Prosthetic Arms as Wearable Intelligent Robots

by

Craig Sherstan

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

Master of Science

in

Computing Science

University of Alberta

Abstract

Prosthetic Arms as Wearable Intelligent Robots

Craig Sherstan

The control of powered prosthetic arms has been researched for over 50 years, yet prosthetic control remains an open problem, not just from a research perspective, but from a clinical perspective as well. Significant advances have been made in the manufacture of highly functional prosthetic limbs, yet the control of such limbs remains largely impractical. The core issue is that there is a significant mismatch between the number of functions available in the arms and the number of functions an amputee can actively attend to at any given moment.

One approach to addressing this mismatch is the idea of treating the arm as an intelligent, goal-seeking agent — such an agent learns from its experience and adapts its actions to improve its ability to accomplish a goal. This is the perspective taken in this thesis. The work of this thesis looks at several aspects of building agency into a prosthetic arm, including pattern recognition methods, compound predictions, and collaborative control between the arm and the user. Essentially, we look at ways of understanding the user's desires and controlling the arm so as to achieve those desires.

The first contribution of this thesis is the identification of a scenario under which current pattern recognition approaches to prosthetic control do not generalize well. The second contribution is the demonstration that it is possible to layer predictors, known as general value functions, and that such layering can improve feature representation and predictive power. Finally, this thesis demonstrates a method for improving the control of a prosthetic arm using a collaborative control method which learns predictions of user behavior which are then used to assist in controlling the arm.

In the long term, the methods and philosophy to prosthetic control explored in this thesis may greatly improve an amputee's ability to control their prosthesis. Further, this approach may be extended to other domains of human-machine interaction where there is a mismatch between the number of functions in a system and the user's ability to attend to those functions, such as smart phones, computers and teleoperated robots.

Committee

Kelvin Jones, Ph.D.:	Associate Professor Faculty of Physical Education and Recreation University of Alberta
Patrick M. Pilarski, Ph.D.:	Assistant Professor Division of Physical Medicine and Rehabilitation Department of Medicine University of Alberta
Richard S. Sutton, Ph.D.:	Professor Computing Science University of Alberta

Committee Chair:

Russ Greiner, Ph.D., Professor, Computing Science, University of Alberta

Preface

Chapter 3 of this thesis is based on a report submitted for credit during this MSc for CMPUT 551 in 2013. It was submitted as Sherstan, C., Ramirez, O., Ahmad, Z. F., and Zhang, H., "Classification of Embedded EMG Signals Amidst Concurrent Contractions". I was responsible for proposing the research topic and significant portions of the experiment and manuscript writing. For inclusion in this thesis the experiments were rerun and additional analysis was performed. The text of the chapter was largely rewritten as well.

Chapter 4 is based on work which was published and presented as Sherstan, C., Pilarski, P. M., "Multilayer General Value Functions for Robotic Prediction and Control", IROS 2014 Workshop on AI and Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, Illinois, September 14-18, 2014. I was responsible for all the experiments presented in this paper, which were originally performed for credit during this MSc for CMPUT 656 in 2014. I was the primary author and presenter of the manuscript. Pilarski contributed to the manuscript.

A version of Chapter 5 has been accepted for publication and presentation as Sherstan, C., Modayil, J., and Pilarski, P. M., "A Collaborative Approach to Effecting Simultaneous Multi-joint Control of a Prosthetic Arm", International Conference on Rehabilitation Robotics (ICORR), Singapore, August 2015. An extended abstract of this paper was also presented as a poster and podium talk at the Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM), Edmonton, Alberta, June 7-10, 2015. I proposed the approach and was responsible for experimental design, implementation, execution, and analysis. I was responsible for the bulk of manuscript composition. Modayil contributed helpful implementation discussions and was involved in manuscript editing. Pilarski was the supervisory author and contributed to the manuscript and provided implementation feedback and insights throughout.

Appendix A is related to the development of control code for the Bento Arm, a custom built robot arm used in the experiments of Chapter 5. A technical paper was presented on the Bento Arm as Dawson, M.R., Sherstan, C., Carey, J.P., Hebert, J.S., Pilarski, P.M., "Development of the Bento Arm: An Improved Robotic Arm For Myoelectric Training and

Research", Myoelectric Controls Symposium (MEC), Fredericton, New Brunswick, August 18-22, 2014, pp. 60-64. I was responsible for developing all control software for the arm and contributed a small portion to the manuscript.

Dedication

Dedicated to Jen, Nika and Kai. Jen thanks for putting up with all my random thoughts about robots, cyborgs, and neuroscience. I have no idea how I convinced you to let me do this, but thanks.

To my parents, Terry and Myrna. Dad, you told me when I was little that I would make my living with my mind... so I guess this is your fault :) Thank you both for your encouragement and support over the years and for not making me be a doctor or a hockey player.

Thank you to all the sci-fi writers out there who have shaped my mind in strange and exciting ways.

Acknowledgments

I would like to thank Patrick M. Pilarski. Not only did he provide technical direction and help me hone my research skills, but he is the singular most positive and enthusiastic person I have ever known. His upbeat outlook, his excitement for our research and his continual encouragement have made this a fantastic experience.

I'd also like to thank Rich Sutton. Beyond his knowledge of reinforcement learning and artificial intelligence, Rich provided me with the critical feedback which has, thus far, been largely absent from my career. The experience was extremely beneficial and shaping and I like to think that it has made me a better scientist.

Further, I'd like to thank Joseph Modayil, Adam White, Harm van Seijen and Michael Rory Dawson for all of their help along the way.

Finally, I would like to thank the various funders and organizations who made it possible, not just for me to complete this work, but to be able to even do an MSc in the first place: the Reinforcement Learning and Artificial Intelligence lab (RLAI), the Bionic Limbs for Improved Natural Control lab (BLINC), the National Sciences and Engineering Research Council of Canada (NSERC), Alberta Innovates Technology Futures (AITF), Alberta Innovates Centre for Machine Learning (AICML), the Alberta government, and of course the University of Alberta.

Contents

List of Tables

List of Figures

Glossary

agent An entity capable of learning and making decisions. ii, 1, 2, 13–15, 19, 21, 24, 53, 61, 68, 91, 94, 95, 97, 99, 106, 107, 109

AI Artificial Intelligence. 3, 47, 50

AICML Alberta Innovates Centre for Machine Learning, University of Alberta, Edmonton, Canada. 119

AR Autoregressive Model. One type of model used to represent how the current value of a signal is related to recent past values of the same signal. 27, 30, 33, 34, 39, 42, 44

BCI Brain Computer Interface. A device which interfaces directly with the brain. 9

BLINC Bionic Limbs for Improved Natural Control lab, University of Alberta, Edmonton, Canada. 2, 119

bootstrapping While this term has different meanings across different fields, it is used here to mean making predictions based on other predictions. 15

DAQ Data Acquisition device. An electronic device used to read in electronic signals into a computer. 36, 123, 124

DARPA Defense Advanced Research Projects Agency. 8–11, 108

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. 6–9, 13, 66–69, 71, 103, 107

DOF Degrees of Freedom. The number of independent motions a system can produce. 6–8, 28, 53

DPCC Direct Proportional Collaborative Control. The method of autonomy demonstrated in chapter 5. 3, 67–69, 71, 73, 74, 77–81, 84–90, 93, 96, 103–105

ECoG Electrocorticography. The measurement of electrical activity in the brain by draping a grid of electrodes over the cortex of the brain. 9

EEG Electroencephalogram. The measurement of electrical activity in the brain as measured through the scalp. 9, 101

EMG Electromyography. The recording of electrical signals generated by muscle contraction. v, 4, 8–11, 23, 26–29, 31–33, 35–37, 42–44, 48, 52, 66, 72, 77, 90, 93, 98, 107, 123–125

environment Everything outside of the agent. 1, 2, 4, 13, 14, 17, 19, 27, 46, 50, 53, 61, 90, 102, 106, 109

FDA Food and Drug Administration. 11

function approximation Function approximation involves representing a complex or unknown target function using a simpler well-known function. 15

GUI Graphical User Interface. 122, 125

GVF General Value Function. General value functions are a reinforcement learning technique for making temporally-extended predictions of measurable signals. 6, 19–24, 47, 50–54, 57, 58, 60, 61, 63, 67, 69, 71–73, 79, 81, 84, 86–89, 92–96, 98–101

IMES Implanted Myoelectric Sensors. Myoelectric sensors implanted directly in muscle tissue. 10–12

IMU Inertial Measurement Unit. An electronic sensor used for measuring velocity, orientation, and acceleration. 10

intelligent An intelligent agent is one that is able to learn about itself, its environment,and the interaction between the two and adapt its actions to improve its ability to accomplish a goal. ii, 1, 2, 27, 46, 61, 85, 91, 102, 106–109

KNN K-Nearest Neighbors. An unsupervised machine-learning algorithm used to separate data into K clusters. 31, 34, 42

LDA Linear Discriminant Analysis. A supervised machine-learning algorithm used for classification and dimensionality reduction. It attempts to find linear combinations of independent variables, that best describe the dependent variable. Further, it assumes that class data is represented by normal distributions, with all classes have the same covariance. 11, 27, 30, 31, 34, 37, 42, 44

off-policy Learning about a target policy, while following a different behavior policy. 19, 20

offline Offline machine learning methods separate the collection of samples from the learning from those samples. That is, there is a sample collection phase, followed by a learning phase. 17, 18, 26, 46

on-policy Learning about the behavior policy. 19

online Online machine learning methods are those which learn from each data sample as it occurs. 15, 18, 26, 46

policy A policy is a way of behaving. It is a deterministic or stochastic mapping from a state to an action. 14, 19, 60, 67, 88

RL 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. 1, 6, 13, 14, 24, 50, 53, 60, 67, 88

ROS Robot Operating System. An open-source framework for robotics software. 36, 55, 119, 124, 126, 127

sEMG Surface Electromyography. Electromyography as measured on the surface of the skin. 7, 11–13

SNR Signal-to-Noise Ratio. 9, 10

state The condition of the universe and everything in it at a given moment in time. In real-world scenarios only a reduced state representation is ever available or perceivable to an agent. x, 14–18, 20, 21, 50, 53, 58, 59, 62, 67, 81, 84, 86–90, 92–97, 99–102, 105

SVM Support Vector Machine. A supervised machine-learning method for classification whereby the algorithm specifically focuses on a defining a hyperplane which maximizes the distance between itself and the data points along the boundary between two classes. 11, 35

TD Temporal Difference. A set of algorithms associated with reinforcement learning which allow predictions to be updated incrementally, while blending between bootstrapping and Monte Carlo updates. 15–17, 19, 51, 52, 58, 79, 81, 95–97

TD Time domain. A set of features commonly used in pattern recognition with myoelectric signals. 27, 30–32, 37, 39, 42, 44

TMR Targeted Muscle Reinnervation. A surgery whereby motor nerve endings are transplanted into new host muscle. 13, 27, 29, 35, 43, 44

TPC Toggling Proportional Control. The name given to describe the most common form of myoelectric powered prostheses control, whereby a user controls a single joint in proportion to a scalar signal, and selects between available joints using a binary toggle signal. 8, 9, 11, 66, 67, 71, 72, 77–79, 89

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. 13

Notation

S, R, A — random variables are indicated by capital letters

$\mathcal{S}, \mathcal{A}$ — calligraphics are used to indicate sets

$\boldsymbol{w}, \boldsymbol{\phi}, \boldsymbol{e}$ — vectors are indicated in bold

$\mathbb{R}^{\mathrm{n}}$ — set of n reals

$^\top$ — Transpose operator

Chapter 1

Increasingly Intelligent Prosthetics

The overarching goal of this thesis is to develop control frameworks for assistive robots. Here I define assistive robots to be ones that directly and physically interact with humans to help them in achieving their goals. In particular, I focus on the control of prosthetic arms using reinforcement learning (RL) techniques. This imposes certain constraints on the system, which are sometimes in conflict with one another. Specifically, there is a need to balance the desire to engineer the best solution with the desire to create solutions general enough that they can be applied to other assistive domains as well. Practically, this means that at times it may seem most appropriate to use a heuristic approach to solving a particular problem. However, I have chosen to look at how the problem might be solved using reinforcement learning techniques instead, as, in the long run, I believe these will provide more mature and robust solutions that need not be domain specific.

The word *intelligent*, as used in the title of this thesis, has many meanings to many different people. In this thesis the word is used to mean a system, or agent, which is able to learn about its environment and adapt its actions to improve its ability to accomplish a goal. Intelligence should be viewed as a scalar value rather than a binary one, with agents having levels of intelligence. There are several key verbs in this definition: learn, adapt, act, and improve, all of which are directed towards accomplishing a goal. This thesis focuses primarily on the actions of learning and adapting. It is my hope that by the end of this thesis, the reader will be persuaded to think of prosthetic devices as platforms for intelligent agents, rather than simply electro-mechanical tools.

With this definition in mind, typical commercial prosthetics are *stupid*, they do not understand the user, the environment or the current situation, they simply respond to a user's commands. They are therefore limited by the command interface itself, i.e., the arm cannot do anything more than what the user can directly tell it to do. In fact, one of the main problems with current prosthesis control is that the number of controllable functions far exceeds the number of functions that the amputee can attend to at any given time. This is a common problem, not just in prosthetics, but in human-machine interfaces in general. One of our goals in the Bionic Limbs for Improved Natural Control (BLINC) lab is to create intelligent systems that bridge this gap in control; we want to develop prosthetic arms that understand an amputee's needs, anticipating them and taking appropriate action to assist the amputee. Such arms must adapt to changes in the user's behavior, in their environment, and their own capabilities. Ultimately, we strive to build a highly intelligent robot that happens to be worn as an amputee's arm. This thesis describes early work on developing such robots. Viewing devices as *intelligent* agents may provide powerful insights and approaches to many domains beyond prosthetics.

However, the improvement of prosthetic control is itself a valuable goal. It is estimated that in 2005 there were 1.6 million persons living with amputations in the United States, 8% of which are consider major upper-limb. This number is predicted to double by 2050 (Ziegler-Graham et al., 2008). Additionally, one study indicates that during US military operations in the middle East from 2001 to 2011, 1573 service persons received major amputations (Fischer, 2014). Another study tells us that nearly 14% of those amputations were upper limb (Krueger et al., 2012). Beyond the immediate physical and emotional trauma endured by those who lose a limb, there are the obvious long-term consequences: loss of functionality, social stigma, and loss of independence. Addressing these needs would have incredible benefits to the individuals and their families. Congenital amputees, those missing a limb since birth, may also benefit from the same technologies developed to address these issues.

1.1 Outline

This thesis is written in the *mixed* format as outlined by the University of Alberta Computing Science guidelines and consists of three main investigations, all of which are directly related to the control of prosthetic arms.

Chapter 2 will first provide a background on current methods used for prosthetic control as well as present the reinforcement learning techniques that are used in Chapters 4 and 5.

Chapter 3 presents a study highlighting some of the limitations of current pattern recognition methods used for prosthetic control. The experiment described involves the recognition of hand motions amidst concurrent contractions of the forearm muscles. This is used as an analogy for concurrent contractions of host muscle used in targeted muscle reinnervation surgery, which is described in 2.2.4.

Chapter 4 is the first presentation of reinforcement learning methods as applied to controlling a prosthetic arm. Additionally, this chapter discusses the control of prosthetic arms as a domain for research in artificial intelligence (AI). Specifically, this chapter looks at how predictive knowledge can be layered to form higher level predictions for robot control and proposes several ways in which this might be applied to the control of prosthetic arms.

Finally, in Chapter 5, a novel control algorithm, *Direct Predictive Collaborative Control* (DPCC), is evaluated using a robotic arm in two different experiments. In this control algorithm the arm learns to make predictions about target joint angles directly from user behavior and then assists the user in completing tasks by moving towards those targets before the user is able to do so themselves. These experiments show that the control algorithm can improve user task performance and reduce burden on the user. This chapter should be considered the most significant contribution of this thesis.

Lastly, Chapter 6 presents future directions of research, some of which are taken from the previous chapters. This chapter is useful for seeing how many different approaches from prosthetic control, pattern recognition, and reinforcement learning might be brought together.

1.2 Contributions

The contributions of this thesis are summarized as follows:

1. **Contribution (Chapter 3):** The identification of a scenario under which current *state-of-the-art* pattern recognition methods used in EMG based control of prosthetic arms do not generalize well.

   Significance: This finding highlights a realistic failing of the pattern recognition methods that are presently being used and treated as state of the art. In fact, this raises concerns over whether or not a current commercially available pattern recognition system might also suffer from the same problem. Further, this finding suggests that additional evaluation criteria or environments should be used when evaluating the performance of pattern recognition systems. In fact, this may signify the need to rerun numerous studies which compared recognition accuracy of various features and classifiers.

2. **Contribution (Chapter 4):** The demonstration that layers of predictors, known as general value functions, can be used to produce compound predictions.

   Significance: The ability to produce compound predictions, that is, predictions based on other predictions, holds significant value in the field of artificial intelligence as it provides a way of representing knowledge of a robot, its environment, and the interaction between the two. General value functions have the added benefit of being able to produce temporally extended predictions in a way that is computationally efficient and performed in real time. Being able to create compound predictions allow us to build more complex representations and to do so in ways that are efficient.

3. **Contribution (Chapter 4):** The demonstration that using such layers of general value functions can produce a boosting like effect whereby a secondary layer predictor can produce more accurate predictions than the primary layer predictors which are input to it.

   Significance: This observation demonstrates additional benefits to the concept of layering general value functions and may suggest a method by which other problems in robotics might be addressed, such as the integration of multiple sensors, or transfer learning.

4. **Contribution (Chapter 5):** The introduction of the *Direct Predictive Collaborative Control* method and the demonstration that it can be used to assist a user, who is bound to controlling only a single joint at a time, in completing tasks quicker and with less effort.

   Significance: Using this control method demonstrates the ability to achieve useful simultaneous multijoint control of a robot arm, which is key for producing smooth, natural motions. This is something that is currently not even possible for most powered prosthesis users and has the potential to significantly improve their control of their arms. Additionally, the concepts demonstrated here, if not the exact implementation, have the potential to be used in many other settings in which the number of controllable functions surpass a user's ability to attend to them at any instant in time.

5. **Contribution (Appendix A):** Development of the control software for the Bento Arm.

   Significance: While not a main part of my thesis, the software developed for the Bento Arm was a significant measurable output of the work of my MSc degree. The Bento Arm is an important piece of our lab infrastructure and is and will be used for many research projects other than my own.

Chapter 2

Background

This chapter will cover the requisite background for the rest of the thesis. It will start off by describing the key problem with prosthetic arm control and presenting current approaches to solving this problem. I will look at methods for collecting control signals from an amputee, methods for processing those signals, and even a surgical method, known as targeted muscle reinnervation used to make more of those signals available.

