to prosthetic arms, but could also be extended to wearable robots such as exo-skeletons, powered orthotics, supernumerary limbs, functional electrical stimulation systems, etc. While these devices face a lot of challenges in both hardware and software design, a major challenge is that the robot usually lacks the capability to adequately recognize the actions and intentions of the human user. Consequently, it cannot assist the user appropriately, a drawback that has been emphasized in the rehabilitation robotics domain.

In most wearable robots, many sensors are already built into the device, such as joint angle sensors, electro-physiological measurements such as electromyography (EMG) or electroencephalography (EEG), or alternatively mechanical sensors or inertial measurement units (IMUs) placed on a part of the body that is not covered by the wearable robot. We can combine this multi-modal information (combining different sensor types) to better learn and adapt to the needs of the user.

In the rehabilitation robotics domain, the DOC can greatly outnumber the number of input channels the user has available. For example, in the case of an amputee user, the disparity between the Degrees of Control and the number of available input signals greatly increases as the level of amputation increases (e.g., those with transhumeral amputations can provide even fewer control signals than those with transradial amputations). Unfortunately, this gross mismatch makes the control of

wearable robots difficult and tedious. Sensor fusion can potentially alleviate some of the issues associated with controlling large DOFs with a small subset of input signals.

In addition to the standard electro-physiological signals, IMUs and joint angle measurement units, we could add artificial vision, gaze vectors (to know where the user is looking) and tactile sensation systems (for example, a camera and capacitive touch systems respectively) to the robotic devices. These systems could provide highly useful sensory information to the learning agent that could be used to better perceive the environment and needs of the user.

5.5.3 Representation Learning

Modern day prostheses could receive a huge density of data about the user, their physiological and psychological needs and their environment. For example, camera data or even additional sensors on the socket of a prosthesis can readily provide enough contextual information to allow an ACRL system to produce varied motor synergies in response to similar EMG signals from the user—e.g., a system can use additional sensor and state information to help manage the user's degree-of-freedom problem, generating synergies that artfully align to different situations in the user's daily life. It is therefore important that efficient ways of structuring prosthetic data are developed to better represent context to a machine learning prosthetic control system without facing the curse of dimensionality. For example, representation learning methods built on Kanerva Coding could potentially be used to better handle this large number of real-world state signals (Travnik and Pilarski (2017)).

While the idea of using a single state representation to better leverage the multi-modal sensory information is extremely appealing, it's been shown that different function approximators can be better at learning about different types of data. For example, convolutional neural nets (CNNs) have been widely successful in image classification and object detection datasets (Krizhevsky et al. (2012)). Similarly, motor primitives (discussed in the upcoming section) have been successful in encoding rhythmic and discrete movements. Recurrent neural networks (RNNs) have been extremely successful in speech recognition and text translation (Graves et al.

(2013)). One simple way to combine all the sensory information is to extract the features for each modality separately, then input all features into a single sensor fusion algorithm.

5.5.4 Study Limitations and Future Developments

The experimental evaluation in this study was designed to test the learning system's ability to make accurate predictions about context-dependent joint trajectories. Therefore, the focus of the study was set primarily on performing a set of functionally relevant ad-hoc experiments (e.g., grasping and lifting distinctly colored, differently oriented objects) designed for everyday multi-DOF prosthesis usage, taking into account the constraints of the setup (e.g., desk mounted Bento Arm, vision sensor mounted on top of the arm, etc.). The ad-hoc experiments were hence only used to evaluate our hypothesis on context-aware control of a multi-DOF prosthesis. We are fully-aware of the importance of standardized evaluations of the functional effectiveness of a powered prosthesis. In the future, we should develop methods that are generalizable across different tasks and objects. Once this improved approach is in a more mature, developmental phase and approaches clinical applicability (i.e., amputees can use it in a practical setup), the current ad-hoc tasks would be replaced with functionally relevant tasks.

5.6 Conclusion

Our approach was able to learn contextually-accurate predictions from joint trajectories demonstrated by an able-bodied user. These results provide a starting point for research into long-term control adaptation. An interesting extension to this work is to explore the use of contextual-prediction architectures along with ACRL or alternative control approaches in complex real-world activities and evaluate predictions for context-aware control adaptation with a population of amputees.

Chapter 6

Discussion & Future Work

In this chapter, we propose possible future extensions for a collaborative control approach for myoelectric prosthesis. In our approach, the user controls a few selective joints and the learning agent controls the rest of the joints. The user and the agent control the arm together to achieve the user's goals. In a sense, the agent is trying to compensate for some of the limitations of the user.

In the first section, we talk about a scalable, efficient way of reducing task complexity for visuo-motor learning. The second section looks at alternative policy gradient and control learning approaches that could be used instead of actor-critic reinforcement learning. While ACRL has several appealing properties, few alternative approached may be better suited to handle this particular application. In the third extension, we try to draw inspiration from the neurobiology of an octopus. The nervous system and neuromuscular system of an octopus has a unique organization. The huge amount of sensory information (suckers, skin and intrinsic arm musculature) and infinite DOFs pose a great challenge to sensory representation and motor control. We also argue that the prosthetic arm should have a "brain" of its own and that such an intelligent arm could potentially give rise to synergistic movements like catching a ball or playing the piano.

6.1 Information Extraction for Visuomotor Learning

The exploits of Cristiano Ronaldo and Lionel Messi represent the pinnacle of motor learning. Even though an average person can't play soccer in such an impressive

fashion, the mere range of motor abilities exhibited by mere mortals is quite remarkable. We exercise our motor learning capabilities to pick up new activites — whether it's mountain climbing or playing the guitar — but also engage in substantial motor learning even on a daily basis to better adapt to changes in our body and the environment.

Skilled performance in any motor skill requires efficient gathering and processing of relevant sensory information and using it to build a motor control policy. We do this actively since what we see, hear, smell, touch and taste is influenced by our movements. During visual search for a target among distractors, people choose to saccade [1] to the location that would best minimize the uncertainty over all possible target locations (Wolpert et al. (2011)).

Land and McLeod (2000) examined the eye movements of cricket batsmen and show that, in general, they do not watch the ball continuously. Rather, they have a distinct focal attention/eye movement strategy to view the ball at crucial moments during its flight. A typical fast bowler in cricket delivers the ball at a speed of $150 - 160km/h$. A batsman watches the fast bowler's ball propel toward him at this high and unpredictable speed, bouncing off the ground(called a pitch in cricket) of uncertain hardness. Although he can only view the trajectory of the ball for $\sim 0.5seconds$, he can accurately predict when and where the ball will reach him. Once the ball is released from the bowler's hand, the batsman's eyes make a predictive saccade to the place where they expect the ball to hit the ground, wait for it to bounce, and follow its trajectory for 100–$200ms$ after the bounce. Land and McLeod (2000) showed that the information provided by these fixations could allow precise prediction of the ball's timing and placement. A shorter latency for the first saccade distinguishes experts from amateurs.