The chapter then introduces a branch of artificial intelligence known as reinforcement learning (RL). A predictive RL method, known as general value functions (GVFs), is described in detail. Finally, a method of creating features from real-valued signals, known as tile coding, is explained. Both GVFs and tile coding are key methods used extensively in this thesis.

2.1 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. That is, the DOC can be less than, equal to, or greater than the DOF of a system.

Take, for example, a hand, which has 27 DOF, if all a user can control is an open/close function, then the hand has only 1 DOC.

There are many domains in which the DOC are significantly more than what the user can attend to at any given moment. Clear examples include using a computer, a smartphone, or any system which requires a menu to operate. Aggregating many DOF into fewer controllable functions is often an effective approach in many domains. However, there are many for which this aggregation is not desirable.

2.2 Prosthetic Arm Control

The control of powered prosthetic arms is one such domain where the DOC can greatly outnumber the number of input channels the user has available, with disparity increasing as level of amputation increases, e.g., those with transhumeral amputations can provide even fewer control signals than those with hand amputations. Unfortunately, this mismatch makes the control of powered prosthetic arms difficult and tedious. In fact, many amputees will reject prostheses outright, while others will eschew electrically powered prostheses for mechanical, body-powered ones. Biddiss and Chau (2007) looked at 22 studies ranging over 25 years, which list rejection rates of upper limb myoelectric prosthesis anywhere from 0 to 75 %. While these figures are not overly helpful, other studies have indicated that of those who reject prosthesis, control and limited functionality were cited as two of the chief causes (Biddiss and Chau, 2007; Peerdeman et al., 2011; E. Scheme and Englehart, 2011; Resnik, Meucci, et al., 2012). While many other issues for powered prosthetic usage have been identified such as noise, weight, comfort and lack of sensory feedback, the focus of this thesis is on improving control.

Since the 1960s the most common way to control a powered prostheses has been through the use of surface myoelectric control or electromyography (sEMG), which involves measuring muscle contractions by the electrical signals they produce as read by sensors worn on the skin. The magnitude of a muscle contraction is used to drive a function or joint of the prosthetic using proportional control; the stronger the contraction the faster or stronger the

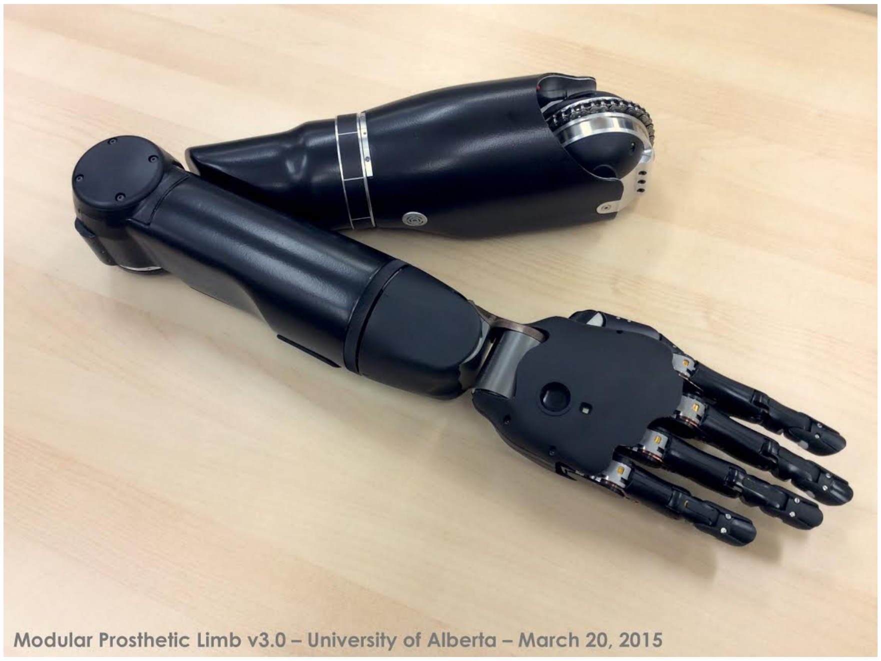

Modular Prosthetic Limb v3.0 – University of Alberta – March 20, 2015

Figure 2.1: The Modular Prosthetic Limb developed by Johns Hopkins University as part of the Revolutionize Prosthetics project funded by DARPA

movement. This method thus requires two contraction sites to control a single joint; one site (e.g., biceps) is used to move the joint in one direction and another site (e.g., triceps) is used to move the joint in the other direction. When more than one joint is available for control a third signal of some kind is used to toggle between joints, using a fixed joint order. This signal might be provided in the form of a third EMG site, a co-contraction of the two existing sites, or through a physical switch the user must press. For brevity, let us refer to this form of control as *toggling proportional control* (TPC).

For a prosthetic with only one or two DOC this can be an effective control system. However, such prostheses are of limited functional use. In recent years the capabilities of prosthetic arms have been greatly increasing due to advances in technology, enabling arms with nearly the same number of DOF as natural human arms. This is in large part due to an initiative of the *Revolutionizing Prosthetics* project of the Defense Advanced Research Projects Agency (DARPA). This project produced two state-of-the-art prosthetic arms, each with approximately 20 DOC: the Modular Prosthetic Limb (Johns Hopkins University) (see Figure 2.1) and the DEKA arm (DEKA Research & Development Corporation).

As one can imagine, using TPC to control large numbers of DOC, as are available in these arms, creates a staggered approach to movement without the natural synergies an able-bodied person would produce. This is discussed in greater detail in Chapter 5. New strategies are needed in order to improve amputee control. The following sections describe several approaches in development.

2.2.1 Alternative Control Signals

Given that the main issue with controlling prosthetic arms via EMG is the limited number of signals it is natural to wonder what other signals might be available. It is my opinion that the ideal interface for restoring arm function would use a bi-directional neural interface, such that natural control and sensation occur between the amputee and the prosthetic in the same way as would have occurred in the natural arm prior to amputation. Work in the area of brain computer interfaces (BCIs) includes: implantation of microelectrode arrays in the cortex, peripheral nerve cuffs, and electrocorticography ECoG, in which an array of sensors are draped over the cortex. While there has been notable success with using these approaches for control (Ohnishi et al., 2007; Wang et al., 2013; Collinger et al., 2013), significant technical and regulatory hurdles remain to be overcome and seeing such approaches in common clinical practice is unlikely in the near future. It should be noted that DARPA funded efforts to develop BCIs for prosthetic control through the *Reliable Neural Interface Technology* program are expected to conclude in 2015 (Miranda et al., 2015).

Such methods are not without their drawbacks. The biggest being that they are invasive, which increases the risks of their use and makes them more costly to develop, implement, and deploy. Some amputees may also find such methods objectionable for various other personal reasons. An alternative, non-invasive approach is the use of electroencephalogram (EEG), which measures the electrical activity of the brain transmitted through the skull, requiring users to wear a grid of sensors on their head. While significant work has been performed in this area, EEG faces several key difficulties including low signal to noise ratio (SNR), and low spatial resolution. Additionally, while it may never be possible to use EEG to directly control a prosthetic arm with the desired accuracy, it still may offer other useful signals, such as a user's satisfaction with a behavior (Esfahani and Sundararajan, 2011). Such signals may serve to complement other approaches. Specifically, they may be of use as a reward signal for reinforcement learning systems like those described in Chapter 2.3.

While working on DARPA's *Revolutionizing Prosthetics* project, DEKA also developed a novel control interface consisting of inertial measurement units (IMUs) worn in shoes in conjunction with torso worn bump switches (Resnik, Lieberman Klinger, et al., 2014). Amputees operated the arm by shifting pressure on their feet. This approach was evaluated over three different generations of the device with 36 subjects having different levels of amputation. By the third iteration, feedback was generally positive from users. Nonetheless, users required a significant training period, with those users having higher levels of amputation requiring more training time. To be effective at using this system, users had to become proficient at pre-planning their motions, requiring significant cognitive load. Perhaps the biggest drawback to this method is that users had to give up some degree of mobility.

2.2.2 Alternative Ways of Reading Muscle Contractions

As has been mentioned, surface EMG reads muscle contractions through electrical signals measured by electrodes in contact with the surface of the skin. However, it is prone to numerous difficulties with SNR caused by: sensor slippage, sensor alignment, changes in conductivity due to perspiration, degradation of signals due to fatigue, orientation dependent signal modification, cross-talk from other muscles and interference from underlying muscle contractions as discussed in Chapter 3. Alternative methods of detecting muscle contractions are being investigated such as topographic force mapping, where high density arrays of force sensors line the socket, and ultrasound imaging (Castellini, Artemiadis, et al., 2014). Topographic force mapping is an interesting approach that seems to be within the realm of clinical feasibility, while ultrasound imaging still requires some technical hurdles to be overcome, such as the miniaturization of the ultrasound sensors. Regardless, both techniques would still be subject to the same data processing approaches currently used.

Yet another approach is the use of implanted myoelectric sensors (IMES), in which microsensors are directly implanted in the muscle tissue (L. J. Hargrove, Englehart, et al., 2007; McDonnall et al., 2012). Such sensors have been developed in conjunction with DARPA and are undergoing clinical evaluation and FDA approval. One study indicated no significant benefit in classification accuracy of IMES over sEMG (L. J. Hargrove, Englehart, et al., 2007). However, the study did not look at classification accuracy in respect to the previously mentioned causes of poor signal quality, which are precisely the areas where IMES offer potential benefit. It is my opinion that, in the long run, IMES hold great promise for improving the reliability of EMG signals. IMES is more invasive than sEMG, but certainly less invasive than brain implants. While IMES is of great interest it is not explored further in this thesis and the reader should assume that surface based methods are used wherever EMG is mentioned.

2.2.3 Pattern Recognition

Pattern recognition, or classification, is a common machine learning technique in which a mapping from input signals to output class is learned. That is, all observed inputs are labeled as belonging to one of a set of classes. From this data, the pattern recognition algorithms then learn how to map input features to the most likely class. Pattern recognition of EMG signals is quite possibly the most studied alternative to the standard toggling proportional control. Typically, a number of predefined movements are specified, such as: wrist extension/flexion, wrist pronation/supination, radial deviation, ulnar deviation, hand open/close, elbow flexion/extension and rest. The amputee then performs a series of training movements that are matched to the classes. These labeled samples are then used to train the classification system. After training, the system classifies incoming signals in real time. Common approaches to pattern recognition with EMG use standard classification methods such as support vector machines (SVM), linear discriminant analysis (LDA), and linear regression to make live classification decisions at intervals around 250 ms (Smith et al., 2011).

Pattern recognition certainly offers tremendous improvement over TPC, and has just recently become commercially available through the Coapt system (Coapt LLC) for use with prosthetics. However, there are numerous issues with this approach that have yet to be fully addressed, some of which have to do with the nature of the algorithms, while others are limitations of sEMG. As was mentioned, there are numerous causes of signal degradation when using sEMG. Many, of these issues present problems for classification algorithms. Specifically, the most common approaches to pattern recognition use a combination of time domain features and autoregressive features (E. Scheme and Englehart, 2011). Unfortunately, either these features or the pattern recognition algorithms themselves do not generalize well to states that have never been seen before. An example of this, which will be discussed in more detail in Chapter 3, is that when a classifier is trained in unloaded conditions to recognize a hand movement, that is, the muscles are not supporting any weight, the classifier has incredible difficulty making accurate classifications when the user makes the same motions while supporting weight.

Additionally, the typical algorithms used (e.g., LDA, SVM, linear regression) are *offline* algorithms, which means the collection of samples is separated from the learning from those samples, i.e., a system first collects a set of labeled data samples and then learns from them before being deployed on live data. This is in contrast to *online* learning algorithms which learn from each data sample as it arrives. Offline methods do not allow for continual adaptation, while online ones do. Practically, this means that throughout the day, the classifier will lose accuracy as muscles fatigue, sockets shift, and perspiration increases. Users must then retrain their system as it loses accuracy. We should expect many of these specific issues to disappear with the use of IMES.

Also, typical classifiers, including the Coapt system, only predict a single movement class at a given instant, which makes movement somewhat unnatural. However, several approaches are being researched in order to provide multi-class predictions (L. J. Hargrove, E. J. Scheme, et al., 2010; Young et al., 2012; Yatsenko et al., 2007; Ameri et al., 2013; Castellini and Nowak, 2014). Another issue, is that straight classification methods do not provide a measure of how fast or how strong a motion class should be performed. Again, a number of approaches have also sought to address this (Castellini and Nowak, 2014; Amsuss et al., 2014; Yatsenko et al., 2007).

All of these issues aside, the use of pattern recognition is a promising approach. However, this approach only works if there are enough muscle sites that can contract to provide a pattern for detection in the first place. For many transradial and hand amputees, this is a viable option. For transhumeral and shoulder level amputations the number of available signals is significantly reduced. This can be addressed through the use of targeted muscle reinnervation (TMR) as described in the next section.

2.2.4 Targeted Muscle Reinnervation

Targeted muscle reinnervation (TMR) is a surgical technique where nerves that would have gone to the amputated limb are transplanted to new host muscle, possibly in the residual limb, or the chest (Dumanian et al., 2009; Hebert et al., 2014). These transplanted nerves are then able to cause contraction in the new host tissue, providing access, through sEMG, to additional control signals, which can then be used to control additional prosthetic DOC or to provide the necessary information for a pattern recognition based controller.

Further, a related technique, known as *targeted sensory reinnervation* (TSR) transplants the severed sensory nerves, which would have gone to the intact limb, into new subcutaneous host tissue (Hebert et al., 2014). This creates a pathway for providing sensory feedback to an amputee. For example, by pressing against a reinnervated sensory nerve on the chest an amputee may experience the sensation of having their thumb pressed.

Taken together, TMR and TSR provide an enhanced bi-directional pathway for interfacing an amputee with a prosthetic limb.

2.3 Reinforcement Learning and General Value Functions

Reinforcement learning (RL) can be described as an agent learning how best to behave by interacting with its environment, where *best* is defined to mean the maximization of some reward over time. An *agent* is defined as an entity which learns and takes actions, and the environment as anything outside of that agent (in truth, this boundary is fuzzy). The

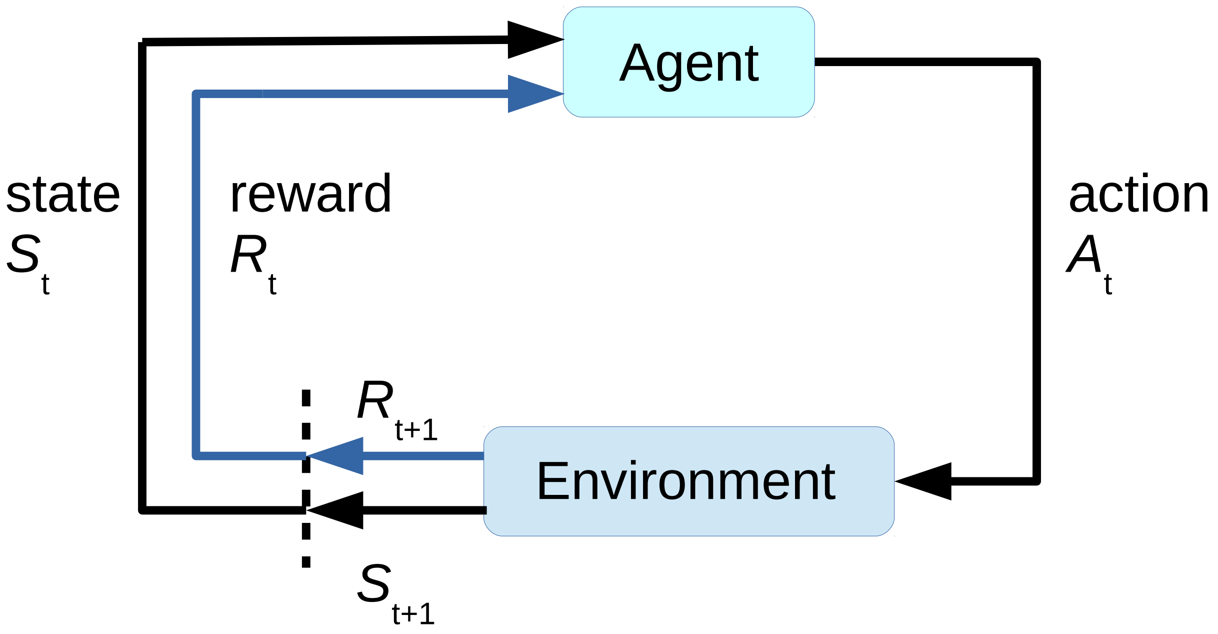


Figure 2.2: Textbook Reinforcement Learning diagram. Recreated from R. S. Sutton and Barto (1998).

textbook representation for RL is show in Figure 2.2. The agent receives information about the *state* of the environment, S, at time t. State is the condition of the agent's universe. In real-world settings the state is effectively infinite and only a subset of states is available or perceivable to an agent. Based on S_t, the agent takes action A_t, which results in a new state, S_{t+1} and a reward signal R_{t+1}. The term *policy* is used to describe how an agent decides which action to take in each state. A policy is a deterministic or stochastic mapping from a state to an action. The agent's job is to learn the best policy so as to maximize its reward in the long term.

2.3.1 Making Predictions

One of the key components in the RL problem is the estimation of value. That is, in order for the agent to improve its policy it must have some way to estimate the value of a particular action from a given state. Typically, we look to estimate the sum of discounted future rewards as shown in (2.1), where $0 \leq \gamma \leq 1$ is a discounting factor, allowing temporal emphasis to be shifted between immediate rewards ($\gamma = 0$) and rewards in the infinite future ($\gamma = 1$).

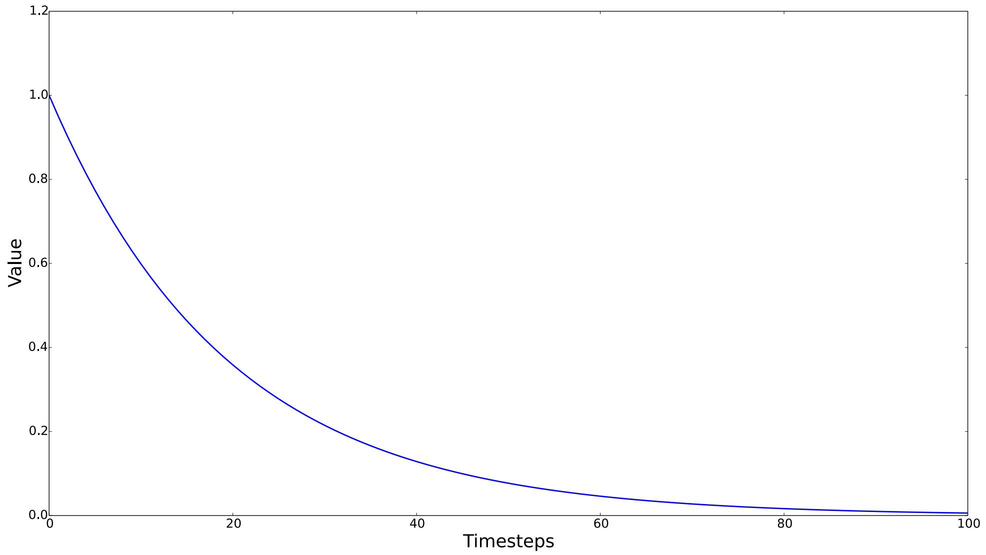


Figure 2.3: Discounted Return

This can be viewed as applying a filter over future rewards as shown in Figure 2.3.

$$G_t = \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i} \tag{2.1}$$

R. S. Sutton (1988) defined a class of algorithms known as *temporal difference* (TD) algorithms. These were a significant advancement to the field of prediction as they allowed predictions to be incrementally updated in an online setting, using *function approximation* and *bootstrapping*.

Function approximation is the technique of representing a complex or unknown function using an known, and often simpler, function. Function approximation is a powerful technique in machine learning for representing state. It allows an agent to compress a large or even infinite number of states into a finite set of parameters. While compressing state in this way can result in the loss of information, it is often the only way to make problems tractable and computationally efficient. Further, function approximation has the advantage of creating generalization across states, which can accelerate learning. The use of function approximation stands in contrast to tabular methods where each state is uniquely represented in some form of a lookup table. Such tabular methods are not used in this thesis, but are discussed in detail in R. S. Sutton and Barto (1998).

While the term bootstrapping has many meanings across different fields, it is used here in the same sense as in dynamic programming, that is, we leverage existing predictions to update other predictions. Effectively, this means that updates to predictions can be made before the predicted outcome has been seen. Bootstrapping is an efficient way to update predictions, however, it can have high *bias* — errors from inaccurate estimates. On the opposite end of the spectrum, Monte Carlo updates are only performed at the end of an episode, once the final state has been reached. Monte Carlo methods can have high variance in stochastic domains — they are sensitive to variations in the return. The TD(λ) algorithm, shown in (2.2), (2.3), and (2.4), uses the parameter $0 \leq \lambda \leq 1$ to allow us to take an intermediate approach lying somewhere between full bootstrapping ($\lambda = 0$) and Monte Carlo ($\lambda = 1$) updates. The result is that we are able to balance the trade off between variance and bias.

The TD algorithms are conceptualized and derived using the concept of forward and backward views through time. The forward view asks "what should the incremental updates for this state's value prediction be if we could see all the way into the future." Obviously, this is not possible to implement, given that one cannot know the future. The backward view looks back in time and says what the estimate updates should have been at each timestep given the actual observed samples. For algorithmic purposes, it is the backward view that is implemented and made possible by the inclusion of traces, vector $\boldsymbol{e} \in \mathbb{R}^{\mathrm{n}}$, such as the accumulating traces equation shown (2.3), or replacing traces, often used with binary feature vectors, as shown in (2.5).

$$\delta_t = R_{t+1} + \gamma \boldsymbol{w}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{w}_t^\top \boldsymbol{\phi}_t \tag{2.2}$$

$$\boldsymbol{e}_t = \lambda\gamma \boldsymbol{e}_{t-1} + \alpha \boldsymbol{\phi}_t \tag{2.3}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \delta_t \boldsymbol{e}_t \tag{2.4}$$

$$\boldsymbol{e}_t(s) = \begin{cases} \lambda\gamma\boldsymbol{e}_{t-1}(s) & \text{if } s \neq S_t \\ 1 & \text{if } s = S_t \end{cases} \tag{2.5}$$

The equations shown have thus far used a linear function approximator for value estimation. The goal of this algorithm is to learn the weight vector $\boldsymbol{w} \in \mathbb{R}^{\mathrm{n}}$ such that a value can be estimated for a given state S using the feature vector $\boldsymbol{\phi}(S) \in \mathbb{R}^{\mathrm{n}}$ via a simple inner product as shown in (2.6) (the transpose operator is denoted as $^\top$). Linear approximators are not the only option, however, they are extremely computationally efficient and I will solely use linear approximators throughout this thesis.

$$V(S) = \boldsymbol{w}^\top\boldsymbol{\phi}(S) \tag{2.6}$$

The term δ in (2.2) is known as the temporal-difference error or TD error. It describes the difference between the target we are trying to predict (the bootstrap estimated discounted sum of future rewards, $R_{t+1} + \gamma\boldsymbol{w}_t^\top\boldsymbol{\phi}_{t+1}$) and our current prediction $\boldsymbol{w}_t^\top\boldsymbol{\phi}_t$. If the algorithm was making perfect predictions, the TD error would be zero.

The weight update in (2.4) is derived from the usual gradient descent algorithm. As such, the term α, which appears in the traces equations, is in fact the usual step size parameter used in gradient descent and 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.

2.3.2 True Online TD(λ)

The forward view of the original TD(λ) algorithm makes the assumption that state estimates remain constant through to the end of an episode. That is, the forward view is derived for the offline case. Despite this, TD(λ) has been practically effective over the years, even in the online case, given that a good choice of parameters, α, and λ, are made. However, recent work by van Seijen and R. S. Sutton (2014) has derived a new temporal difference algorithm, known as True Online TD(λ), which does not make the same assumption about static state estimates, but rather accounts for the fact that the value estimates may change at every timestep. This is a significant advancement as it means that, for both the offline and online case, the forward and backward views are equivalent. In practice what has been shown is that True Online TD(λ) can produce better results than TD(λ), with lower error, faster learning rates, and stability over larger ranges of parameters α, and λ. Additionally, True Online TD(λ) eliminates the need to select between accumulating (see (2.3)) and replacing traces (see (2.5)).

Algorithm 1 True Online TD(λ)

```
initialize $\boldsymbol{w}$ arbitrarily
**loop** {over episodes}
  initialize $\boldsymbol{e} = \mathbf{0}$
  initialize $S$
  $\hat{v}_{S_{old}} \leftarrow \boldsymbol{w}^\top \boldsymbol{\phi}(S)$
  **repeat** {for each step in the episode}
    generate reward $R$ and next state $S'$ for $S$
    update $\gamma$ if needed *(if $S'$ is terminal: $\gamma \leftarrow 0$)*
    $\Delta\hat{v}_S \leftarrow \boldsymbol{w}^\top \boldsymbol{\phi}(S) - \hat{v}_{S_{old}}$
    $\hat{v}_{S_{old}} \leftarrow \boldsymbol{w}^\top \boldsymbol{\phi}(S')$   *($S'$ is used here, because $S'$ becomes $S$ in the next iteration)*
    $\delta \leftarrow R + \gamma \boldsymbol{w}^\top \boldsymbol{\phi}(S') - \boldsymbol{w}^\top \boldsymbol{\phi}(S)$
    $\boldsymbol{e} \leftarrow \boldsymbol{e} + \alpha\big[1 - \boldsymbol{e}^\top \boldsymbol{\phi}(S)\big]\, \boldsymbol{\phi}(S)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \delta \boldsymbol{e} + \Delta\hat{v}_S(\boldsymbol{e} - \alpha\boldsymbol{\phi}(S))$
    $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e}$
    $S \leftarrow S'$
  **until** $S$ is terminal
**end loop**
```

The True Online TD(λ) algorithm is shown in Algorithm 1. The key differences are in the calculation of traces, (2.7) and (2.9), and weight updates, (2.8).

$$\hat{e}_t = \boldsymbol{e}_{t-1} + \alpha \left[1 - \boldsymbol{e}_{t-1}{}^\top \boldsymbol{\phi}_t\right] \boldsymbol{\phi}_t \tag{2.7}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \delta_t \hat{e}_t + (\boldsymbol{w}_t - \boldsymbol{w}_{t-1})^\top \boldsymbol{\phi}_t (\hat{e}_t - \alpha \boldsymbol{\phi}_t) \tag{2.8}$$

$$\boldsymbol{e}_t = \gamma \lambda \hat{e}_t \tag{2.9}$$

2.3.3 General Value Functions

The TD algorithms are typically associated with estimating value, specifically, the discounted sum of reward, R. However, they can also be used in a broader sense to predict any measurable signal. This is done by replacing R, in the TD error equation, with the signal of interest (R. S. Sutton, 1988). Let us refer to this term as the *cumulant*, of which reward is a specific instance (A. White, 2015). When the algorithms are used in this way they are called general value functions (GVFs). GVFs allow us to represent knowledge about an agent, its environment and the interaction between the two using temporally extended predictions (R. S. Sutton, Modayil, et al., 2011).

In GVFs, knowledge is represented as the answer to a computationally specified question about a policy, or way of behaving. The GVF question is programmatically described by (2.2) and parameterized by three functions: the cumulant (R), γ, and the policy. For example, with a GVF one can ask, "How much current will the left motor draw over the next 2 s if the robot drives straight?" One way of representing the answer is through standard TD value functions. The answer is then approximated by the value function with the parameters λ and the weight vector $\boldsymbol{w}$.

In order to learn as much about the world in as short a time as possible an agent must be able to learn many things in parallel. *Off-policy* learning plays this role, allowing learning about one policy, the target policy, while following another policy, the behavior policy. In the simplest setting the behavior policy and the target policy are the same. This is referred to as *on-policy* learning. While off-policy learning is of significant interest for developing real world robots, it is not used directly in this thesis and will not be explained further in

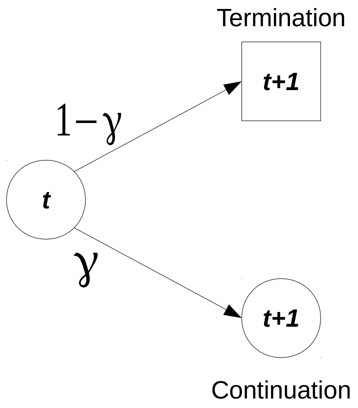


Figure 2.4: Expected timesteps to termination

this introduction. For a more detailed discussion of *off-policy* learning with GVFs see R. S. Sutton, Modayil, et al. (2011), Modayil, A. White, and R. S. Sutton (2014) and A. White (2015).

GVFs allow us to make temporally extended predictions. That is, they allow us to make a prediction for some timescale in the future. The γ term determines how far into the future the prediction is made, with $\gamma = 0$ looking only one step ahead and $\gamma = 1$ looking to infinity. The γ function can be thought of as the probability of continuing past the current state, and $1 - \gamma$ is therefore the probability of terminating at the current state. With this in mind the expected number of timesteps until termination, $\tau \equiv E[\#\text{ timesteps}]$, can be derived by considering the trajectory shown in Figure 2.4.

To calculate the expected number of timesteps, τ, simply consider the length of the trajectory along each path and the probability of taking each path (This is nothing more than the standard calculation of expectation $E[X] = \sum_i x_i p_i$, where p is probability).

$$\begin{aligned} \tau = \text{Length of Terminating Trajectory} \cdot \text{Probability of Terminating} + \\ \text{Length of Continuing Trajectory} \cdot \text{Probability of Continuing} \end{aligned} \tag{2.10}$$

$$\begin{aligned}
\tau =& 1(1-\gamma) + (\tau+1)\gamma && (2.11)\\
\tau =& 1 - \gamma + \tau\gamma + \gamma && (2.12)\\
\tau =& 1 + \tau\gamma && (2.13)\\
\tau(1-\gamma) =& 1 && (2.14)\\
\tau =& \frac{1}{1-\gamma} && (2.15)
\end{aligned}$$

Thus, we can say that in expectation γ is related to the number of prediction timesteps as

$$\gamma = 1 - \frac{1}{\tau}. \tag{2.16}$$

Both γ and the cumulant can be state dependent, which allows an agent to ask more complicated questions. For example, an agent could ask, “How long until I collide with a wall if I go straight?” To ask this question set $\gamma = 1$ and $R = 1$ for all states in which the robot has not collided with the wall and $\gamma = 0$ and $R = 0$ when the robot does collide with the wall (R. S. Sutton, Modayil, et al., 2011). While state based γ and cumulant functions are very powerful, they are not used in this thesis and are therefore not described further.

The experiments in Chapters 4 and 5 use fixed γ values. Using a fixed γ has several implications. First, the value computed in (2.6) is an estimate of the multiplication of Figure 2.3 and the target signal over the future timesteps. This assumes that the value at some time τ is related to the values before and after. Essentially, everything in the full trajectory of the target signal is compressed down to a single number to give us a prediction for a specific point in time, in expectation. As one would expect, the prediction is not unique and does not capture the behavior of a signal in time. For example, Figure 2.5 shows a signal which is always 1.0 except around the 20 timestep mark, which happens to be where we want to predict. If our predictor were exact it would predict a value of 0.0. However, our GVF based predictor gives a value that is nearly 1.0 as all the other values around this

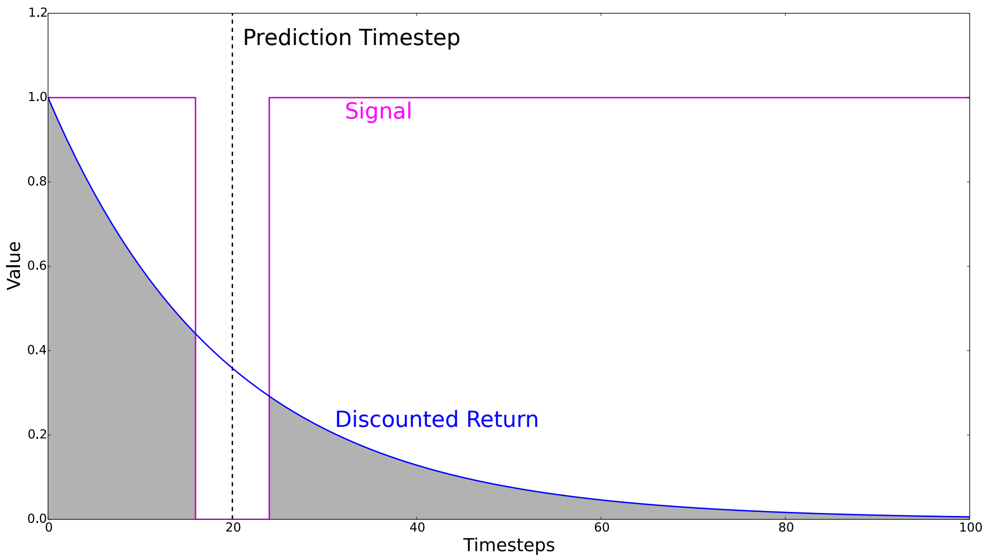


Figure 2.5: Notch signal

notch weights the signal towards 1.0. For the purposes of this thesis this is a benefit in that it generalizes and smooths out predictions. In other scenarios where an exact prediction is required this would not be desirable.

In Figure 2.6 we see two different signals, shown in blue and red. Again, we want to make a prediction for 20 timesteps in the future. Signal 1 is a small, brief signal that occurs very soon, while Signal 2 is a large, long signal which occurs much later. However, for both of these signals our GVF predictor would give similar values. Again, in some situations this sort of generalization is desirable, while in others it may be detrimental.

The equations described thus far have predicted the *cumulative* discounted sum of a signal; *instantaneous* values of signals can be predicted by prefixing the cumulant with $(1-\gamma)$. This is simply a matter of dividing the target signal by the expected number of timesteps as given in (2.15). In this way the GVF predicts the cumulative sum of $\frac{R}{\tau}$, or a per-timestep measure of the signal, which, when summed, produces an estimate of the instantaneous signal. This scaling can also be applied to the output prediction of the GVF itself, rather than the target signal directly.

Further, if the target signal is a binary event and this signal is scaled as described in the

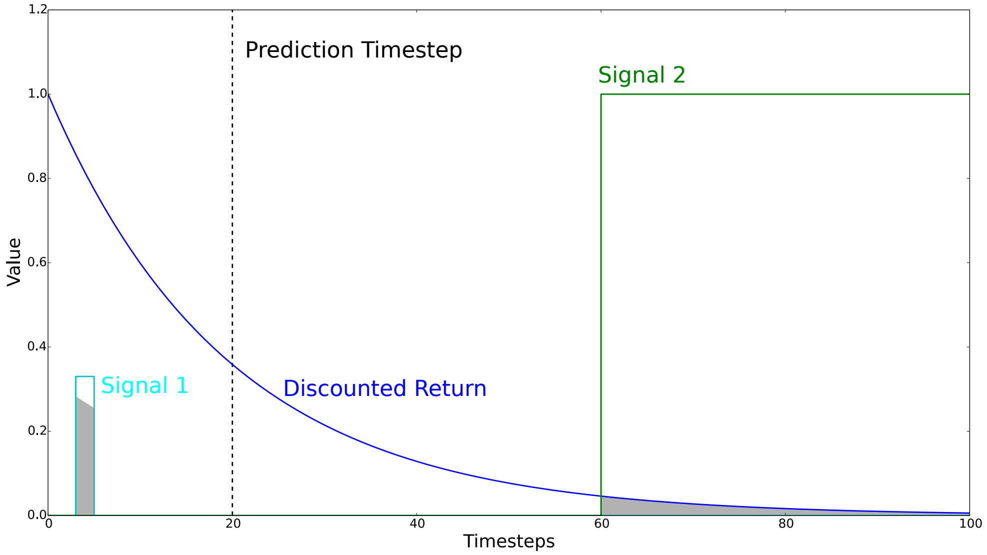


Figure 2.6: Dual signal

previous paragraph, then prediction of the GVF can be thought of as giving the psuedo-probability of the event occurring over some time scale, e.g., “If the robot drives forward, what is the probability of colliding with the wall in the next 3 s?”

The use of GVFs is fairly new but already several promising applications have been demonstrated with robots. Several authors have demonstrated that GVFs could be used to efficiently learn to predict large numbers of sensorimotor signals, simultaneously, in real time on a mobile robot (R. S. Sutton, Modayil, et al., 2011; Modayil, A. White, Pilarski, et al., 2012). Additionally, GVFs have been used in order to predict various signals used in prosthetic arms, such as joint positions, user EMG signals, joint movement and joint selection (Pilarski, Dawson, Degris, Carey, Chan, et al., 2013; Edwards et al., 2014).

How such predictions should be used in control is an open research question, with some recent investigations. Modayil and R. S. Sutton (2014) have used prediction to control a simple mobile robot, such that when a prediction exceeds a threshold the robot will activate and follow a fixed, hand-coded behavior, producing Pavlovian-like responses. Specifically, when their mobile robot predicted a future over-current condition it would shut off its motors. This approach is similar to many prediction-based reflexive reactions found in humans and other animals (Redish, 2013; Linden, 2003). Additionally, Pilarski, Dick, et al. (2013), performed a comparative study of various techniques by which GVF based predictions could be used in order to generate target commands for the control of a prosthetic arm joint. This study is the basis for the work in Chapter 5 and will be discussed later in further detail.