This suggests that the motor system is used to sample the sensory world to selectively extract task-relevant information since the attentional and processing resources are limited. Wolpert et al. (2011) also suggest that motor learning can also push the limits of what our perceptual system can do. For instance, expert gamers (video game players) can develop an amazing ability to extract information

[1] A rapid movement of the eyes that changes fixation from one point to another

and spread their attention over a wide spatial frame without any apparent decrease in attentional performance (Green and Bavelier (2003)).

We explored the context-aware control of a prosthesis equipped with artificial vision and proprioception to perceive the state of the user and the environment. Even though this gives us a lot of sensory information to analyze the scene, estimate the shape, size and orientation of the target object, modern day prosthesis do not have enough computational resources to leverage this additional data. Given the size constraints of the hardware, a prosthetic arm could fit a small computational device like a raspberry pi or a beaglebone black. Hence, we need smarter state representations and scalable, strictly incremental algorithms that could handle the complexity. One possible future extension is to use ideas drawn from feature integration theory (Treisman and Gelade (1980)).

6.1.1 Feature Integration Theory of Visual Attention

Feature Integration theory is a two-stage theory of visual attention (Treisman and Gelade (1980)). In the first stage, basic features are processed automatically, independently, and in parallel. In the second stage, other properties, including relations between features of an object, are processed in series, one object (or group) at a time, and "bound" together to create a single object that we perceive.

Based on findings from parallel search and conjunction search tasks, Treisman and Gelade (1980) suggested that in the early stage of object perception, different features of an object (e.g. colour, orientation, direction of motion) are thought to be analyzed separately (and in parallel) by several pre-attentive cognitive mechanisms, and the role of attention is to 'glue together'(or integrate) the different features to form a coherent representation.

Treisman and Schmidt (1982) predicted that when attention is diverted or overloaded, features may be wrongly recombined and this give rise to *illusory conjunctions*[2]. When we are not paying attention, nor expecting any particular object, the

[2]Illusory conjunctions are perceptual phenomena which may occur when several different stimuli are presented simultaneously to an observer whose attention has been diverted. For example, during a conjunction search task, the user may perceive a red cross and a green circle when a red circle and a green cross are presented. This type of error is called a "conjunction error".

world doesn't turn invisible. Instead some the features detected in parallel in the first stage could be randomly conjoined, although others may remain unconnected and therefore doomed to remain unconscious. For example, observers are tasked with identifying two digits when presented on the display. When distracted, they often report seeing dollar signs, even though the S and the straight line which make up the sign are never in the same location. It is as though, pre-attentively, the S and the straight lines are 'free-floating' and the observers are able to combine them to form a $ symbol even though it is not physically present in the display. These illusory conjunctions support feature integration theory (Treisman and Schmidt (1982); Treisman (1986)).

Rensink (2000) suggested that humans focus attention (or fixate) selectively on parts of the visual space to acquire information when and where it is needed, and combine the processed information from different fixations over time to build up an internal representation of the scene. This fits very well with feature integration theory.

Mnih et al. (2014) formulated the attention-based processing of visual sensory information as a control problem. They presented a recurrent neural network model that is capable of extracting information from an image or video by adaptively selecting a sequence of regions or locations and only processing the selected regions at high resolution. Inspired by this work, Xu et al. (2015) created an attention-based model that learns to automatically describe the contents of an image (or generate captions for an image).

Similarly, an attention-based model can be used in the context of wearable robots in order to focus on a particular object in a cluttered environment. Imagine a situation where an amputee donning a prosthetic arm (with artificial vision) is in a kitchen. Attention-based models for control could potentially identify desired tools in a cluttered environment and help with cooking.

6.2 Alternative Approaches to Learn a Robust Control Policy

In this section, we discuss several learning from demonstration approaches for robotics applications. All of these approaches treat the robot as an autonomous system which learns to do a particular task from expert demonstrations. However none of these tasks are dependent on control signals provided by the human user. Nevertheless, there are several appealing properties to these approaches and they can extended to the prosthesis domain. This section is intended to serve as a concise summary of autonomous robot control approaches that could potentially be used in the assistive robotics domain.

6.2.1 Motor Primitives

Motor primitives can be thought of as neural control modules that can be flexibly combined to generate a large repertoire of behaviours. For example, a primitive might represent the temporal profile of a particular muscle activity. The overall motor output is just the sum of all primitives, weighted by the level of the activation of each module (d'Avella et al. (2006)).

Ijspeert et al. (2003) conceive of motor primitives as simple dynamical systems that can generate either discrete or rhythmic movements about every DOF. To get a movement started, we just need to initialize the speed and amplitude parameters. Learning is required to fine-tune certain additional parameters in order to improve the movement. This approach allows us to learn movements by just adjusting a relatively small set of parameters. The resulting movement generation has a variety of advantages—re-scalability with respect to both time and amplitude (i.e., amplitude and durations/periods of learned patterns can be modified independently without affecting the qualitative shape of the trajectory), basic stability properties and the possibility to encode either single-stroke or rhythmic behaviors. If a good function approximator is chosen in this context (ideally one that is linear in its parameters), then learning can be online for real-time robotics applications.

But estimating the parameters of the dynamical system is slightly more daunt-

ing (i.e., the duration of discrete movements is extracted using motion detection and the time-constant is set accordingly). Similarly in Ijspeert et al. (2003), the base period for the rhythmic dynamical motor primitives were extracted using the first set of repetitions and, again, the time-constants τ are set accordingly. As the start-up phase in rhythmic presentations may deviate significantly from the periodic movement, the baseline of the oscillation often needs to be estimated based on the later part of the recorded movement, the amplitude is determined as the mean of the amplitudes of individual oscillations in this portion of recorded movements (Schaal et al. (2007)).

Motor primitives have been used for a variety of basic motor skills such as tennis swings (Ijspeert et al. (2003)), T-ball batting (Peters and Schaal (2006)), drumming (Pongas et al. (2005)) and planar biped walking (Kober and Peters (2009)). All these approaches focus on learning by imitation without subsequent self-improvement. They were formulated as a supervised learning approach with the target trajectory obtained from the recorded motions of a human player by kinesthetic teach-in. Kinesthetic teach-in means "taking the robot by the hand", performing the task by moving the robot while it is in gravity-compensation mode and recording the joint angles, velocities and accelerations (Kober and Peters (2009)).

In Kober and Peters (2009), a novel algorithm which combines motor primitives and RL based on expectation-maximization called Policy learning by Weighting Exploration with the Returns (PoWER) was introduced. It can be derived from the same higher principle as previous policy gradient approaches. This method was evaluated on two learning problems on a real, seven degree of freedom Barrett WAM—the discrete task of Ball-in-a-Cup and the rhythmic task Ball-Paddling. They first initialize the motor primitives by imitation and, subsequently, improve them by reinforcement learning. Interestingly, the robot fails to reproduce the presented behavior even if they used all the recorded details for imitation learning using the supervised learning approach for motor primitives described earlier.