2.3.4 Tile Coding

Tile coding is a common, non-linear approach used for creating sparse binary features from real-valued signals and is well suited to the online learning of RL. Tile coding forms a tiling by taking a sensor space and partitioning it into non-overlapping regions called tiles as shown in Figure 2.7. The left side of the figure shows a 2 dimensional tiling for a hypothetical sensor space, which has been normalized and scaled between [0,1] and partitioned into uniform widths of 0.5. Tilings need not be square in shape and need not have the same resolution across all sensors as is shown. Further, they can be used with any number of dimensions. For any given vector in sensor space there will be exactly one active tile with the value 1 with all other tiles set to 0. Adding offset tilings, as shown in the right side of the figure, increases resolution, but unlike simply using a single tiling with higher resolution, this approach allows broad generalizations to be rapidly learned, i.e., data points are not isolated from one another and a learner is able to apply what was learned in nearby regions. It should be noted that tile coding can also be used with hashing, which is an effective way to limit the resulting feature size and memory footprint. For further discussion on tilecoding refer to R. S. Sutton and Barto (1998). Tile coding is used in Chapters 4 and 5 in order to represent various signals as features for the learning agents.

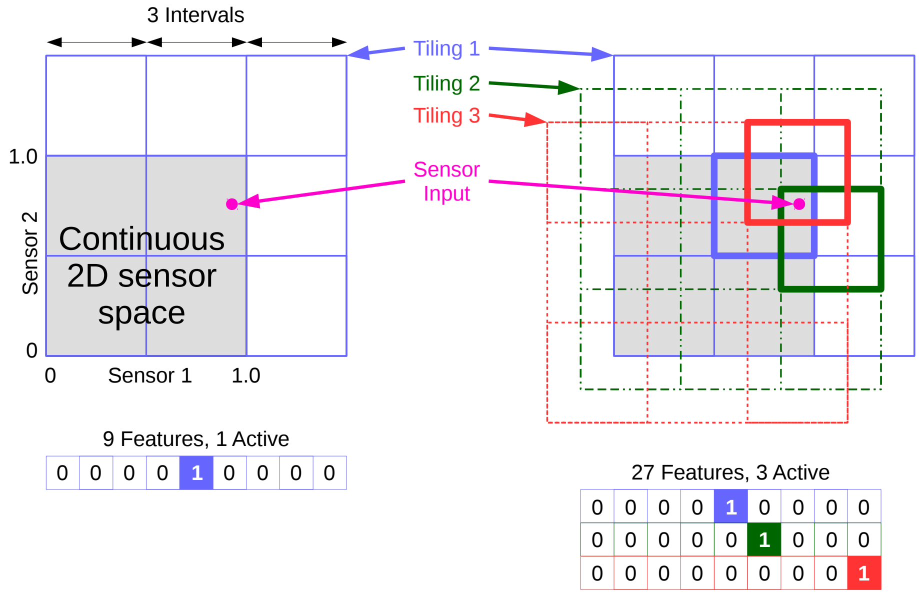


Figure 2.7: Tile coding - *left*) A normalized continuous 2D sensor space is tiled using 3 tiles per dimension with a bin width of 0.5. *right*) Three offset tilings are made over the same sensor space, producing 27 features with 3 active at any given instant.

Chapter 3

Classification of Embedded EMG Signals Amidst Concurrent Contraction[1]

Pattern recognition is currently considered the state of the art for EMG based control of prosthetics. However, despite the significant potential for improving control, this approach has limitations. Here a particular scenario is identified under which the standard pattern recognition features and classifiers does not generalize well. Further, these findings highlight current problems in the way that pattern recognition methods are evaluated in the prosthetics domain.

This chapter is arguably the smallest contribution to this thesis. This chapter relies on offline methods, which are not adaptive. However, in Section 6.2 I propose one way in which offline pattern recognition methods might be combined with the online learning method of general value functions to produce an adaptive system. While this chapter focuses on the use of offline methods, it does show the limitations of such an approach and further motivates the work in Chapters 4 and 5.

[1] This chapter is based on a project submitted for course credit in CMPUT 551 during my MSc in Computing Science. It was submitted as *Sherstan, C., Ramirez, O., Ahmad, Z. F., and Zhang, H., "Classification of Embedded EMG Signals Amidst Concurrent Contractions"*. I have since reevaluated the work and rerun the experiments for inclusion in this thesis. While some of the text and figures of this chapter are taken from that original report, it has largely been rewritten and incorporates updated results and additional analysis.

Further, one of the key functions of an *intelligent* arm is the ability to understand the user and their environment. This is a recurring theme throughout this thesis and is explored in more detail in the following chapters using general value functions. Pattern recognition is one alternative approach to making predictions about a user's intentions and therefore serves as a comparative and likely compatible methodology.

3.1 Overview

Pattern recognition of EMG signals offers to significantly improve the control of prosthetic limbs, particularly prosthetic arms, by allowing amputees to move beyond controlling a single degree of freedom at a time. Additionally, the control provided is more natural for the amputee, e.g., when the amputee thinks to open or close their hand, the nerves that would have performed those activities in the intact limb fire in a specific pattern which can be directly translated to opening and closing the prosthetic hand. This approach is particularly useful when muscles are still available and innervated, as may be the case for transradial, hand, and partial-hand amputees, providing a wealth of natural muscle signals. A surgical technique, known as targeted muscle reinnervation (TMR) may extend these benefits to other amputees as well, by providing access to nerve signals that would have controlled the intact limb. Here we show that the most common methods of pattern recognition with EMG, specifically time-domain and autoregressive features (TD/AR) with a linear discriminant analysis classifier (LDA), do not generalize well to the case where host muscle is undergoing concurrent contraction when trained with only unloaded movement samples, causing accuracy levels to drop well below usable levels. An example of this might be when an amputee is holding a heavy object. This is demonstrated using a single able-bodied subject in an experimental setting meant to replicate the concurrent contractions mentioned. These results indicate that additional approaches may be required in order for classification systems to be useful in real-world scenarios.

It should be noted that the acronym TD, used in this chapter, refers to *time domain* features as opposed to *temporal difference* algorithms as in the rest of this thesis. Unfortunately, this is the accepted acronym used to refer to each in their respective fields

3.2 Introduction

The use of pattern recognition with EMG signals has been explored since the late 60s and early 70s (Finley and Wirta, 1967; Lyman et al., 1976; Lawrence et al., 1972). The basic premise involves recording EMG samples while a user makes prescribed motions. The samples are labeled with the corresponding movement class and converted to some representation or features, which are then used to train a mapping from a set of features to a corresponding class using a supervised machine-learning classification algorithm. After training, the system then records live signals from the user, converts them to features, and passes the features through the mapping to get the predicted movement class, which is then converted to the corresponding motion of the prosthetic arm. Significant improvements in algorithms and in computational power now make such approaches potentially clinically viable. In fact, the past year, 2014, saw the release of the first commercial pattern recognition system for use in controlling a prosthetic limb via EMG by Coapt LLC.

Pattern recognition offers significant benefit over traditional proportional control methods in that they give the the user greater access to the functions of their prosthetic. While most classification approaches are limited to classes of motions which can be held statically, those classes might include single DOF activations or synergies of DOF, thereby allowing control of more than one joint at a time.

Although typical classification approaches produce only a class label, at least one study has looked at combining classification with proportional control to allow the user to also control the speed with which motions are performed (E. Scheme, Lock, et al., 2013).

One of the key benefits to pattern classification is that it may be possible, depending on each amputee of course, that prosthetic control can be performed intuitively, e.g., the user thinks to open their hand and the hand opens. This is because the amputee's thoughts are translated directly to patterns of muscle contractions. These patterns of contractions, whatever they might be, are learned by the classifier. While it is reasonable to expect that users adapt to the system, pattern recognition methods bring control closer to what a user might be used to, thereby lowering the barrier to use.

Pattern recognition assumes that an amputee has sufficient muscle signals to record from, which is a realistic scenario in transradial, hand or partial-hand amputees, where the muscles used for hand and wrist control are often still intact and functional. For amputees lacking the muscle signals needed for pattern recognition there is a surgical method, known as targeted muscle reinnervation (TMR), which can make those signals available. TMR transplants the nerves that would have gone to control the hand and wrist into new host muscle such as the biceps, triceps, and pectorals. The new host muscle then responds to these nerve signals by contracting, providing a way to record the nerve signals through EMG, and thus making pattern recognition a viable option for higher-level amputees.

3.2.1 The Problem of Generalization

A common problem faced by all forms of machine learning is the problem of generalization. That is, does the knowledge learned from training transfer well across samples that have never been seen? This is the problem we explored in these experiments – do the common approaches used for classification of EMG signals generalize well.

Often, in EMG pattern recognition studies, the experiments only consider classification accuracy for movements made in unloaded conditions where the subject is relaxed, sitting down and not holding anything in their hand (Hudgins et al., 1993; L. J. Hargrove, Englehart, et al., 2007), or, in the case of an amputee, not wearing the prosthesis (Zardoshti-Kermani et al., 1995; Boostani and Moradi, 2003). One can imagine that host muscles used in the TMR surgery may still be used for regular functions. For example, imagine an upper-limb amputee opening a door by its handle. The prosthesis is receiving signals to grasp the handle and turn. Additionally, the non-reinnervated pectoral muscle is contracting to push the door open. The control system needs to distinguish the activation pattern used to control gripping and turning amidst the background contraction of the host pectoral. This is what we refer to as concurrent contraction. Here we look at whether or not a classifier trained in unloaded

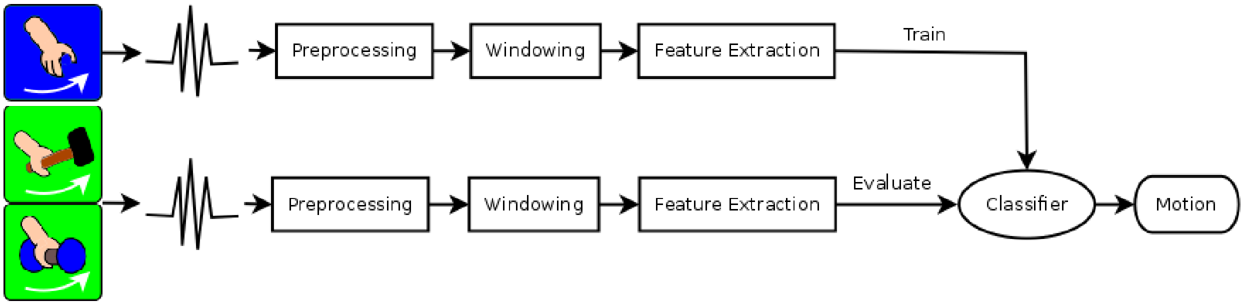


Figure 3.1: Classification System - Training samples are recorded from movements made while the user is empty handed (blue). Evaluation is then performed on motions recorded with the user holding different objects (green).

conditions generalizes to the case where the muscles are undergoing concurrent contraction.

An ideal control system would able to accurately classify signals amidst concurrent contractions having been trained only in unloaded conditions but still able to classify accurately under loaded ones (See Figure 3.1). This is desirable as it is not feasible to have a user train the system on all possible loads. A usable system should have a target accuracy above 90% (E. Scheme and Englehart, 2011) and a response time under 200 ms (Smith et al., 2011).

Many features and many classifiers have been evaluated over the years (E. Scheme and Englehart, 2011). The most common choice, at this time, is the use of time domain (TD) and autoregressive (AR) features with a linear discriminant analysis (LDA) classifier, similar to what is used by the approach taken by the Coapt system already mentioned.

Here we evaluate the generalization of classification accuracy using several common features and classifiers including the TD/AR features with an LDA classifier. Our results indicate that these approaches do not generalize well in the case of concurrent contractions.

Related Studies

Cipriani et al. performed a similar study looking at the effects of motion and loading on classification accuracy caused by the muscular contraction needed to stabilize the prosthetic itself and showed that there is significant degradation of classification accuracy (Cipriani, Sassu, et al., 2011). However, they only looked at a single classifier, K-nearest neighbors

(KNN), and a single feature, mean absolute value. Unfortunately, this is a limited evaluation and the choice of classifier and feature are arguably not the most common or likely to generalize. We would expect the mean absolute value to vary widely between various states of prosthetic stabilization and concurrent contraction.

Another study was performed by Tkach et al. in which they examined the robustness of TD features with an LDA classifier to three types of perturbations: electrode shift, variability of contraction effort, and fatigue (Tkach et al., 2010). They found that there was significant reduction of classification accuracy caused by these perturbations, with fatigue having the smallest effect. However, it is debatable whether or not their method of inducing fatigue was a good analogue for what might be seen in the real world. While the studies look at similar perturbations, they are not identical. Specifically, in our study we expect more bias to be present in one or two channels due to having to hold objects against gravity. Additionally, our study looks at a wider range of classifiers, and also introduces a novel feature, which attempts to capture the relationship between the channels.

3.3 Methods

3.3.1 Signal Processing

A common step in processing incoming EMG signals is to use a sliding window like the one shown in Figure 3.2; incoming samples are buffered up to the size of the window and then that window is passed through the classification system (E. Scheme and Englehart, 2011). The window is then shifted over slightly and new incoming samples are appended to the buffer. In order for EMG to be used effectively as a means of control it must be both accurate and relatively fast. Two studies showed significantly different estimations of the maximum allowable delay for a usable controller, ranging from 50 ms (Childress and Weir, 2004) to 400 ms (Hefftner et al., 1988), but others have suggested a middle-of-the-road value with the optimal window length for pattern recognition control being between 150 and 250 ms (Smith et al., 2011). We therefore chose a data window of 150 ms and window shift

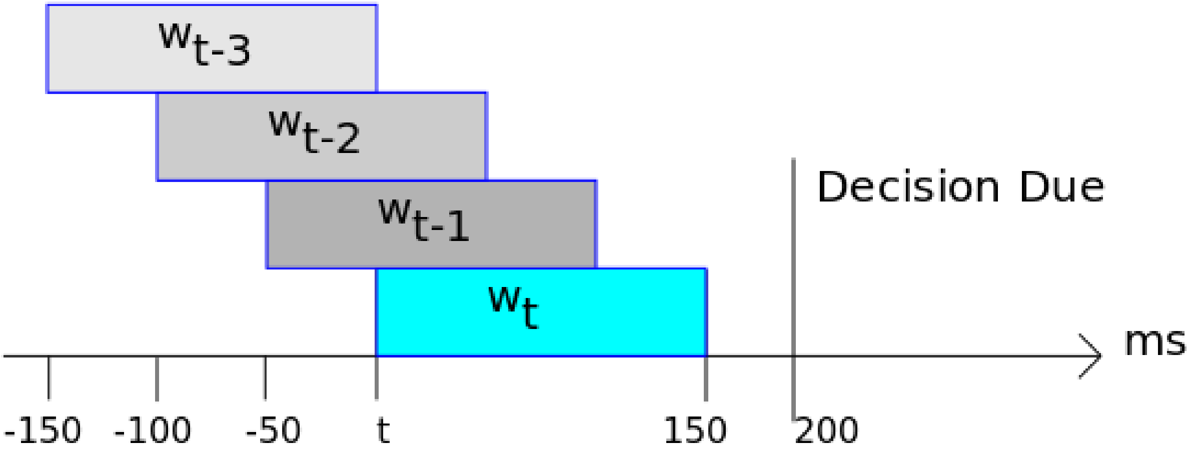


Figure 3.2: Windowing Data - sliding windows of 150 ms with an offset of 50 ms were used for classification

of 50 ms, resulting in a new decision every 50 ms based on a 150 ms window that had begun 200 ms before.

Features

Studies have indicated that, of the methods tested, feature selection is more important than classifier selection when classifying EMG signals (L. Hargrove et al., 2007). A common approach when creating features for a data-window is to split up the window into smaller segments. For some features this produces better classification results. The features considered in this experiment are listed below and include the common TD features as well as several other common features and a novel feature, *Channel Ratios*, introduced in this work.

1. *Time-domain (TD)* - TD features are possibly the most commonly used for EMG processing and are described by Hudgins et al. (1993). They are simple and fast to compute, producing high accuracy in many of the cases studied in the literature. These are described below for completeness:

   (a) *Mean Absolute Value* - The mean absolute value of the signal is taken on a segment. This feature is also referred to as the Integrated Absolute Value by Zardoshti-Kermani et al. (1995).

$$\bar{X_i} = \frac{1}{N}\sum_{k=1}^{N}|x_k| \quad \text{for } i = 1, \cdots, I \tag{3.1}$$

(b) *Mean Absolute Value Slope* - This feature takes the difference between adjacent segments.

$$\Delta\bar{X_i} = \bar{X}_{i+1} - \bar{X_i} \quad \text{for } i = 1, \cdots, I \tag{3.2}$$

(c) *Zero Crossings* - This provides a simple frequency measure by counting how many times the waveform crosses 0. A threshold is applied such that a zero crossing is only counted if the absolute difference between two consecutive samples exceeds the threshold.

(d) *Slope Sign Change* - This feature is believed to provide another measure of frequency. It counts how many times the slope of the waveform changes sign. Like *Zero Crossings*, the absolute difference between two consecutive samples must exceed a threshold to be counted.

(e) *Waveform Length* - This feature simply calculates the length of the waveform. It is intended that this feature provide information about the complexity of the waveform.

$$l_i = \sum_{k-1}^{N}|\Delta x_k| \tag{3.3}$$

2. *Variance* - Variance is a measure of a signal's power (Zardoshti-Kermani et al., 1995).

$$VAR = \frac{1}{N-1}\sum_{i=1}^{N}x_i^2 \tag{3.4}$$

3. *Histogram* - The EMG histogram was introduced by Zardoshti-Kermani et al. (1995), it breaks the absolute signal voltage range into bins and then counts the number of samples in a segment which fall into each bin.

4. *Autoregressive (AR) Coefficients* - In an AR model, a sample X_t is assumed to be

related to N previous samples by (3.5), where a_i are called the AR coefficients, which say how the current sample is related to previous samples. The ϵ term is a Gaussian noise term centered at the origin. AR coefficients have been shown to add slightly better accuracy when combined with TD features (L. J. Hargrove, Englehart, et al., 2007), at the cost of increased computational complexity. AR features were implemented using the NiTime library (*Nitime* 2015).

$$X_t = \sum_{i=1}^{N} a_i X_{t-i} + \epsilon_t \tag{3.5}$$

5. *Cepstral Coefficients* - can be interpreted as the rate of change in the different spectrum bands (E. Scheme and Englehart, 2011). Cepstral Coefficients, c_i, can be calculated from the AR Coefficients, a_i, as follows (Pattichis and Elia, 1999):

$$c_1 = - a_1 \tag{3.6}$$

$$c_n = - a_n - \sum_{k=1}^{n-1} (1 - k/n) a_k c_{n-k} \quad \text{for } 1 < n \leq p \tag{3.7}$$

$$c_n = - \sum_{k=1}^{n-1} (1 - k/n) a_k c_{n-k} \quad \text{for } n < p \tag{3.8}$$

6. *Channel Ratios* - Finally, we introduce a novel feature of our own, which is the ratio of the mean absolute value, $\bar{X}$, between unique pairs of channels:

$$Ratio = \frac{\bar{X}_{ch=i} - \bar{X}_{ch=j}}{\bar{X}_{ch=i} + \bar{X}_{ch=j}} \quad \text{where } i > j \tag{3.9}$$

While all other features tested looked only at single channels in isolation, *Channel Ratios* explicitly considers the relationship between pairs of channels.

Each feature vector for a window also included a bias feature. Parameter sweeps were performed for all the features implemented as per Table 3.1 using four classifiers: K-nearest neighbors (KNN), linear discriminant analysis LDA, Random Forests, and support vector

Table 3.1: Feature parameter sweeps

Feature	**Parameter**	**Evaluations**
Mean Absolute Value	Segments	1 to 10, step 1
Zero Cross	Segments Threshold	1 to 10, step 1 0.1 to 0.2 V, step 0.1
Slope Sign	Segments Threshold	1 to 10, step 1 0.1 to 0.2 V, step 0.1
Wavelength	Segments	1 to 10, step 1
Variance	Segments	1 to 10, step 1
Mean Absolute Value Slope	Segments	1 to 10, step 1
Histogram	Segments Bin Count	1 to 10, step 1 10 to 70, step 10
Ratios	Segments	1 to 10, step 1
Cepstral Coefficients	Segments Order	1 to 3, step 1 1 to 10, step 1
Autoregressive	Segments Order	1 to 10, step 1 1 to 10, step 1

machine SVM. The same set of parameters was used for all classifiers. However, the sweeps revealed that the same parameters did not necessarily perform equally well across all classifiers. Thus, it was necessary to choose features that compromised between all the classifiers. The first criteria used to select parameters was to find values which did reasonably well across all of the classifiers. The second criteria, all other things being equal, was to choose the simplest and smallest representation. Table 3.2 shows the parameters selected.

3.4 Experimentation

In order to simulate the conditions of TMR under unloaded and loaded conditions we measured the forearm EMG signals of a single able-bodied subject performing several hand motions under various degrees of loading including unloaded (empty-handed), holding 5 lb and 15 lb weights, and holding a hammer. Holding these objects allowed us to simulate concurrent contractions with different contraction levels and biases.

EMG data was collected using 4 bipolar surface electrodes placed equidistantly about the circumference of the subject's forearm approximately two-thirds up from the wrist. Previous

Table 3.2: Feature parameters

Feature	**Segment Count**	**Parameters**
Mean Absolute Value (MAV)	10	
Zero Cross	1	Threshold: 0.1 V
Slope Sign	1	Threshold: 0.1 V
Wavelength	5	
Variance	6	
Mean Absolute Value Slope	5	
Histogram	1	Bin Count: 70
Ratios	5	
Cepstral Coefficients	1	Order: 2
Autoregressive	1	Order: 2

work has shown that 4 electrodes in such a configuration produce similar levels of accuracy to placing three electrodes at optimal positions and minimal gains are seen by adding additional electrodes (L. J. Hargrove, Englehart, et al., 2007; E. Scheme and Englehart, 2011). The signals were recorded using a Delsys Bagnoli 8 EMG and read into the computer via a National Instruments USB-6216 digital acquisition unit (DAQ). Signals were recorded at 1 kHz and bandpass filtered between 20 and 450 Hz internally by the Delsys system.

The implementation for these experiments was done mostly in Python using the sci-kit learn library (Pedregosa et al., 2011) for the classifiers. The Robot Operating System (ROS) (Quigley et al., 2009) was used to facilitate data recording and analysis.

Resting and the six movement classes shown in Figure 3.3 were chosen for these experiments, giving seven different classes. While seated and with the arm in a relaxed, downward position, the subject was prompted to randomly perform one of the movement classes. Each motion was held for 3 s followed by a 3 s rest. Separate datasets were recorded where the subject held nothing (unloaded), 5 and 15 lb weights, and a hammer held by the end of its handle. Samples were labeled with the corresponding desired movement as they were recorded. Resting samples were labeled as such. Delays in the subject's response resulted in the need for labels to be aligned with signals by hand during post-processing.

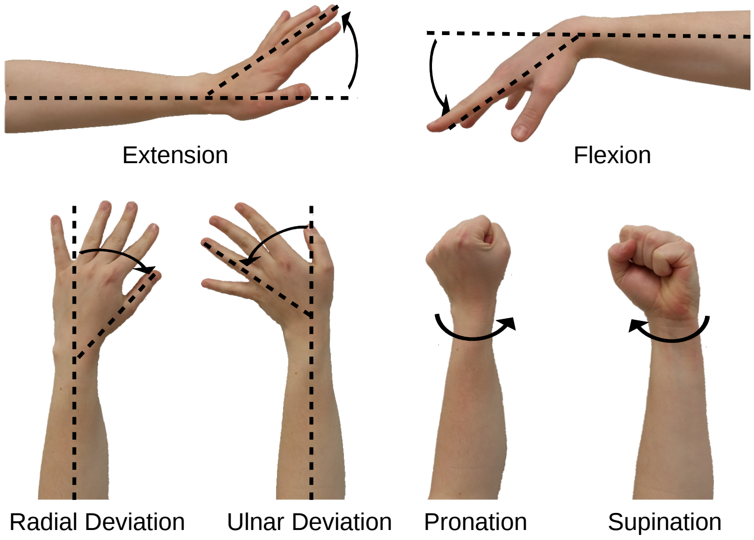


Figure 3.3: Different motions of the wrist

3.5 Results

The classification system was first evaluated against the unloaded dataset alone. Training was performed using 80% of the samples, selected randomly, with the other 20% used for testing. This splitting and testing was performed 10 times and the average accuracy was recorded and can be seen in Figure 3.4. We achieved high levels of accuracy using the standard TD features consistent with results reported by L. J. Hargrove, Englehart, et al. (2007). However, we did not observe equal accuracy across all classifiers for the TD set as they did, which could be because we evaluated different classifiers. It can be seen from this plot that different features can produce very different results depending on which classifier is used. These results even suggest that Random Forests might be a better choice of classifier than LDA.

To evaluate the performance during concurrent contraction the classifiers were first trained using the entire unloaded training set. Loaded datasets were then windowed and featurized and those features were passed through the pretrained classifiers. It can be clearly seen from Figure 3.5a that the most common methods of EMG pattern recognition do not generalize under concurrent contractions. While addition of the remaining features described

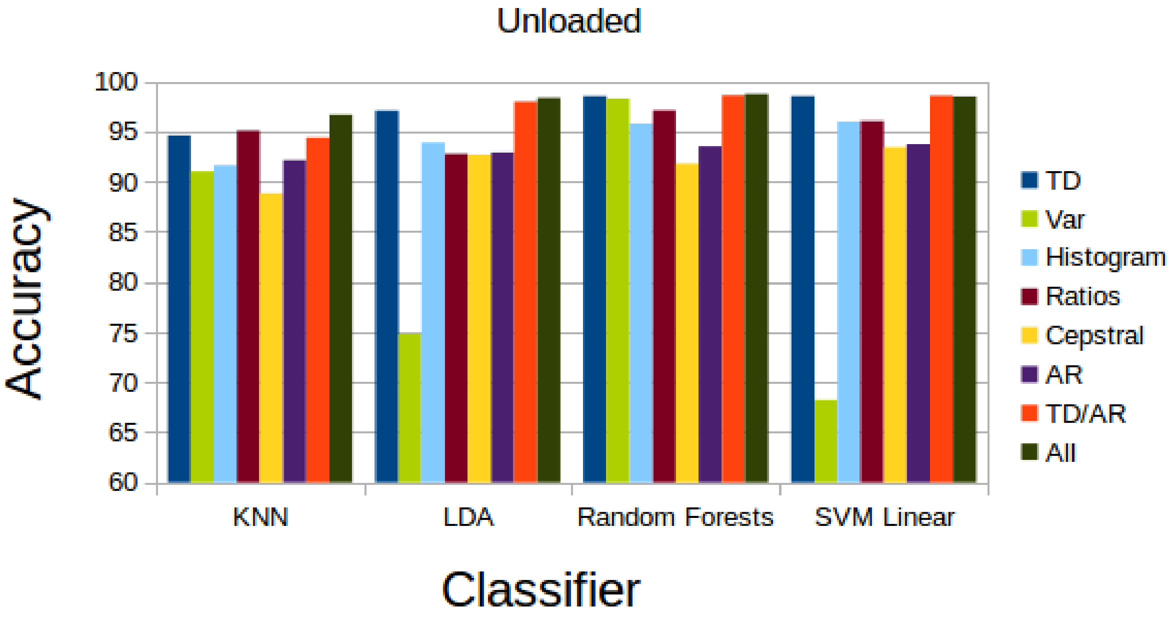


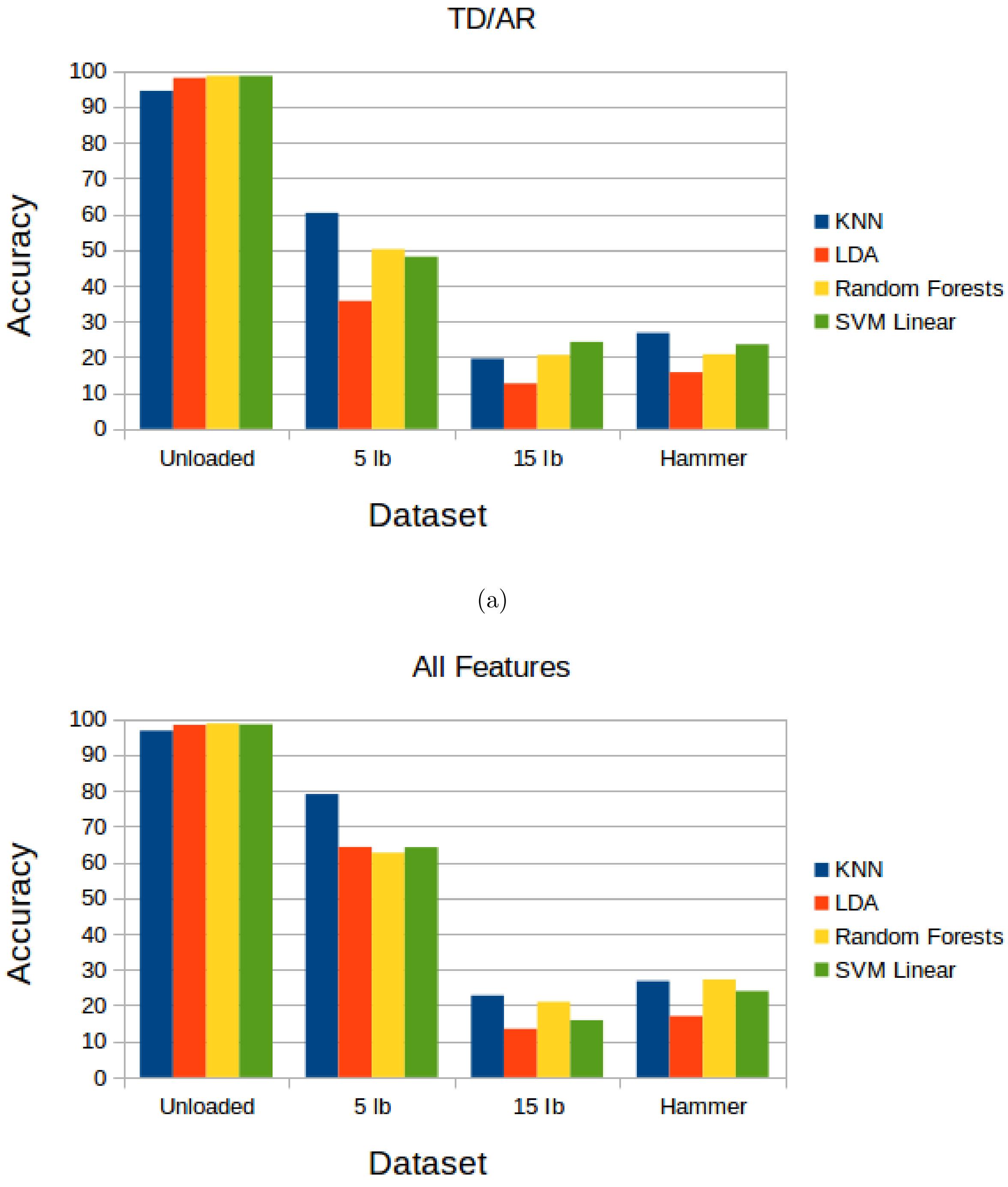
Figure 3.4: Classification accuracy of unloaded dataset. 80% of the unloaded samples were randomly chosen for use in training, with the additional 20% held out for testing.

in Section 3.3.1 showed slight improvement for the 5 lb dataset it is still well below the usable threshold (see Figure 3.5b). The other datasets did not see any improvement. It should be noted that given 7 classes, picking randomly across a uniform distribution should produce 14% accuracy. We can see that some of the features chosen perform even worse than random.

Figure 3.6 shows the performance of each individual feature across the various classifiers. What is clear from these figures is that classification drops off with increasing load or increasing bias. Additionally, we see that some features do better than others, but which features do better depends on the choice of classifier. In general though, we see that the novel *Channel Ratios* feature performs best or at least near the top.

3.6 Discussion and Future Work

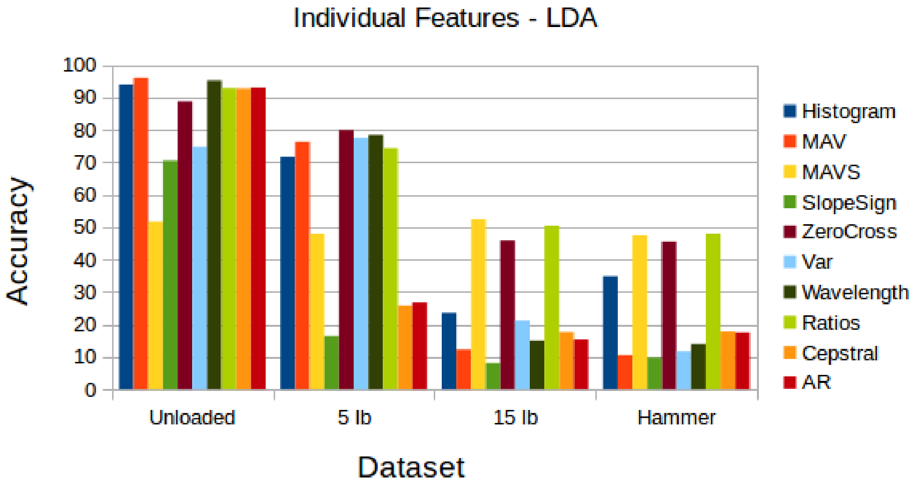
Hudgins et al. showed that the onset of contractions displayed "significant nonrandom components" (Hudgins et al., 1993). That is, when transitioning from rest to one of the



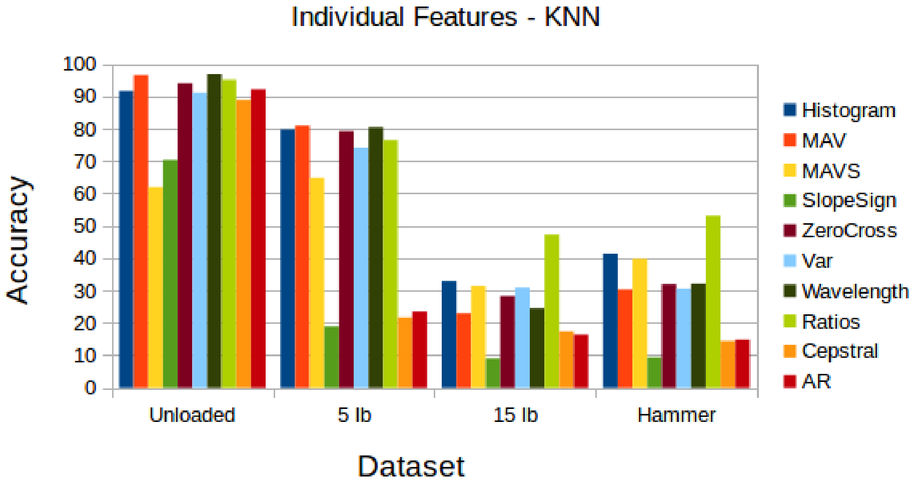
(a)

(b)

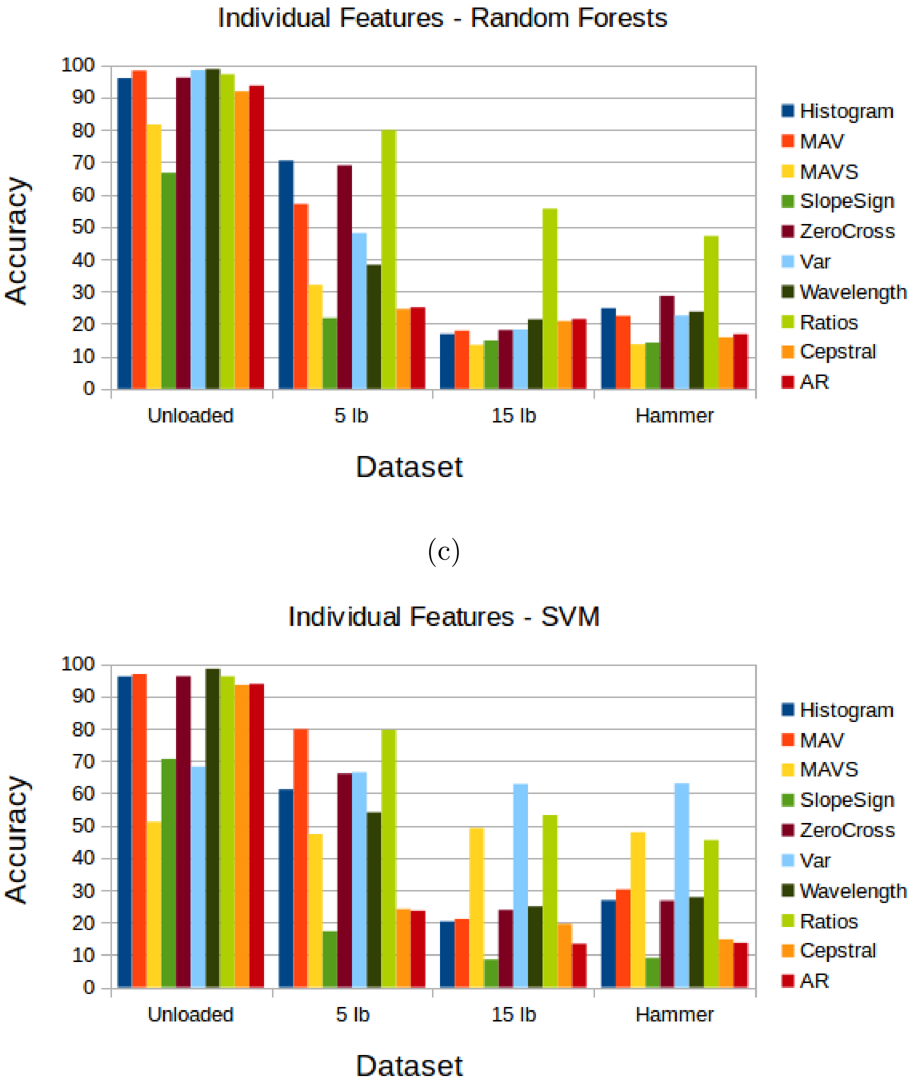
Figure 3.5: **Primary Result** a) The typical TD/AR feature set does not generalize well for concurrent contractions. b) Addition of the remaining features described in Section 3.3.1 does little to improve accuracy.