6.2.2 Natural Actor-Critic (NAC)

Policy gradients approaches have rather strong convergence guarantees, even when used in conjunction with approximate value functions, and recent results created a theoretically solid framework for policy gradient estimation from sampled data (Sutton et al. (1999)), (Konda and Tsitsiklis (2003)). However, even when applied to simple examples with rather few states, policy gradient methods often turn out to be quite inefficient (Kakade (2001)), partially caused by the large plateaus in the expected return landscape where the gradients are small and often do not point directly towards the optimal solution (see Fig. 6.1).

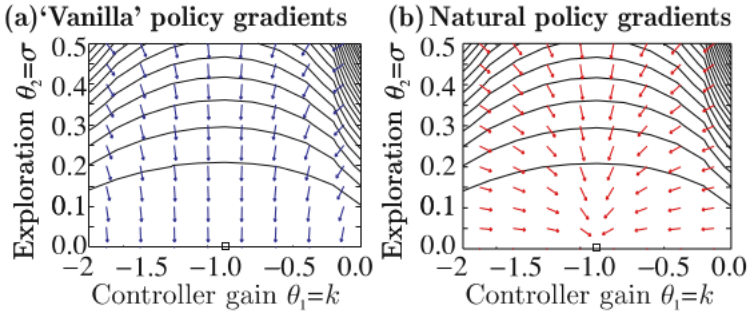


Figure 6.1: When plotting the expected return landscape for simple problem as 1d linear quadratic regulation, the differences between 'vanilla' and natural policy gradients becomes apparent. This image is from Peters et al. (2003).

Similar as in supervised learning, the steepest ascent with respect to the Fisher information metric (Amari (1998)), called the 'natural' policy gradient, turns out to be significantly more efficient than normal gradients. Such an approach was first suggested for reinforcement learning as the "average natural policy gradient" by Kakade (2001), and subsequently shown in preliminary work to be the true natural policy gradient by Peters et al. (2003).

Standard gradient descent follows the direction of steepest descent. However, this choice is not necessarily appropriate. It is better to define a metric based not on the choice of coordinates, but rather on the manifold (i.e., the surface) that these coordinates parameterize. Natural gradients solve this issue in supervised learning problems.

Actor-Critic and many other policy iteration architectures consist of two steps,

a policy evaluation step and a policy improvement step. The main requirements for the policy evaluation step are that it makes efficient usage of experienced data. The policy improvement step is required to improve the policy on every step until convergence while being efficient. The requirements on the policy improvement step rule out greedy methods as, at the current state of knowledge, a policy improvement for approximated value functions cannot be guaranteed, even on average. 'Vanilla' policy gradient improvements often get stuck in plateaus as explained in the previous discussion.

Actor-Critic and other policy iteration architectures consist of two steps - policy evaluation and policy improvement. A major requirement for the policy evaluation step is that it should make efficient use of the sampled data/experience. The policy improvement step is required to improve the policy efficiently on every step until convergence (Peters and Schaal (2008b)).

The requirements on the policy improvement step rule out greedy methods as, at the current state of knowledge, a policy improvement for approximated value functions cannot be guaranteed, even on average. 'Vanilla' policy gradient improvements which follow the gradient of the expected return function often get stuck in plateaus as described earlier in section 2.2.6. Natural gradients avoid this pitfall (Kakade (2001)). These methods do not follow the steepest direction in parameter space but the steepest direction with respect to the Fisher metric (Peters and Schaal (2008b)).

In Peters and Schaal (2008b), they evaluate the performance of the Episodic Natural Actor-Critic (eNAC) algorithm on standard RL benchmarks like Mountain Car and Cartpole as well as on a baseball swing robot example. Degris et al. (2012) apply the Incremental Natural Actor-Critic algorithms on Mountain Car, Cartpole and for the control of a myoelectric prosthesis. Peters and Schaal (2008a) use RL to learn motor skills with policy gradients. Their example of motor primitive learning for baseball underlines the efficiency of natural gradient methods for complex movement systems.

Peters and Schaal have shown that the original Actor-Critic and Bradtke's Linear Quadratic Q-Learning are in fact special cases of the Natural Actor-Critic al-

gorithms (Peters and Schaal (2006)). When we use a Gibbs policy $\pi(a_t|s_t) = \frac{\exp(\theta_{sa})}{\sum_b \exp(\theta_{sb})}$ and TD(0) for the critic, the update rules of the original Actor-Critic correspond to the ones of NAC and both algorithms are equivalent.

6.2.3 Apprenticeship Learning via Inverse Reinforcement Learning

As discussed earlier, learning from demonstration based techniques have also been called as imitation learning, apprenticeship learning, behavioral cloning and programming by demonstration. In traditional apprenticeship learning, the agent tries to replicate the demonstrations of the teacher. This can be achieved by using only supervised learning. While RL can be used here (as described in Chapters 3 and 4), it can be successful only if a good reward function can be specified. In a 'vanilla' learning from demonstration approach, the agent would try and mimic even irrelevant, sub-optimal actions or even mistakes.

Proponents of inverse reinforcement learning (IRL) argue that the reward function is often hard to specify (Abbeel (2008)). For example, how can we formulate a reward function for "driving well"?

Apprenticeship learning via inverse reinforcement learning tries to infer the goal of the teacher. In other words, it will learn a reward function by observing expert demonstrations, which can then be used later in Reinforcement Learning. Taking the previous scenario of hitting nails in a concrete wall as an example, if the agent discovers that the goal is to hit a nail with a hammer, it will ignore blinking or grunting noises from the human teacher, as they are irrelevant to the goal. In IRL, the reward function is not explicitly given. Instead, the agent observes an expert demonstrating the task it should learn to perform and recovers this unknown reward function. This recovered reward function is later used in an RL algorithm for control.

Abbeel (2008) was successful in applying apprenticeship learning via inverse Reinforcement Learning to the problem of autonomous helicopter flight. Their results include the first autonomous in-place flips, in-place rolls, loops and highly-challenging acrobatic maneuvers as a part of a complete airshow. Their controllers

perform as well, and often surprisingly better than their expert pilot.

6.2.4 Guided Policy Search

Direct policy search methods are often employed in high-dimensional applications such as robotics, since they scale gracefully with dimensionality and offer appealing convergence guarantees (Peters and Schaal (2008a)). However, it is often necessary to carefully choose a specialized policy class to learn the policy in a reasonable number of iterations without falling into poor local optima (for e.g., motor primitives (Ijspeert et al. (2003)). This specialization comes at a cost in generality, and can restrict the types of behaviors that can be learned. For example, a policy that tracks a single trajectory cannot choose different trajectories depending on the state. Levine and Koltun (2013) introduced a guided policy search algorithm that uses trajectory optimization to assist policy learning. The algorithm uses differential dynamic programming (DDP) to generate "guiding samples," which assist the policy search by exploring high-reward regions.