Individual Features - LDA
Accuracy
0
10
20
30
40
50
60
70
80
90
100
Unloaded
5 lb
15 lb
Hammer
Dataset
Histogram
MAV
MAVS
SlopeSign
ZeroCross
Var
Wavelength
Ratios
Cepstral
AR

(a)


Individual Features - KNN
Accuracy
0
10
20
30
40
50
60
70
80
90
100
Unloaded
5 lb
15 lb
Hammer
Dataset
Histogram
MAV
MAVS
SlopeSign
ZeroCross
Var
Wavelength
Ratios
Cepstral
AR

(b)



(c)

(d)

Figure 3.6: Individual Feature Generalization. Here we see the performance of each individual feature across the various datasets. It should be noted that the Unloaded dataset is an average of 10 cross-validations, while the other datasets show a single run as was previously described.

movement classes, consistent waveforms in the EMG signal could be seen within the first 300 ms. The development of their TD feature set was inspired by this and meant to capture more than just a crude mean absolute value, which we would not expect to generalize well, but to also capture aspects of the frequency response of the muscles. It is therefore surprising that the TD features do not generalize well. One might hypothesize that while levels of signals might shift in magnitude certain synergy signatures would still be present. However, we do not yet know the behavior of these synergy signatures when under concurrent contractions or when transitioning from one movement to another; they may not even be present. A crucial next-step in searching for features, that address the problem at hand, is to first study the signals themselves to better understand what is happening under concurrent contraction.

There are numerous features that have been proposed over the years for classifying EMG signals. Perhaps some among these sets might generalize better. In particular, if the synergy signatures are in fact present we might expect time-frequency features, such as wavelets, to perform better (Englehart et al., 1999).

As was previously mentioned, all of the standard features explored in this study looked only at single channel behavior. The novel *Channel Ratios* feature we developed was the only one to explicitly look at the interaction between the various channels. Looking at Figure 3.6 we see, that in general, this feature generalized the best. This feature not only looks at the interaction between the channels, but performs a normalization between the channels. It is therefore worth exploring additional features that would capture the relationships between channels as well as looking at normalization techniques.

While the performance of the *Channel Ratios* feature implies that exploring the relationship between channels may be beneficial, it is thus surprising to see that the KNN classifier appeared to generalize the best, with the LDA classifier doing the worst, since the LDA is able to take into account some of the interaction between features, while the KNN does not.

Often, the details of an implementation make all the difference. It is interesting to see that performance with all the features, or even just the TD/AR set, was often worse than the performance of the individual features making up those sets. This should not be that surprising as this runs into the typical curse of dimensionality issue often seen in machine learning, which might be overcome with more training samples. Feature reduction techniques, such as principal component analysis, which is often used in EMG classification experiments, may be of benefit here. Additionally, the features implemented in this experiment were not scaled relative to one another. It is known that scaling features such that they are all in approximately the same range can improve classification accuracy (Bishop, 2006).

One of the ways we might get around the issue of generalization is by simply ignoring it – that is, if we provided training samples that included biases and offsets this may serve to significantly improve online accuracy. While it is not possible to train for all scenarios, it is conceivable that only a few training samples with variations of loading might be sufficient to allow the classifier to interpolate and extrapolate between these samples. In fact, this is what the Coapt systems does; when an amputee finds that the classification accuracy is falling too low, they simply press a button, and the arm goes into a training phase where it collects new samples from the user and integrates these into its existing learned model. Another approach could be to create simulated signals that are then used for training. If the signal behavior during concurrent contractions can be understood it may be clear what sorts of effects are to be expected. These might then be used to generate new, artificial signals, by modifying the user generated signals, which could then be included in the training set.

The Coapt system was not commercially available at the time this study was performed. Now that it is, it would be valuable to evaluate the system under the same conditions explored in our experiments.

A clear limitation of this study is that only a single, able-bodied subject was used. A larger population would be useful in determining the robustness of the classifiers. Additionally, it has thus far been assumed that the experimental scenario explored here is a good analogue for the situation faced when classifying signals from an amputee with TMR surgery. This assumption would be greatly bolstered by actual experiments with TMR patients.

3.6.1 Functional tasks

After reviewing the literature on EMG classification systems and the results of this experiment, it is increasingly clear that EMG classification studies need to be carried out using amputee subjects, with mounted prosthetics, performing real-life tasks.

L. Hargrove et al. (2007) say "Pattern recognition based myoelectric control systems have been well researched; however very few systems have been implemented in a clinical environment. Although classification accuracy or classification error is the metric most often reported to describe how well these control systems perform, very little work research [sic] has been conducted to relate this measure to the usability of the system." Unfortunately, in this same paper they themselves continue evaluating their system in a virtual environment without any of the real effects that are seen in an actual worn prosthesis.

It seems clear that the evaluation of classification systems must move beyond sterile, controlled evaluations and on to real-world clinical tests. An evaluation method for upper-arm prosthesis, known as the *Southampton Hand Assessment Procedure* or SHAP test, is intended to provide many real-world outcome measures (Light et al., 2002). It includes tests such as lifting heavy objects, carton pouring, and handle turning. However, this is still performed in a controlled way, with subjects often seated performing a single movement task at a time. I do not believe that the example motivation given in this chapter, where a TMR patient might need to turn a handle while pushing the door, would be captured by this test, although similar issues with concurrent contractions might be.

3.7 Conclusion

In this study we looked at the generalization of typical pattern recognition implementations for classification of EMG signals. Specifically, we looked at whether a classifier, trained only with unloaded movement samples, could accurately classify movements made while the user was holding various objects. It was clear from the results that the most common approach, TD/AR features with an LDA classifier, does not generalize amidst concurrent contractions. This is significant as it is seems reasonable that concurrent contractions could arise commonly for amputees, either from the need for surrounding muscle to perform active functions, or simply the need to support the prosthetic socket itself. There is a clear need to design more robust classifier systems and to evaluate such systems in more varied conditions under actual usage scenarios.

Chapter 4

Multilayer General Value Functions for Robotic Prediction and Control[1]

While Chapter 3 focused on the use of offline pattern recognition methods, this chapter is the first time in this thesis where online methods, using the reinforcement learning technique known as general value functions, are introduced.

The first and probably most significant contribution of this chapter is a detailed discussion of the problem of prosthetic control and how we might go about making the arms more intelligent.

Further, this chapter shows that layers of general value functions (i.e., the output of one or more predictor is used as input to another predictor) are indeed possible and that such configurations may improve accuracy of the resulting predictions by improving the representation of the data. While experiments are conducted on a mobile robot and in simulation they are still applicable to the control of a prosthetic arm as they focus on establishing complex predictions about sensori-motor data. Such predictions should allow a system to better understand the user, the robot, and the environment and therefore make

[1] This chapter is based on a paper presented at the Workshop on AI and Robotics at the International Conference on Intelligent Robots and Systems, 2014. The original experiments were performed as part of a research project completed for credit in CMPUT 656, during my MSc in Computing Science. This version includes some updated experiments. The paper originally appeared as *Sherstan, C. and Pilarski, P. M., "Multilayer General Value Functions for Robotic Prediction and Control", in IROS 2014 Workshop on AI and Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, Illinois, September 14-18, 2014.*

better informed decisions.

4.1 Overview

Predictions are a key component to intelligence and necessary for accurate motor control. In reinforcement learning, such predictions can be made through general value functions (GVFs). This chapter introduces prosthetic arms as a domain for artificial intelligence and discusses the role that predictions play in prosthetic limb control. We explore the use of multilayer predictions, that is, predictions based on predictions, using robotic and simulation experiments. From these experiments two observations are made. The first is that compound predictions based on GVFs are viable in a robotic setting. The second, is that strong GVF predictors can be built from weaker ones with different input and target signals, similar to boosting. Finally, we theorize how such topologies might be used in transfer learning and in the simultaneous control of multiple actuators. Our approach to integrating machine intelligence with robotics has the potential to directly improve the real-world performance of bionic limbs.

4.2 Gentle Integration

When combining machine intelligence systems with electromechanical devices such as mobile or mounted robots, it is natural to think of the machine intelligence as providing most or all of the key aspects of the robot's control system. Integration of this kind is often challenging—it simultaneously addresses many important barriers faced by our computing technology—but is incredibly fruitful for both the fields of robotics and artificial intelligence. Another, complementary approach is the use of machine intelligence to supplement an existing control system or sensorimotor interface. Machine learning and artificial intelligence (AI) can augment the capacity of existing systems in small but important ways. While more modest in its aims, this kind of staged deployment is well suited to the refined study of individual machine learning methods as they impact real-world domains of use. It further provides a smooth pathway to machine intelligence seeing practical use within complete, deployed systems.

In this chapter we look specifically at the second, more gentle approach to integrating machine intelligence within a robotic device. In particular, we highlight one area where our group has made recent progress: improving robotic artificial limbs (Figure 4.1) through real time learning and utilization of temporally extended predictions. This setting lends itself well to translating algorithmic and conceptual advances into tangible benefit within a deployed environment; machine learning can improve the ability of people with amputations to control their bionic limbs. Sharing the challenges and opportunities of prosthetics as a domain for AI Robotics is the first contribution of this chapter. We present a brief overview of our machine learning work within the prosthetic domain, and follow on this overview with a concrete example on a simple robotic platform of how real time predictions can be beneficially combined into a learning hierarchy. Lastly, we discuss how multilayer predictions can be integrated back into prosthetic control approaches to further extend their practical reach.

4.3 Bionic Limbs

Bionic limbs are robotic devices fixed directly to the body of someone with a motor impairment or complication (e.g, someone with an amputation), or for the purposes of extending or augmenting the abilities of healthy individuals. These devices have multiple actuators and sensors, both on and off the human body, and use this sensorimotor information to interpret a user's intent and actuate the joints of the robot limb accordingly. Despite the growing availability of dexterous robotic prosthetic arms, amputees often reject these arms due to the difficulty they find in their control (Peerdeman et al., 2011; E. Scheme and Englehart, 2011; Resnik, Meucci, et al., 2012). The most common approach to controlling such arms is the use of electromyographic signals (EMG), which are the electrical activities of muscles. Unfortunately, the number of control signals available from EMG is much lower than the control space of the robot arms, creating a large gap between user intent and achievable

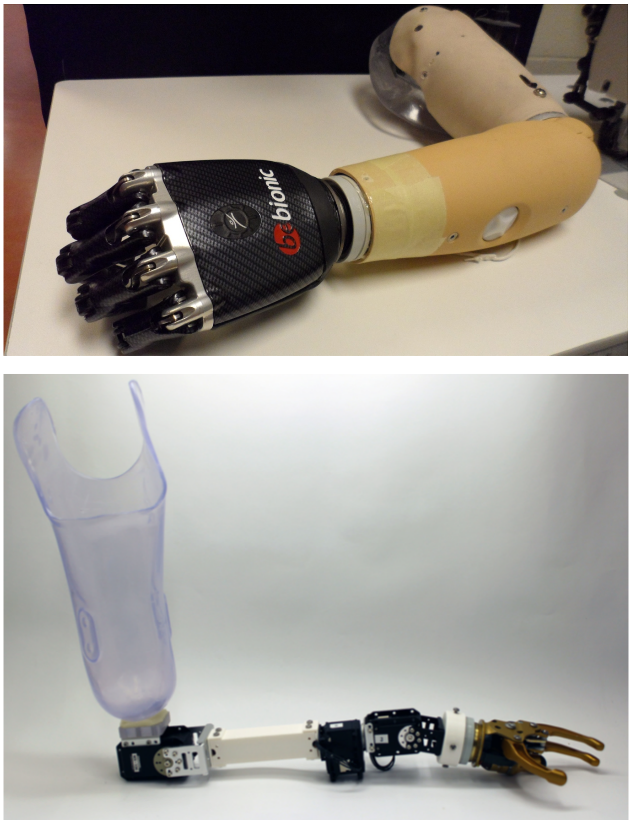


Figure 4.1: Augmentative and restorative prosthetics are of specific interest for incrementally integrating AI into a robotic setting. *Top:* commercial limb system prescribed to an amputee for use during daily life. *Bottom:* research robot limb system with direct access to a rich sensorimotor stream (Dawson et al., 2014).

motor outcomes. There are a number of techniques people have tried to address this gap, as noted in Chapters 2 and 3 some on the software side, such as pattern recognition (E. Scheme and Englehart, 2011), and some on the clinical side, such as targeted muscle reinnervation (Hebert et al., 2014). However, control remains difficult and indeed, there will almost always be a disparity between signal and control spaces.

Our goal is to apply artificial intelligence to the control of these arms, in such a way as to make using them more intuitive and functional for the users (Pilarski, Dick, et al., 2013; Pilarski, Dawson, Degris, Carey, Chan, et al., 2013; Edwards et al., 2014; Pilarski, Dawson, Degris, Carey, and R. S. Sutton, 2012). We propose that a more complete way to think about the prosthetic control problem is that we are looking to create an assistive, context aware robot, which happens to be a prosthetic arm. The techniques we are developing here are also applicable beyond the scope of prosthetic arms. Our approach has been to incrementally apply AI techniques to existing control schemes for other assistive and augmentative devices. One of the great benefits of working with prosthetic devices is that the users of these devices have clear objectives that they need the prostheses to address, and concrete measures for the success of the system. Additionally, there is a clear path to commercial and clinical use.

4.4 Improvement from Ongoing Experience

Making forward predictions is believed to be a key component in making accurate motor commands (Wolpert et al., 2001; Redish, 2013; Linden, 2003). Furthermore, predictions have been shown to be an important way to think about and formalize the state information being provided to a learner (for example, predictive representations of state (Littman et al., 2002)). By learning and maintaining predictions in real time, it is possible for a robotic system to acquire and self-verify small pieces of knowledge in an autonomous fashion as it interacts with the world (R. S. Sutton, Modayil, et al., 2011; Modayil and R. S. Sutton, 2014; Modayil, A. White, and R. S. Sutton, 2014; A. White et al., 2014).

Incremental, ongoing knowledge can be acquired using techniques known as general value functions (GVFs), as introduced in Chapter 2, are a generalization of the reward-based value functions common in reinforcement learning (RL) (R. S. Sutton, Modayil, et al., 2011). While other forms of machine learning might be used for prediction, GVFs are somewhat unique in their ability to learn online and continuously in a computationally efficient manner. In GVFs, replacing reward with a target signal allows a system to learn either cumulative, (4.1), or instantaneous, (4.2), predictions for any scalar signal. For example, we can ask “How much total current will the shoulder servo use in the next 10s?” or “What will the light sensor read in 3s?” GVFs can also be used to give the probability of a binary event occurring, e.g., “What is the probability of colliding with the wall in the next 5s?” GVFs can be thought of as representing temporally extended knowledge about a robot, its environment, and the interaction between the two.

$$\delta_{t+1} = R_{t+1} + \gamma\phi_{t+1}^{T}\boldsymbol{w}_t - \boldsymbol{\phi}_t^{T}\boldsymbol{w}_t \tag{4.1}$$

$$\delta_{t+1} = \beta R_{t+1} + \gamma\boldsymbol{\phi}_{t+1}^{T}\boldsymbol{w}_t - \boldsymbol{\phi}_t^{T}\boldsymbol{w}_t \tag{4.2}$$

where

δ – temporal difference error

R – in GVFs this represents the target signal to be predicted

γ – continuation probability, # timesteps lookahead$= 1/(1-\gamma)$

$\boldsymbol{\phi}$ – input feature vector

$\boldsymbol{w}$ – learned weight vector

β – termination probability $= 1 - \gamma$

The GVF algorithm is composed of three main steps: calculation of the temporal difference (TD) error ((4.1), (4.2)), calculation of traces, and weight vector update. Equations (4.3) and (4.4) show the traces and weight update used in TD(λ) with replacing traces, which are employed in Section 4.5.2.

$$\boldsymbol{e}_{t+1} = \lambda\gamma\boldsymbol{e}_t + \frac{\alpha\boldsymbol{\phi}_t}{max(1, ||\boldsymbol{\phi}_t||_0)} \tag{4.3}$$

where

$\boldsymbol{e}$ – is the eligibility trace

λ – trace decay rate (amount of bootstrapping)

α – step size

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \delta_{t+1}\boldsymbol{e}_t \tag{4.4}$$

The True Online TD(λ) algorithm is a more recent development with better performance and stability guarantees (van Seijen and R. S. Sutton, 2014). Equations (4.5) and (4.6) show the newer traces and weight updates, which are employed in the Create robot experiments in Section 4.5.1.

$$\mathbf{e}_t = \gamma\lambda\mathbf{e}_{t-1} + \alpha_t\boldsymbol{\phi}_t - \alpha_t\gamma\lambda[\mathbf{e}_{t-1}^{\mathsf{T}}\boldsymbol{\phi}_t]\boldsymbol{\phi}_t \tag{4.5}$$

$$\boldsymbol{w}_{t+1} = \mathbf{w}_t + \delta_t\mathbf{e}_t + \alpha_t[\boldsymbol{w}_{t-1}^{\mathsf{T}}\boldsymbol{\phi}_t - \boldsymbol{w}_t^{\mathsf{T}}\boldsymbol{\phi}_t]\boldsymbol{\phi}_t \tag{4.6}$$

GVFs have seen some promising application with robots. Sutton et al. demonstrated that GVFs were able to simultaneously learn to predict large numbers of sensorimotor signals in an online fashion on a mobile robot (R. S. Sutton, Modayil, et al., 2011; Modayil, A. White, Pilarski, et al., 2012). Some studies have also looked at using GVFs in control. In particular, Modayil and Sutton have used prediction with a nexting approach to control a simple mobile robot, such that, when a prediction exceeds a threshold the robot will activate and follow a fixed, hand-coded behavior (Modayil and R. S. Sutton, 2014). Specifically, when their mobile robot predicted a future over-current condition it would shut off the motors. This approach is similar to many prediction-based reflexive reactions found in humans and other animals (Redish, 2013; Linden, 2003).