In Gupta et al. (2016), the robot autonomously learns to imitate object-centric demonstrations (i.e., a human demonstrates the desired motion of manipulated objects with their own hands) reinforcement learning. These object-centric demonstrations are provided by placing trackers on the objects being manipulated and using a human demonstrator to physically move the objects along the desired trajectories. Such demonstrations consist of just the trajectories of the object trackers, without any other states or actions. The algorithm introduced in Gupta et al. (2016) enables complex dexterous manipulators to learn from multiple human demonstrations, selecting the most suitable demonstration to imitate for each initial state during training. The algorithm alternates between softly assigning demonstrations to individual controllers, and optimizing those controllers with an efficient trajectory-centric RL algorithm (Guided policy search in Levine and Koltun (2013)).

6.3 Intelligent Behavior Principles Drawn from the Neurobiology of an Octopus

The rich behavioral repertoire of the Octopuses is an extraordinary biological example of motor control in a soft-bodied, flexible invertebrate. They are rather unique amongst invertebrates — they have an extremely large nervous system which has roughly half a billion neurons (as much as a dog brain). This helps them compete successfully with other vertebrates for the same ecological niche (Hochner (2012)).

The most obvious characteristic feature of an octopus is its eight long and flexible arms. These flexible, hyper-redundant arms endow it with high maneuverability but also pose a great challenge for achieving for precise goal-directed movements and coordinated locomotion. The task of processing the incoming sensory information and selecting proper motor commands places a great burden on the nervous system. First, coordinating movement is a formidable task since there are DOFs that have to be controlled; and second, it is extremely hard to use body coordinates (i.e., a frame of reference based on a rigid body structure) in this flexible animal to represent sensory information in a central control system. Skeletal animals evolved a solution for this problem by using central "representation maps" that represent the sensory and motor information in a way that maintains the spatial relationships of the body morphology (also known as *somatotopic* representation (Penfield and Boldrey (1937))) in the central nervous system. These representation maps likely serve as a "look-up table" for the brain to compute motor commands for feed-forward control. While somatotopic representations of this kind work for skeletal animals (limited number of joints and fixed configuration of skeletons limit the DOFs), it is computationally infeasible for an octopus (infinite DOFs and lack of fixed spatial relationships between flexible body parts).

The nervous system of the octopus is divided into three main parts: a central brain surrounded by a cartilaginous capsule; two large optic lobes connected to the retinae of the highly-developed eyes and the peripheral nervous system. Numerically, the peripheral nervous system is the most prominent part since it contains roughly two-thirds of the 500 million nerve cells (Young (1971)). These researchers

also deduced from the relatively few afferent and efferent nerve fibers interconnecting the three main parts that a significant portion of the sensorimotor information processing is performed in the peripheral nervous system and in the optic lobes, with the central brain acting as a coordinating and decision-making unit (Young (1971); Hochner (2012)).

The octopus nervous system has a unique organization — the peripheral nervous system performs computations usually attributed to the central nervous system of vertebrates. We've already established that an amputee does not have enough input signals to control every single DOF of an advanced prosthesis (e.g., MPL) that is capable of restoring full-functionality. We propose a high-level architecture similar to this unique octopus nervous system organization. The goal is to build a processing unit (similar to the octopus peripheral nervous system) which performs a bulk of the sensory and motor information processing and learns a rich behavioral repertoire (e.g., motor primitives). This would be a decentralized control approach where the agent would execute a set of motor primitive and the human brain would act only as a decision-making unit. For example, if an amputee subject looks at an object and triggers the robotic arm with a control command, the robot arm executes the most appropriate motor primitives and picks up the object. In such a scenario, the user chooses *what* or *which object* to manipulate and the robot learns *how* to execute the desired set of movements. In Chaper 5, we've shown that context-aware control of this type is feasible on a set of reaching and grasping tasks. While we used a small, limited representation to achieve those results, we could use all the sensory information available to learn a richer behavioral repertoire.

Although this approach promises to make great functional improvements for a prosthesis user, it is a big assumption to think that a prosthetic arm taking actions on its own will be acceptable to prosthesis users. If the users feel that they lack *agency*, they would reject such a prosthesis. Ultimately, we'd require a bi-directional neural interface where the high-level commands come directly from the nervous system and the sensor signals return directly to the nervous system. The users should *feel* that they are responsible for the actions of the prosthesis. In my opinion, Proprioception and exteroception feedback from the prosthesis should be

sent to the nervous system so that the human brain could re-learn to associate its actions with corresponding prosthetic joint movements.

Chapter 7

Conclusion

A major issue in the control of myoelectric prosthesis is the gap between the number of available functions in the prosthetic arm and the user's ability to control the desired joints at any given instant. Currently, the neural interface between the user and the prosthetic arm is functionally limited. The amputee user is unable to control every single DOF in advanced prosthesis (like the MPL) and receives zero/minimal proprioception information. This problem is not unique to the prosthesis domain but can also be found in many human-machine interfaces.

The overarching goal of my research has been to improve the control of prosthetic arms. We proposed a collaborative approach to myoelectic control that could exploit the amputee subject's repertoire of motor behaviours. An amputee would interact with his/her myoelectric prosthesis everyday and it's reasonable to assume that they would have some ideas and expectations about the device behaviour. Demonstration have the attractive feature of being an intuitive medium for communication from humans. We already use demonstrations to teach other humans (e.g., sports, music, martial arts, etc). This work presented a learning-from-demonstration framework based on ACRL that will potentially allow an amputee to use their non-amputated arm to teach their prosthetic arm how to move in a natural and coordinated fashion. To the best of our knowledge, this study is the first demonstration of the training an upper-limb myoelectric prosthesis with a user's contralateral limb (in this case we taught a left handed prosthetic arm with an intact right arm).

We carried out studies with both able-bodied and amputee subjects and showed that an ACRL agent can use these demonstrations to learn and generate accurate

hand and wrist synergies during testing and free-form control by a user. Though our experiments were limited to motions involving three DOFs, our approach could be easily extended to incorporate more DOFs and finer motions. These results would benefit from repeated experiments with more subjects. In addition, we can greatly improve the implementation to achieve a better performance. For example, we could use predictions as state information (Pilarski et al. (2013b)), alternative policy gradient methods which could potentially learn faster (as discussed in Section 6), select an appropriate timescale for learning, action persistence and predictions. Nevertheless, these preliminary results are promising and warrant further investigation.

In Chapter 5, we provided the system with additional sensory information (artificial vision) to enable context-aware control of the prosthetic arm. We used TD(λ) to make predictions about the target joint positions and were successful in learning a single value function that was able to distinguish between the objects. While the experiment by itself was heavily engineered to make it easier for the TD learner to make accurate predictions, we want to stress on the fact that the main objective was to show that an agent can learn to make context-aware predictions. Instead of using a simple red/green/blue pixel-counting scheme for visual features, we could use traditional image processing features (for example SIFT, Histogram of Gradients, watershed transform, etc.) or an end-to-end deep convolutional neural network. Assuming that we are able to teach the control learner multiple context-dependent muscle synergies, it would be interesting to see how well the controller could generalize to new tasks/situations.

We've applied RL techniques to myoelectric prosthesis with the goal of developing an agent that could learn and adapt to the needs of the user and changes in the environment. RL provides a nice set of tools that allow us to incorporate numerous sensory streams (vision, tactile sensation, myoelectric readings, inertial measurements, servo position, etc.) in ways that are computationally tractable and suitable for online learning. In the future, we need to continue enhancing these RL algorithms to make it more efficient and scalable.

LfD approches can be formulated such that they do not require expert knowl-

edge of the domain dynamics. This removes performance brittleness resulting from model simplifications. Since this approach does not require expert domain knowledge, non-robotics-experts can also help shape the control policy. For example, clinicians and patients are not required to know the implementation details of the system in order to specify a desired behavior. They can provide show various movements and grasp patterns to the system and the controller can use these high-level demonstrations to learn an appropriate control policy. Ideally, we imagine someone with an amputation could use this approach to continue to train a powered prostheses at home on a regular basis.

As discussed earlier, there exists a gross mismatch between the number of input channels available to the user and the number of controllable functions in a prosthetic arm. This issue is also faced in the human-computer interaction tasks. We expect that the LfD approach can be easily extended to human-computer interaction and any other domain which faces the degrees-of-freedom problem. In the long run, we expect these methods to improve the quality of life for people with amputations by providing them better ways of communicating their intentions and goals to their myoelectric prosthesis. We believe that we could extend the LfD approach in a straightforward way such that these devices may someday allow users to perform fluid and intuitive movements like playing the piano, catching a ball, and comfortably shaking hands.

Bibliography

Pieter Abbeel. *Apprenticeship learning and reinforcement learning with application to robotic control*. PhD thesis, Stanford University, 2008.

Edgar D Adrian and Detlev W Bronk. The discharge of impulses in motor nerve fibres. In *The Journal of Physiology*, volume 67, number 2, pages 9–151. Wiley Online Library, 1929.

Shun-Ichi Amari. Natural gradient works efficiently in learning. In *Neural Comput.*, volume 10, number 2, pages 251–276, Cambridge, MA, USA, February 1998. MIT Press. doi: 10.1162/089976698300017746. URL http://dx.doi.org/10.1162/089976698300017746.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. In *Robotics and Autonomous Systems*, volume 57, number 5, pages 469–483. Elsevier, 2009.

Manfredo Atzori, Henning Muller, and Micheal Baechler. Recognition of hand movements in a trans-radial amputated subject by semg. In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, pages 1–5, June 2013. doi: 10.1109/ICORR.2013.6650486.

Sharon Barbour. Soldier's mind-control bionic op. http://www.bbc.co.uk/news/uk-england-tyne-16702983, 2012. [BBC Online; accessed 25-Jan-20121].

Richard Bellman. *Dynamic programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957. URL http://books.google.com/books?id=fyVtp3EMxasC&pg=PR5&dq=dynamic+programming+richard+e+bellman&client=firefox-a#v=onepage&q=dynamic%20programming%20richard%20e%20bellman&f=false.

Hamid Benbrahim and Judy A Franklin. Biped dynamic walking using reinforcement learning. In *Robotics and Autonomous Systems*, volume 22, number 3-4, pages 283–302. Elsevier, 1997.

Nikolai Bernstein. *The coordination and regulation of movements*. Pergamon Press, Oxford, England, 1967. ISBN 0262193981. doi: 10.1016/j.brainres.2010.09.091.

Neil E Berthier, Michael T Rosenstein, and Andrew G Barto. Approximate optimal control as a model for motor learning. In *Psychological Review*, volume 112, number 2, pages 329–345. American Psychological Association, 2005.

Baldin Llorens Bonilla and H Harry Asada. A robot on the shoulder: Coordinated human-wearable robot control using coloured petri nets and partial least squares predictions. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 119–125, May 2014. doi: 10.1109/ICRA.2014.6906598.

Michael M Bridges, Matthew P Para, and Michael J Mashner. Control system architecture for the modular prosthetic limb. In *Johns Hopkins APL Technical Digest*, volume 30, number 3, pages 217–222, 2011.

Claudio Castellini, Panagiotis Artemiadis, Michael Wininger, Arash Ajoudani, Merkur Alimusaj, Antonio Bicchi, Barbara Caputo, William Craelius, Strahinja Dosen, Kevin Englehart, et al. Proceedings of the first workshop on peripheral machine interfaces: Going beyond traditional surface electromyography. In *Frontiers in Neurorobotics*, volume 8, number 22. Frontiers Media SA, 2014.

Andrea d'Avella, Alessandro Portone, Laure Fernandez, and Francesco Lacquaniti. Control of fast-reaching movements by muscle synergy combinations. In *Journal of Neuroscience*, volume 26, number 30, pages 7791–7810. Soc Neuroscience, 2006.

Michael R Dawson, Craig Sherstan, Jason P Carey, Jacqueline S Hebert, and Patrick M Pilarski. Development of the bento arm: An improved robotic arm for myoelectric training and research. In *Proc. of MEC'14: Myoelectric Controls Symposium*, pages 60–64, Fredericton, New Brunswick, Aug 8-22 2014.

Thomas Degris, Patrick M Pilarski, and Richard S Sutton. Model-free reinforcement learning with continuous action in practice. In *American Control Conference, ACC 2012, Montreal, QC, Canada, June 27-29, 2012*, pages 2177–2182, 2012. URL `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6315022`.

Ann L Edwards. Adaptive and autonomous switching: Shared control of powered prosthetic arms using reinforcement learning. Master's thesis, University of Alberta, Edmonton, AB, Canada, April 2016.

Ann L Edwards, Michael R Dawson, Jacqueline S Hebert, Craig Sherstan, Richard S Sutton, K Ming Chan, and Patrick M Pilarski. Application of real-time machine learning to myoelectric prosthesis control: A case series in adaptive switching. In *Prosthetics and Orthotics International*, volume 40, number 5, pages 573–581, 2016. doi: 10.1177/0309364615605373. URL `http://dx.doi.org/10.1177/0309364615605373`. PMID: 26423106.

J Randall Flanagan, Philipp Vetter, Roland S Johansson, and Daniel M Wolpert. Prediction precedes control in motor learning. In *Current Biology*, volume 13, number 2, pages 146–150. Elsevier, 2003.

Arjan Gijsberts and Barbara Caputo. Exploiting accelerometers to improve movement classification for prosthetics. In *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, pages 1–5, June 2013. doi: 10.1109/ICORR.2013.6650476.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013. doi: 10.1109/ICASSP.2013.6638947.

C Shawn Green and Daphne Bavelier. Action video game modifies visual selective attention. In *Nature*, volume 423, number 6939, pages 534–537. Nature Publishing Group, 2003.

Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3786–3793. IEEE, 2016.