The idea to make a predictive link to known control behaviors also fits well within the domain of artificial limbs. A typical control setup for using EMG to control a prosthetic arm is to use two EMG signals to proportionally control the velocity of one joint at a time. Active joint selection is performed by toggling through a fixed joint list via another EMG signal or a mechanical switch. As one can imagine, this is a tedious way to control an arm. Edwards et al. (2014) have demonstrated improved task performance using an adaptive switching order based on learned predictions. When an amputee user begins a toggle sequence, the joints are selected in the order that the learner predicts will be most likely needed at the moment; this was found to reduce the number of voluntary switching interactions needed to complete a simple manipulation task, and thus also the time needed to complete the task. Users appeared to be happy with the improvement and to develop increased trust in the system. Additionally, Pilarski, Dick, et al. (2013) controlled the wrist joint of a 3 DOF robot arm where the objective was to have the controller place the wrist in the position it predicted it should go in the near future, given the current state. This study demonstrated the ability to use GVF predictions as direct target signals for control as well as in combination with actor-critic RL agents (e.g., as predictive state information).

As has already been discussed, GVFs can be thought of as representing temporally extended knowledge about an agent, its environment and the interaction between the two. Thus far, they have been presented as only answering simple primary questions. However, they can certainly be used for more complex questions as well. Imagine that we have two predictors: 1. "Is a Tiger nearby?", 2. "Will I have an asthma attack in the near future?" An important prediction to make is, "Am I in danger?", for which the previous two predictions would be valuable. In the context of a robotic arm we can imagine the usefulness of such compound questions. For example, in a prosthetic task we could structure a set of GVF predictions as follows:

- Where is the elbow moving to? Where is the shoulder moving to? → Where should the wrist move to?
- Where is my hand moving to? Does Joe want coffee? Is there coffee in front of Joe? → Should I open my hand?

The work presented in this chapter explores such compound questions using layers of predictions. That is, the output of one or more predictors is used as input for another predictor as shown in Figure 4.2. We first demonstrate that GVFs can be used in this way to produce compound predictions and secondly investigate some properties

4.5 Experiments

To examine the feasibility of multilayer GVF predictions in prosthesis use, we first performed a set of preliminary tests on a more controlled experimental setting. Architectures like those

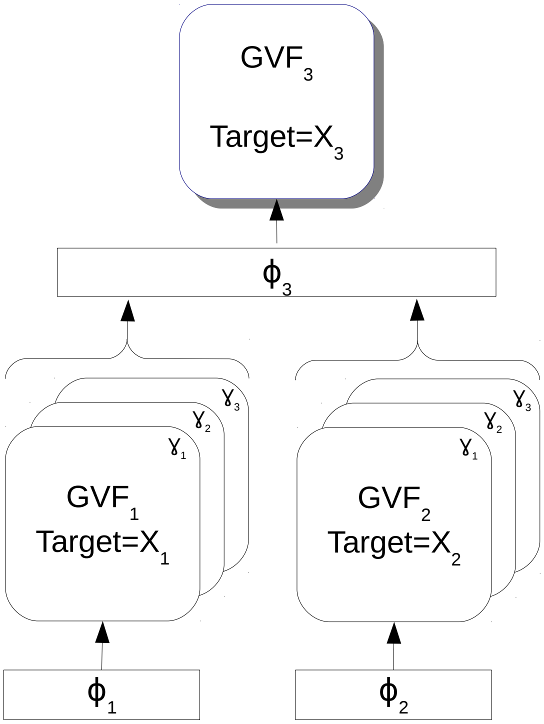


Figure 4.2: Topology. From bottom up: $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are primary layer feature vectors, which may or may not be the same, depending on the experiment. Primary layer GVFs are grouped by the target signal, with one or more lookahead values (γ). The output of the primary layers are then used as input, possibly with other inputs, as features to a secondary layer GVF.

shown in Figure 4.2 were tested both in simulation, using simple square wave trains, and on an iRobot Create mobile robot (Figure 4.3). GVF learning was conducted as described in prior work (Modayil and R. S. Sutton, 2014; Modayil, A. White, and R. S. Sutton, 2014; Pilarski, Dawson, Degris, Carey, Chan, et al., 2013).

4.5.1 Create Robot

The Create robot (iRobot, Inc.) is a simple mobile robot with a limited number of sensors, similar to a Roomba vacuum. The sensors used in this experiment are listed below.

- 4 downward facing cliff sensors along the front edge. Thresholded to a binary on/off signal.
- 1 forward facing wall sensor. Thresholded to a binary on/off signal.

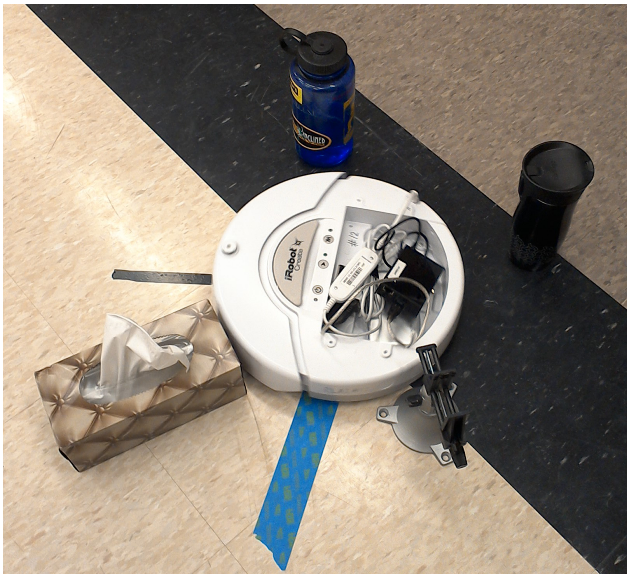


Figure 4.3: Create recording session.

The goal for this experiment was to accurately predict the turning on and off of three of the cliff sensors (Left, Front-Left, Front-right) at the primary layer and then make predictions about the fourth cliff sensor (Right) based only on the outputs of the primary layer. The Create rotated counter-clockwise for 20 minutes, randomly changing speed every 2 minutes. As the Create spun it passed over various surfaces: black tape, blue tape, beige tiles, and black tiles. Additionally, objects were placed around the Create, which gave readings for the forward facing wall sensor. Under this behavior the Right cliff sensor would be the last sensor in the sequence to pass over a given surface. Figure 4.3 shows the experimental setup.

Control and data recording was performed at 30 Hz on a Raspberry Pi running the Robot Operating System (ROS) (Quigley et al., 2009). Prediction was performed offline after recording.

Cliff and wall sensor values were converted to binary values by using a threshold. Signals above the threshold were given a binary value of 1.0, and those below it were given the value 0.0. As each sensor had slightly different sensitivities different threshold values were used for each. History traces of each of these values were made using (4.7), with decay rates 0.8

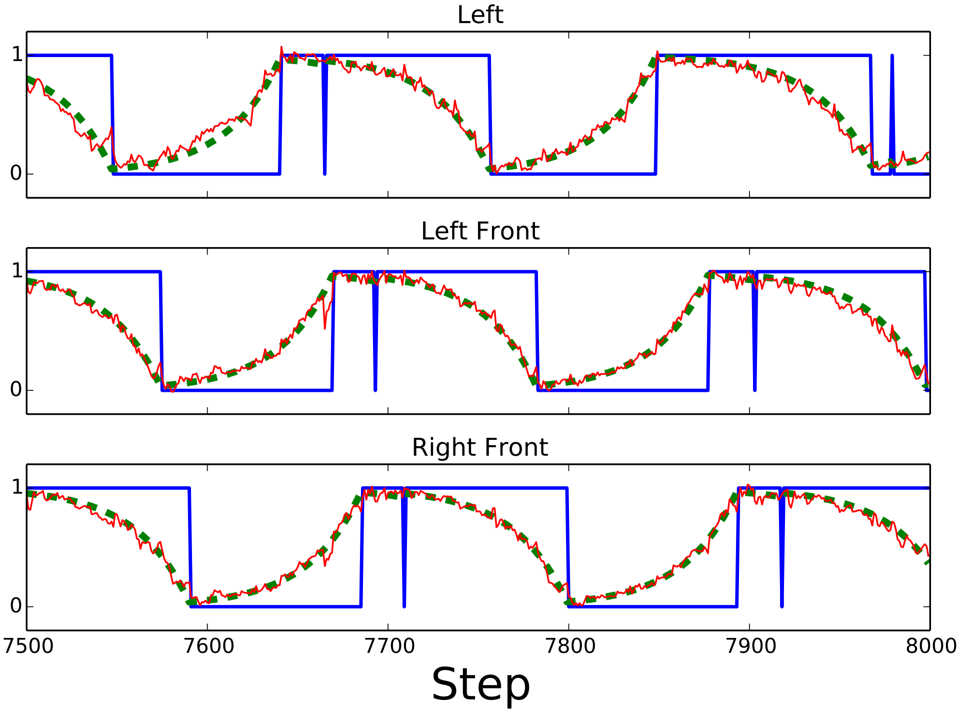


Figure 4.4: Primary layer predictions of Create cliff sensors. The target signal (blue) is predicted at 30 timesteps (red). The ideal prediction is shown with the dotted green line

and 0.98, for a total of 10 traces.

$$H_{t+1} = (1 - \xi) * X + \xi * H_t \tag{4.7}$$

where

H – the history trace

X – the target signal being tracked

$0 \leq \xi \leq 1$ – a decay rate

These traces were then used in a 10 dimensional tile-coding using Sutton's library (R. Sutton, 2005) to produce 100 tilings. The bin width along each dimension was 1.0 and the results were hashed to a size of 2048. Combined with a bias feature this produced a feature vector of 2049.

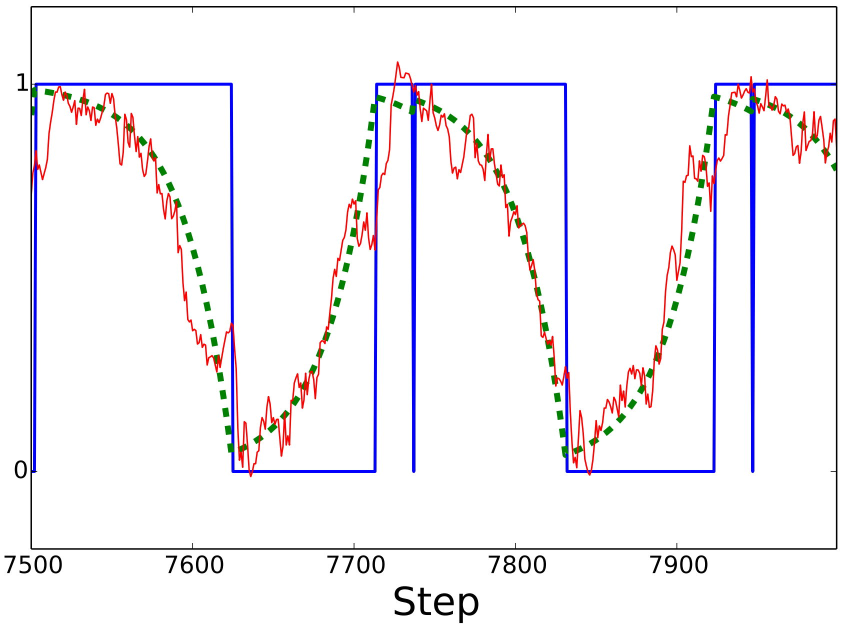


Figure 4.5: *Key Result:* Successfully predicting the Right Cliff sensor at the secondary layer GVF. A 30 timestep (1 s) prediction (red) is compared against the ideal prediction (dashed green) for the target signal (blue).

The secondary layer took the output of the the primary layer GVFs for Left, Front Left and Front Right cliff sensors. Traces of these signals were made using (4.7), with decay rates 0.8 and 0.98, producing 6 traces. These traces were then used in a 6 dimensional tile-coding to produce 100 tilings, with each dimension having a bin width of 1.0. The results were hashed to a size of 2048 and a bias feature was added for a total feature vector size of 2049.

The GVFs used in this experiment were computed using the True Online TD(λ) algorithm with $\alpha = 1.0/(\#\text{ Tilings} + 1)$ and $\lambda = 0.95$ (van Seijen and R. S. Sutton, 2014).

In Figure 4.4 we see that very good predictions were made for the Left, Front Left, and Front Right cliff sensors at the primary layer.

As can be seen in Figure 4.5, it was indeed possible to learn to make predictions for the Right cliff sensor using the outputs of the primary layer GVFs. While these results are expected, it is important to establish the validity of the topology and implementation used.

4.5.2 Combining Weak Predictors to Produce a Stronger Predictor

We also examined whether combining weak predictors could produce a stronger predictor, akin to the concept of boosting in machine learning (Schapire, 1990). This scenario was tested using three square pulses of the same size, but different temporal offsets, as targets and input signals. In this setting GVF_1 (Target=X_1, timescales=2,4,8,10 timesteps) and GVF_2 (Target=X_2, timescales=2,4,8,10 timesteps) used an impoverished feature space that was not sufficient to predict the signal. The primary layer features, ϕ_1 and ϕ_2, contained only a bias feature, the target signal (X), and 1-X, for a total of 3 features.

The 8 predictions from the primary layer GVFs (4 predictions from each of the 2 GVF groups) were then tile coded individually and in pairs. Each tile coding consisted of 3 tilings with bin widths of 0.5 and 6 tilings with bin widths of 0.17. For each individual signal this produced 51 features. For each of the 28 pairs of signals this produced 321 features. In total the feature vector for the secondary layer was 9397 features long, including the bias feature. GVF_3's target was the third square pulse, X_3, and was predicted at a lookahead of 4 timesteps. GVFs were implemented using TD(λ) using (4.2),4.3,4.4. Additionally, tile coding was performed using a custom written library without the use of hashing.

The best that the primary layer GVFs could do with such an inadequate state space was to chase the signal. Despite this, GVF_3 was able to learn to accurately predict X_3 as shown in Figure 4.6.

Essentially, the output of the primary GVFs served as a form of history for the two signals, providing more information about the signals than was directly available from the representation used. It seems reasonable to conclude that we should expect this sort of boosting behavior as long as the primary GVFs are at least somewhat temporally correlated with that secondary layer target.

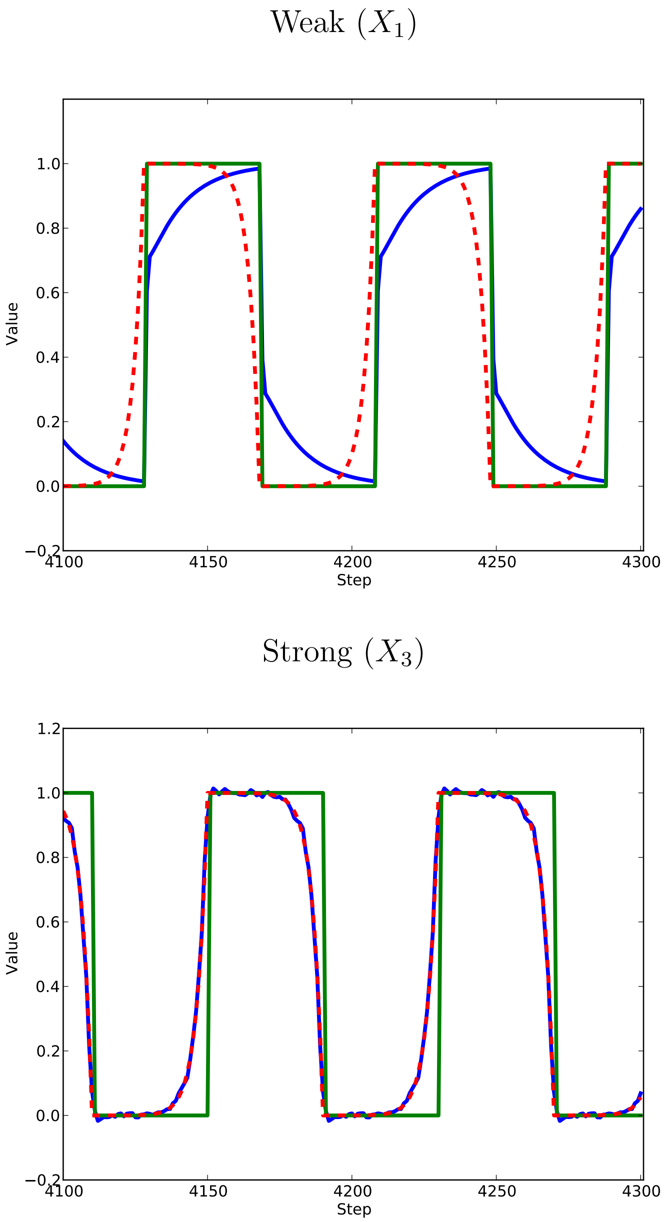


Figure 4.6: Weak and strong predictors. *Top*: A Weak predictor, target=X_1, which is only able to chase the signal, not predict it. The distinctive shape of the blue line is a clear indicator that the state space is insufficient. The graph for target X_2, not shown, is the same, just shifted in time. *Bottom*: A Strong predictor, target=X_3 is learned with only weak predictors as input. *Solid Green:* Target signal, *Solid Blue:* Prediction at 4 timesteps, *Dashed Pink:* Ideal prediction for 4 timesteps calculated from post-processing

.

4.6 Moving Forward: Opportunities for Integration

We believe that the use of GVFs and layers of GVFs will prove beneficial to the simultaneous multi-joint control of prosthetic arms. In particular, we propose two applications that go beyond what has already been demonstrated with the adaptive switching work demonstrated by Edwards et al. (2014).

4.6.1 Transfer Learning between Simulation and Real-world

Learning on a robot is expensive in terms of time and risk to biological and mechanical hardware. For these reasons it is desirable to be able to train in simulation and then transfer what is learned to the real world. One approach in RL is to learn a policy in simulation and then use that learned policy in the real world, although this has had limited success (Kormushev et al., 2013). The topologies of GVFs presented in this paper suggest an approach like the one shown in Figure 4.7. In this scenario, a GVF learns to predict some signal, such as joint angle, in simulation. In the real world, another GVF learns to predict the same signal, using the output of the simulation learned GVF as an adviser, in coordination with other input data. In theory, this should allow for more rapid learning in the real world, with GVFs that are already partially learned. In reality, we do not expect that the transferred GVF should predict that well, given the difficulty of accurately simulating. However, the results presented in this paper, where a strong predictor was based on weak ones, lead us to believe that we should still see some benefit using this technique. Our hope is that this will greatly reduce the amount of time needed for an amputee to train their prosthetic. Additionally, this technique could also be used with aggregated learning where the adviser is a GVF representing the cumulative predictive advice learned by many robots or from interactions with many users.