Janne M Hahne, F Biessmann, Ning Jiang, H Rehbaum, Dario Farina, FC Meinecke, K-R Müller, and LC Parra. Linear and nonlinear regression techniques for simultaneous and proportional myoelectric control. In *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, volume 22, number 2, pages 269–279, March 2014. doi: 10.1109/TNSRE.2014.2305520.

Levi J Hargrove, Kevin Englehart, and Bernard Hudgins. A comparison of surface and intramuscular myoelectric signal classification. In *IEEE Transactions on Biomedical Engineering*, volume 54, number 5, pages 847–853, May 2007. doi: 10.1109/TBME.2006.889192.

Binyamin Hochner. An embodied view of octopus neurobiology. In *Current Biology*, volume 22, number 20, pages R887–R892. Elsevier, 2012.

Yonghong Huang, Kevin B Englehart, Bernard Hudgins, and Adrian DC Chan. A gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses. In *IEEE Transactions on Biomedical Engineering*, volume 52, number 11, pages 1801–1811, Nov 2005. doi: 10.1109/TBME.2005.856295.

Auke J Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems 15*, pages 1523–1530. MIT Press, 2003.

Mark Ison, Ivan Vujaklija, Bryan Whitsell, Dario Farina, and Panagiotis Artemiadis. High-density electromyography and motor skill learning for robust long-term control of a 7-dof robot arm. In *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, volume 24, number 4, pages 424–433. IEEE, 2016.

Ning Jiang, Strahinja Dosen, Klaus-Robert Muller, and Dario Farina. Myoelectric control of artificial limbs: Is there a need to change focus? [in the spotlight]. In *IEEE Signal Processing Magazine*, volume 29, number 5, pages 152–150, Sept 2012. doi: 10.1109/MSP.2012.2203480.

Sham Kakade. A natural policy gradient. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, pages 1531–1538. MIT Press, 2001. URL `http://books.nips.cc/papers/files/nips14/CN11.pdf`.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In Claude Sammut and Achim Hoffman, editors, *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, pages 267–274, San Francisco, CA, USA, 2002. Morgan Kauffman. ISBN 1-55860-873-7. URL `http://ttic.uchicago.edu/~sham/papers/rl/aoarl.pdf`.

Jens Kober and Jan Peters. Learning motor primitives for robotics. In *IEEE International Conference on Robotics and Automation*, pages 2112–2118. IEEE, 2009.

Jens Kober and Jan Peters. Reinforcement learning in robotics: a survey. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 579–610. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_18. URL https://doi.org/10.1007/978-3-642-27645-3_18.

Vijay R Konda and John N Tsitsiklis. On actor-critic algorithms. In *SIAM Journal on Control and Optimization*, volume 42, number 4, pages 1143–1166, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. URL http://web.mit.edu/people/jnt/Papers/J094-03-kon-actors.pdf.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, Alex K.Krizhevsky, Ilya Sutskever, and Geoffrey E Q. Weinberger Hinton, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

Michael F Land and Peter McLeod. From eye movements to actions: how batsmen hit the ball. In *Nature Neuroscience*, volume 3, number 12, page 1340. Nature Publishing Group, 2000.

Sergey Levine and Vladlen Koltun. Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1–9, 2013.

Marko Markovic, Strahinja Dosen, Dejan Popovic, Bernhard Graimann, and Dario Farina. Sensor fusion and computer vision for context-aware control of a multi degree-of-freedom prosthesis. In *Journal of Neural Engineering*, volume 12, number 6, page 066022. IOP Publishing, 2015.

Kory W Mathewson and Patrick M Pilarski. Simultaneous control and human feedback in the training of a robotic agent with actor-critic reinforcement learning. In *IJCAI Workshop on Interactive Machine Learning*, New York, July 9th 2016.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and K Kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2204–2212. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf.

David E Orin. Application of robotics to prosthetic control. In *Annals of Biomedical Engineering*, volume 8, number 4, pages 305–316. Springer, 1980.

Philip A Parker, Kevin B Englehart, and Bernard S Hudgins. Control of powered upper limb prostheses. In *Electromyography*, pages 453–475. John Wiley & Sons, Inc., 2005. ISBN 9780471678380. doi: 10.1002/0471678384.ch18. URL http://dx.doi.org/10.1002/0471678384.ch18.

Philip A Parker, Kevin B Englehart, and Bernard S Hudgins. Myoelectric signal processing for control of powered limb prostheses. In *Journal of Electromyography and Kinesiology*, volume 16, number 6, pages 541–548. Elsevier, 2006.

Bart Peerdeman, Daphne Boere, HJB Witteveen, HJ Hermens, Stefano Stramigioli, JS Rietman, PH Veltink, and Sarthak Misra. Myoelectric forearm prostheses: State of the art from a user-centered perspective. In *J. Rehab. Res. Dev,*, volume 48, number 6, pages 719–738, 2011.

Wilder Penfield and Edwin Boldrey. Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation. In *Brain*, volume 60, number 4, pages 389–443, 1937. doi: 10.1093/brain/60.4.389. URL +http://dx.doi.org/10.1093/brain/60.4.389.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225, 2006. doi: 10.1109/IROS.2006.282564.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. In *Neural Networks*, volume 21, number 4, pages 682–697. Elsevier, 2008a.

Jan Peters and Stefan Schaal. Natural actor-critic. In *Neurocomputing*, volume 71, number 7, pages 1180–1190. Elsevier, 2008b.

Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids2003)*, 2003. URL http://www-clmc.usc.edu/publications/p/peters-ICHR2003.pdf.

Patrick M Pilarski and Jacqueline S Hebert. Upper and lower limb robotic prostheses. In P. Encarnacao and A. M. Cook, editors, *Robotic Assistive Technologies: Principles and Practice*, chapter 4, pages 99–144. CRC Press, Boca Raton, FL, 2017. ISBN 978-1-4987-4572-7.

Patrick M Pilarski, Michael R Dawson, Thomas Degris, Farbod Fahimi, Jason P Carey, and Richard S Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, June 2011. doi: 10.1109/ICORR.2011.5975338.

Patrick M Pilarski, Michael R Dawson, Thomas Degris, Jason P Carey, K Ming Chan, Jacqueline S Hebert, and Richard S Sutton. Adaptive artificial limbs: a real-time approach to prediction and anticipation. In *IEEE Robot. Automat. Mag.*, volume 20, number 1, pages 53–64, 2013a. doi: 10.1109/MRA.2012.2229948. URL http://dx.doi.org/10.1109/MRA.2012.2229948.

Patrick M Pilarski, Travis B Dick, and Richard S Sutton. Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints. In *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on*, pages 1–8, June 2013b. doi: 10.1109/ICORR.2013.6650435.

Patrick M Pilarski, Richard S Sutton, and Kory W Mathewson. Prosthetic devices as goal-seeking agents. In *Second Workshop on Present and Future of Non-Invasive Peripheral-Nervous-System Machine Interfaces: Progress in Restoring the Human Functions (PNS-MI),*, Singapore, Aug 11 2015.

Tobias Pistohl, Christian Cipriani, Andrew Jackson, and Kianoush Nazarpour. Abstract and proportional myoelectric control for multi-fingered hand prostheses. In *Annals of Biomedical Engineering*, volume 41, number 12, pages 2687–2698,

2013. doi: 10.1007/s10439-013-0876-5. URL http://dx.doi.org/10.1007/s10439-013-0876-5.

Dimitris Pongas, Aude Billard, and Stefan Schaal. Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2911–2916, Aug 2005. doi: 10.1109/IROS.2005.1545257.

Ronald A Rensink. The dynamic representation of scenes. In *Visual Cognition*, volume 7, number 1-3, pages 17–42. Taylor & Francis, 2000.

Stefan Schaal, Peyman Mohajerian, and Auke Ijspeert. Dynamics systems vs. optimal control—a unifying view. In *Progress in Brain Research*, volume 165, pages 425–445. Elsevier, 2007.

Erik Scheme and Kevin B Englehart. Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use. In *J. Rehab. Res. Dev.*, volume 48, number 6, pages 643–660, 2011.

Craig Sherstan. Towards prosthetic arms as wearable intelligent robots. Master's thesis, University of Alberta, Edmonton, AB, Canada, Aug 2015.

Richard S Sutton and Andrew G Barto. *Reinforcement learning : an introduction*. MIT Press, 1998. ISBN 0262193981. doi: 10.1016/j.brainres.2010.09.091.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press. URL http://dl.acm.org/citation.cfm?id=3009657.3009806.

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

István Szita and András Lörincz. Learning tetris using the noisy cross-entropy method. In *Neural Computation*, volume 18, number 12, pages 2936–2941. MIT Press, 2006.

Jaden B Travnik and Patrick M Pilarski. Representing high-dimensional data to intelligent prostheses and other wearable assistive robots: A first comparison of tile coding and selective kanerva coding. In *Rehabilitation Robotics (ICORR), 2017 International Conference on*, pages 1443–1450. IEEE, 2017.

Anne Treisman. Features and objects in visual processing. In *Scientific American*, volume 255, number 5, pages 114–125, 1986.

Anne Treisman and Hilary Schmidt. Illusory conjunctions in the perception of objects. In *Cognitive Psychology*, volume 14, number 1, pages 107–141. Elsevier, 1982.

Anne M Treisman and Garry Gelade. A feature-integration theory of attention. In *Cognitive Psychology*, volume 12, number 1, pages 97–136. Elsevier, 1980.

John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. In *IEEE Transactions on Automatic Control*, volume 42, number 5, pages 674–690, 1997. URL http://www.stanford.edu/~bvr/psfiles/td.pdf.

Michael T Turvey, Hollis L Fitch, and Betty Tuller. The bernstein perspective: I. the problems of degrees of freedom and context-conditioned variability. In Kelso J A Scott, editor, *Human motor behavior: An introduction*, chapter 10, pages 239–252. Erlbaum Hillsdale, NJ, 1982.

Vivek Veeriah, Patrick M Pilarski, and Richard S Sutton. Face valuing: Training user interfaces with facial expressions and reinforcement learning. In *IJCAI Workshop on Interactive Machine Learning*, New York, USA, July 9th 2016.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, volume 8, number 3, pages 229–256, May 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Daniel M Wolpert, Jörn Diedrichsen, and J Randall Flanagan. Principles of sensorimotor learning. In *Nature Reviews. Neuroscience*, volume 12, number 12, page 739. Nature Publishing Group, 2011.

Thomas W Wright, Arlene D Hagen, and Michael B Wood. Prosthetic usage in major upper extremity amputations. In *The Journal of Hand Surgery*, volume 20, number 4, pages 619–622. Elsevier, 1995.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2048–2057, 2015.

John Z Young. *The anatomy of the nervous system of octopus vulgaris*. Oxford: Clarendon Press, 1971.

Appendices

Appendix A

ROS Infrastructure

This section is intended to serve as a reference for students/researchers interested in using the Delsys Trigno, CyberTouch II, Thalmic Myo Armband and the Bento Arm. Craig Sherstan had previously set up an elaborate control software for operating and interfacing with the Bento Arm using Robot Operating System (ROS), an open source robotics platform. I've built similar control softwares for operating the Delsys Trigno Wireless Lab, CyberTouch II and Thalmic Myo Armband on ROS. They can be used as independent systems which can communicate with the Bento Arm. The code repository can be accessed through the BLINC Lab Bitbucket repository. To researchers outside the BLINC Lab, please contact me for private access to the code base.

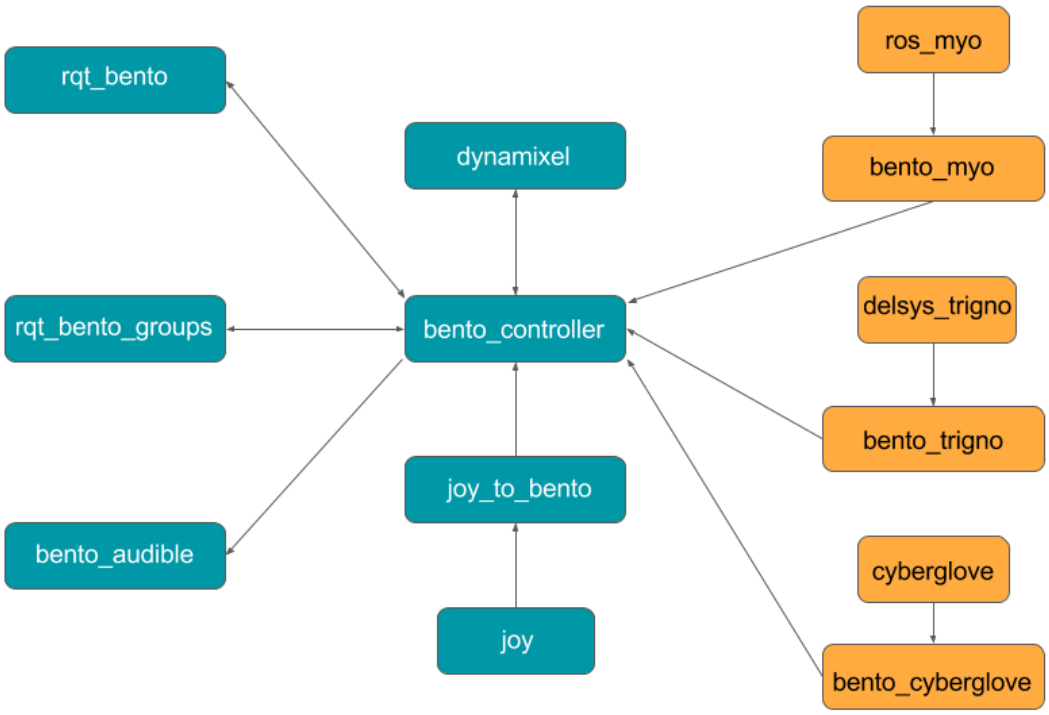


Figure A.1: Bento, Myo, Trigno and CyberTouch II Architecture.

Appendix B

Thalmic Myo Armband

This ROS package creates a ROS node which publishes raw data from the Thalmic Labs Myo Armband (tested with firmware version 1.1.4.2) in the form of both standard and custom ROS messages. These messages can be subscribed to and used in standard ROS architectures.

There are four topics generated by the $myo - rawNode.py$ node. These are:

- $/myo\_imu$ - a standard IMU message with quaternion pose, accelerometer and gyro axes
- $/myo\_arm$ - Arm: a custom arm Arm message populated after calibration that shows current arm and orientation on the arm
- $/myo\_emg$ - EmgArray: a custom message that is comprised of the EMG readings from the eight sensors
- $/myo\_gest$ - Gesture data populated after calibration (UInt8 value of enumerated poses)

The $multiple\_myo.launch$ file can be used to pair with two or more Myo Armbands simultaneously. The launch file requires appropriate arguments for device ID and the topic names.

Appendix C

Delsys Trigno Wireless Lab

The Delsys Trigno Wireless system is capable of streaming data digitally into EMGworks® or Trigno Control Utility. EMGworks® is a software for acquiring and analyzing EMG and other physiological signals. It is best suited when the only purpose of the study is to record data and process/use it post-hoc. While EMGworks® provides a nice, intuitive GUI for viewing and recording data, it is unable to communicate with other third party applications or drivers required for control.

Delsys also provides a software called the Trigno Wireless System SDK. It is a software package designed to allow programmers to interact with the Delsys Hardware. The SDK runs as a TCP/IP server with the *Trigno Control Utility*. It is important to note that Delsys Trigno works only on Windows Operating Systems (at least Windows 7).

The Trigno Control Utility can be used to pair sensors to desired slots (see the Trigno Wireless EMG System User Guide for help with pairing). While the Trigno Control Utility is running, any other software can connect to the command port and instruct the base to begin streaming data. The command port is port 50040 on the host (running the Trigno Control Utility) computer.

I've set it up such that the Windows laptop must be connected to ElsieVision Network (Internal Lab WiFi). Once Open Trigno Control Utility software. The Trigno units necessary for the trial can be chosen and turned on (Once the Trigno units are removed from the charging station, the button should be pressed once to switch it on.) To receive data, connect to the appropriate ports:

EMG Data	50041
Accelerometer Data	50042
IMU Data	50043
Quaternions or PRY (Pitch, Roll, Yaw) Format Data	50044

All data values are IEEE floats containing 4 bytes each. If multiple channels are being streamed over a port, the data are multiplexed. Data are streamed independently for EMG channels and accelerometer channels. The sampling rate is different for EMG and accelerometer channels—$2000Hz$ and $148.15Hz$ respectively. To compensate for this disparity, we read 27 EMG samples for every 2 IMU samples obtained. This helps us synchronize the data obtained across different sensors.

Reading EMG signals To maintain synchronization, I always read or processed bytes in multiples of 64 bytes. (16 channels * 4 bytes/channel = 64 bytes). 64 bytes of data will contain 1 EMG sample from each sensor.

Reading Accelerometer signals Accelerometer data are returned as 48 multiplexed channels (16 sensors * 3 axes/sensor = 48 channels), where each sample is an IEEE float occupying 4 bytes. Overall, it returns 192 bytes.

Reading IMU signals Inertial Measurement Sensor data are returned as 144 multiplexed channels (16 sensors * 3 axes/measurement unit * 3 measurements units = 144 channels), where each sample is an IEEE float occupying 4 bytes. To maintain synchronization, I always read or process bytes in multiples of 576 bytes (144 channels * 4 bytes/channel = 576 bytes). 576 bytes of data will contain 1 sample from each of the 144 channels.

Reading Quaternion signals If the data format is set to Quaternions (I've set that as default), the data will be a stream containing 80 multiplexed channels ([16 sensors]*[5 values/sensor] = 80 channels), where each sample is an IEEE float occupying 4 bytes. Overall, it returns 320 bytes per sample.

Appendix D

CyberTouch II

The Cyberglove (a.k.a CyberTouch II) is a 18-sensor data glove features two bend sensors on each finger, four abduction sensors, plus sensors measuring thumb crossover, palm arch, wrist flexion, and wrist abduction. It uses proprietary resistive bend-sensing technology to accurately transform hand and finger motions into real-time digital joint-angle data.

The Cyberglove drivers are compatible with Windows 7. It can connected to a PC using a serial to USB converter. I've written a script with the help of researchers from Cleveland Clinic to use the Cyberglove system outside its proprietary software. The Cyberglove SDK on windows can be used to visualize the hand movements and calibrate different finger and hand motions. This calibration file is required by the system to provide accurate joint angles.

We do not get access to the raw values of each sensor on the Cyberglove. Instead, it has a processing unit (a small embedded processor like a raspberry pi) which gets the raw inputs, processes it and sends this processed information to the PC. The sampling rate for the pre-processed joint angle information is $\sim 25Hz$.

The python script $Cyberglove\_server\_python27\_idle.py$ connects with the CyberTouch II system and creates a server which is capable of transmitting the sensory information to a client via TCP. This script would load a calibration file in case of successful connection with the CyberTouch II. Once the client sends the "connect" command and the experiment time, the server starts transmitting TCP packets to the client. By default, the host_ip of the Windows laptop on *ElsieVision Network (BLINC Lab WiFi)* is $10.0.1.120$ and the port through which Cyberglove is stream-

ing data is 5050. The port number can be changed on this script if necessary.

Appendix E

Integrating Bento Arm with Trigno & CyberGlove

The central controller for the Bento arm is $bento\_controller$. This node is responsible for facilitating all interactions with the arm. It is intended to allow the arm to be used with various configurations of joints an motors, facilitated by the use of configuration files. It provides methods for moving each joint individually, either by position, velocity, or position and velocity based commands. State information is published at regular intervals.

The $bento\_trigno$ node subscribes to the topics published by $delsys\_trigno$. The $bento\_trigno$ node allows pairs of channels to be configured to send proportional control messages to a multiple joints. The EMG channel messages are first rectified. There are several online parameters that affect each channel: gain, threshold, max. Gain adjusts the amplification on a particular channel. The threshold allows an operator to adjust for noise; signals below this value are ignored. Output of each channel is scaled from 0 to 1 across threshold and max, i.e., threshold produces a 0 value, while max produces 1, values higher or lower are capped. The max setting is used for adjusting the sensitivity of the control signal to the magnitude of contraction, i.e., if max is set higher then the user will need to produce a stronger contraction in order to send a 1. One of the channels in a pair produces positive group commands, while the other produces negative group commands.

The **cyberglove** node receives the pre-processed sensor readings obtained from the CyberTouch II. The $bento\_cyberglove$ node allows channels to be configured to

send position control messages to multiple joints.