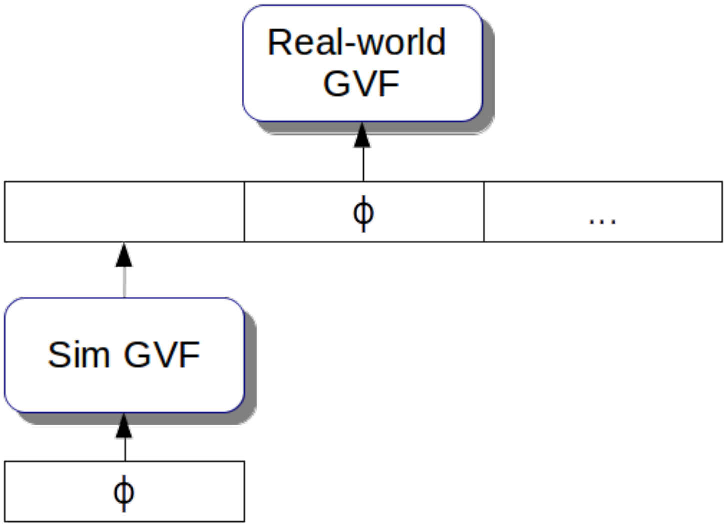


Figure 4.7: Transfer learning using a multilayer topology of GVFs.

4.6.2 Predictions for Simultaneous Multi-joint Control

Ultimately, our aim is to use predictions for control (Figure 4.8). One particular challenge of interest is the simultaneous control of multiple joints of a prosthetic limb via limited input channels—an open issue in the prosthetic domain (E. Scheme and Englehart, 2011).

As was mentioned, prior studies have shown clear, task specific ways of basing control on predictions (Modayil and R. S. Sutton, 2014; Edwards et al., 2014; Pilarski, Dawson, Degris, Carey, and R. S. Sutton, 2012; Pilarski, Dawson, Degris, Carey, Chan, et al., 2013). Predictions represent a type of temporal forward model, which are useful to thinking agents, be they biological or mechanical, and are a necessary component in developing good motor control, asking questions like, "Where is my hand moving?", "Am I going to collide with something?", and "What direction will I be heading 3 s from now?" They are also important for higher levels of intelligence and control. For an intelligent assistive robot, such as the prosthetics we are creating, understanding a user, their environment and the current situation are long term goals. In order to do this, higher level predictions are necessary, such as, "Is

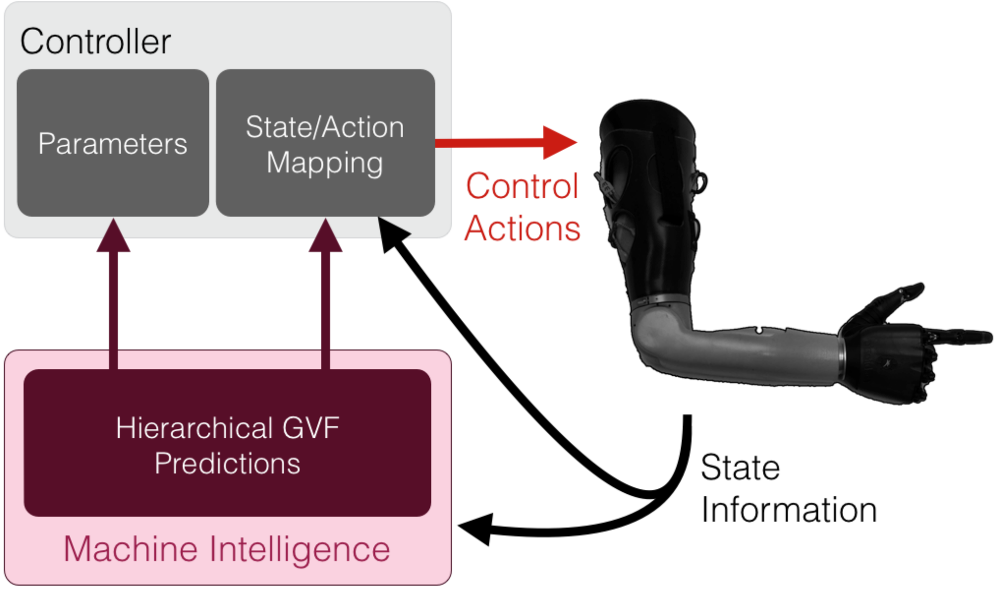


Figure 4.8: *Integration approach:* machine intelligence and automatically acquired knowledge—in this case a multilayer topology of predictions—is used to extend the capacity of conventional control systems within an artificial limb.

the user upset?", "Is the user hungry?", "Is the user in danger?", "Which object might the user want to grab?". At both levels predictions are useful and understanding them at the more primitive level is an incremental step towards understanding the more complicated types of predictions needed for the higher level.

For low-level control there are specific ways in which we might leverage layers of predictors. For example, it may be useful to make a prediction about the target position of the wrist given predictions about the target positions of all the other joints in a robot arm. Additionally, by using layers of predictions we have the potential benefit of speeding learning, where, under certain circumstances, we can imagine a reduction in state space at the secondary or higher levels of predictors. Finally, under certain circumstances, we would expect to see a gain computationally where a particular prediction might be leveraged in many layers. This would be more efficient than having each of the secondary layers calculating predictions directly from the data themselves, each performing the same calculations.

4.7 Conclusion

As a first contribution of this work, we identified one domain—that of robotic artificial limbs—where the integration of machine intelligence with robotic systems has both clear utility and immediate areas for incremental progress. The second contribution of this work was to examine the use of multilayer topologies of prediction learners, particularly as they would apply to robots. Two main results were observed from these experiments. The first is that it is possible to learn a reliable prediction during robot operation when using the output of other predictors as input. To our knowledge, this is the first example of multilayer GVFs being applied during robot control. The second result is that it is possible to combine the output of weak GVF predictors with different target signals and input spaces to create a strong predictor of a third target signal. These two results will be useful in developing robust control methods for prosthetic robots; as a final contribution of this chapter, we suggested two ways that multilayer predictions could be beneficially deployed within bionic limbs and other robotic applications. Future work in this area promises to benefit both the users of human-machine interfaces and researchers seeking to better understand the links that can be made between robot control and advances in machine intelligence.

Chapter 5

A Collaborative Approach to the Simultaneous Multi-joint Control of a Prosthetic Arm[1]

This chapter presents the most significant contribution of this thesis. While the previous chapter focused on prediction using general value functions, this chapter now incorporates such predictions in real time control. The chapter introduces a new collaborative control method whereby a user and a robot arm control the arm together to achieve the user's goals. While the evaluation of the method was limited to a single user and a single task, the results demonstrate the potential to improve control of a prosthetic arm.

As is discussed in subsequent sections of this thesis, there are many ways in which collaborative control might be implemented. This chapter has focused on one particular approach. The real take away message is that collaborative control, in general, offers a way of improving prosthetic control.

[1] This chapter is largely based on a paper submitted for publication as *Sherstan, C., Modayil, J., and Pilarski, P. M., "A Collaborative Approach to Effecting Simultaneous Multi-joint Control of a Prosthetic Arm", International Conference on Rehabilitation Robotics (ICORR), Singapore, August 2015.* The paper has been accepted and will be given as a podium presentation. An extended abstract of this paper was also presented as both a podium and poster presentation at the Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM), Edmonton, Alberta, June 7-10, 2015.

5.1 Overview

We have developed a real time machine learning approach for the collaborative control of a prosthetic arm. Upper-limb amputees are often extremely limited in the number of inputs they can provide to their prosthetic device, typically controlling only one joint at a time with the ability to toggle their control between the different joints of their prosthesis. Many users therefore consider the control of modern prostheses to be laborious and non-intuitive. To address these difficulties, we have developed a method called Direct Predictive Collaborative Control that uses a reinforcement learning technique known as general value functions to make temporally extended predictions about a user's behavior. These predictions are directly mapped to the control of unattended actuators to produce movement synergies. We evaluate our method during the myoelectric control of a multi-joint robot arm and show that it improves a user's ability to perform coordinated movement tasks. Additionally, we show that this method learns directly from a user's behavior and can be used without the need for a separate or pre-specified training environment. Our approach learns coordinated movements in real time, during a user's ongoing, uninterrupted use of a device. While this paper is specifically focused on the control of prosthetic arms, there are many human-machine interface problems where the number of controllable functions exceeds the number of functions a user can attend to at any given moment. Our approach may therefore benefit other domains where a human and an assistive device must coordinate their efforts to achieve a goal.

5.2 Introduction

The control of powered prosthetic arms is at times difficult and tedious. In fact, many amputees will eschew their powered arms for mechanical ones because of the difficulty of control (Peerdeman et al., 2011; E. Scheme and Englehart, 2011; Resnik, Meucci, et al., 2012). Those amputees that have adopted powered prosthetic arms typically control their device using electrical signals produced by muscle contraction in their residual limb, which is known as electromyography (EMG) or *myoelectric control.* While many EMG-based control approaches have been explored since initial work in the 1960s, in myoelectric control two contraction sites are still typically needed to control a single joint (Parker et al., 2006). One site (e.g., biceps) is used to move the prosthetic joint in one direction and another site (e.g., triceps) is used to move the joint in the other direction. When more than one prosthetic joint is available, a third signal or combination of signals is used to toggle between joints. We will refer to this form of control as *toggling proportional control* (TPC). As can be expected, controlling robotic arms using TPC becomes increasingly difficult as the number of joints increases. With the recent development of arms with high degrees of freedom, such as the Modular Prosthetic Limb (Johns Hopkins University) and the DEKA arm (DEKA Research & Development Corporation), there is increasing need for control systems that reduce the burden of control on amputees while accommodating increased prosthesis complexity. Furthermore, the inability to control more than one joint at a time rules out natural synergies (coordinated multi-joint actions) that are available to non-amputees.

Assistive rehabilitation robotics, and prosthetics specifically, are technologies where functions—also called degrees of control (DOC)—often significantly outnumber the control inputs that can be provided by their user. As such, managing and coordinating multiple DOC simultaneously is crucial to the use of more advanced assistive technology. Automation is one approach to managing multiple simultaneous operations that is commonly applied in engineering and industrial settings. Users of next-generation artificial limbs may therefore also benefit from a mixture of autonomous control and user control that helps them better utilize the multiple DOC provided by their prostheses. Previous explorations of the application of autonomy to assistive devices indicate that users prefer to have some degree of direct involvement rather than having their assistive device behave fully autonomously (Viswanathan et al., 2014; Cipriani, Zaccone, et al., 2008). However, it is still not clear what form of partial autonomy would be preferred by amputees, and in fact it may be different for each person and change depending on the situation.

In this work we explore how machine learning of predictions may help manage the integration of user and automatic control. Research suggests that prediction is a key component in movement planning and that anticipatory action plays a role in producing coordinated movements, or synergies (Wolpert et al., 2001). With this in mind, recent work by our group explored several methods for producing synergies using anticipatory movements of unattended joints in a prosthetic arm based on predictions made about a target angle (Pilarski, Dick, et al., 2013). In this work, a user controlled the elbow and gripper of a three-DOC arm, while automation controlled wrist rotation with the goal of moving the wrist joint to an anticipated target angle based on the current state. Put differently, a prediction about the target angle, some time in the future, was used to direct the movement of the wrist joint. This target angle was programmatically provided to the system in some way. To make predictions about the target wrist angle, Pilarski et al. used a generalization of the reward-based value functions used in reinforcement learning (RL), known as general value functions (GVFs), which are capable of learning multi-step predictions about any measurable signal (R. S. Sutton, Modayil, et al., 2011). GVFs incrementally learn predictions in an online setting, with linear computation. One particular control policy explored by Pilarski et al. was called *Direct Predictive Control*, which directly maps predictions about target angles into control commands (Pilarski, Dick, et al., 2013). It was shown to be effective in quickly learning a good control policy given a provided target angle.

Our first contribution in the present work is the description and evaluation of an extension of this method, which we term *Direct Predictive Collaborative Control* (DPCC). In order to compare the performance of our method against TPC we adopt the commonly held perspective that it is better to complete a manipulation task faster, and that manual interactions with the system (toggles) places a burden on the user. We show that DPCC achieves movement synergies and improves user performance by reducing task time and the number of toggles. As a second contribution, we demonstrate, that unlike previous work, it is possible to learn target angles without the need to programmatically provide these target angles to the system. Rather, they can be learned directly by observing the user's behavior.

One of the goals of the present work was to develop technology that could be readily translatable to clinical application. As such, our work respects constraints inherent in conventional myoelectric control under what is arguably the most limited case, i.e., where the

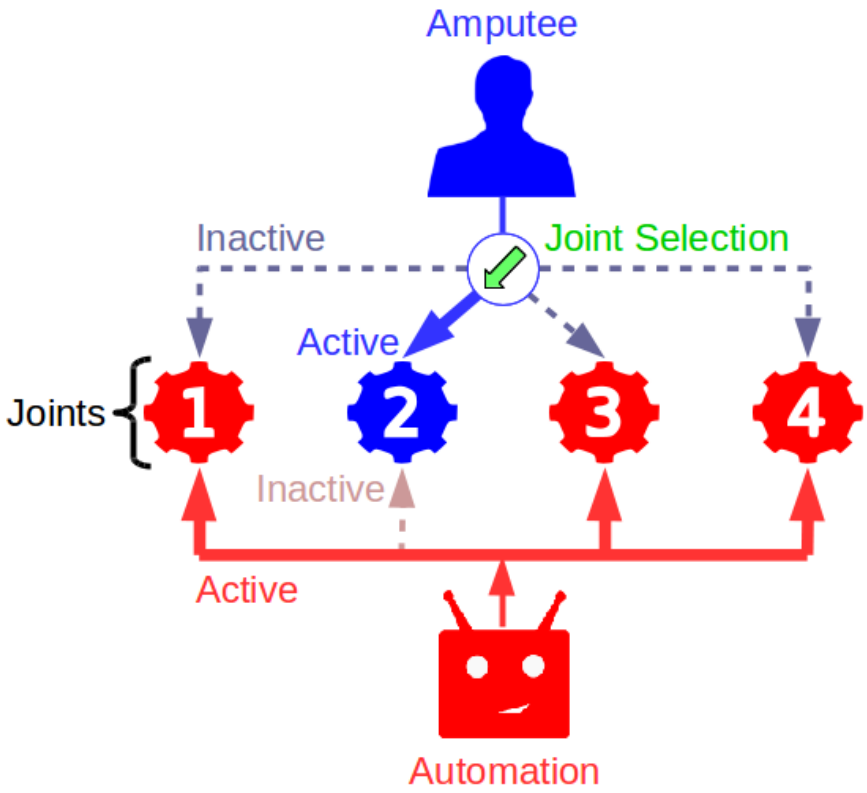


Figure 5.1: The collaborative control scheme used in this paper. The user (blue) can control only one joint at a time, while the robot (red), in our case the machine learner, controls all the other joints. Joints are indicated by the gears and active control is indicated by solid lines. Dashed lines indicate inactive control pathways available to either the human or the robot.

user can only produce a scalar signal and a toggle signal, as might be the case with a transhumeral amputee. Additionally, we sought to work within the input and output constraints of conventional prosthetic hardware. As such, we were interested in ways to improve control without adding extra user interface channels or additional sensor modalities like those described in Novak and Riener, 2014.

5.3 Direct Predictive Collaborative Control

In what follows, we use the term collaborative control to mean two or more agents working together towards a common goal, with only one agent acting on any one DOC at a time. Our proposed approach, Direct Predictive Collaborative Control (DPCC), is a method where a user can choose to attend to any controllable function, but is limited to attending to only a

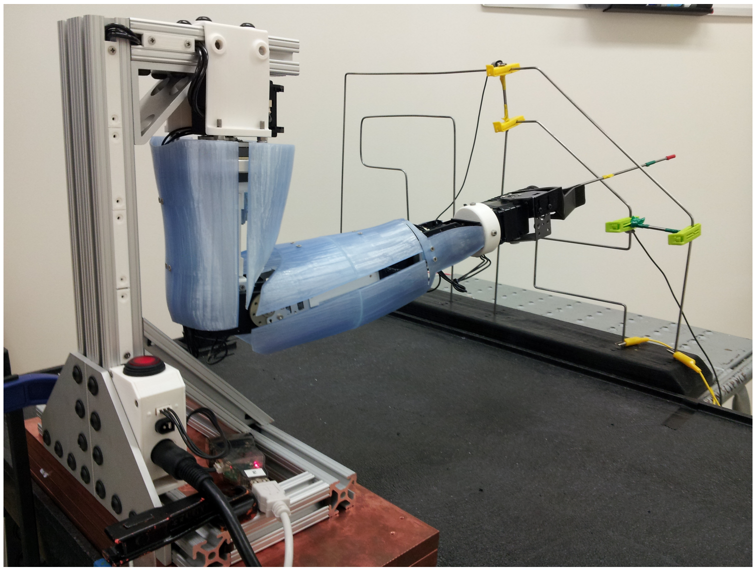


Figure 5.2: The Bento Arm (Dawson et al., 2014), shown configured for the Angle Maze experiment with a conductive rod attached to the gripper.

subset at a given time. Unattended DOC are then controlled by automation (see Figure 5.1). As noted, DPCC extends the *direct predictive control* method of Pilarski, Dick, et al. (2013). We employ DPCC in prosthetic control with the user controlling one joint at a time while automation controls the other joints. Predictions about future joint angles are used to generate velocity commands for a given joint according to (5.1).

$$V_{t+1} = (P_{t+1}^{(\tau)} - \theta_{t+1}) \cdot f \cdot k \tag{5.1}$$

Here the difference between the current position, θ_{t+1} and the predicted position, P_{t+1} (looking τ timesteps in expectation into the future), is used with the update rate, f, given in Hz, to calculate the velocity needed to achieve that position in one timestep. This value is then scaled by $0 < k < 1$. Unattended joints only move when the user is moving an attended joint and after each joint toggle the system reverts to manual control for 0.5 s during which unattended joints are held still. For clarity, note that a single update is equivalent to one timestep, i.e., for an update rate of 30 Hz there are 30 timesteps per second.

Temporally extended predictions of joint angles (θ) are made using GVFs that are learned using the True Online TD(λ) algorithm (van Seijen and R. S. Sutton, 2014), as defined by three update equations below (shown for a single joint θ).

$$\delta_t = R_{t+1} + \gamma \mathbf{w}_t^{\mathsf{T}} \boldsymbol{\phi}_{t+1} - \mathbf{w}_{t-1}^{\mathsf{T}} \boldsymbol{\phi}_t \tag{5.2}$$

$$\mathbf{e}_t = \gamma\lambda\mathbf{e}_{t-1} + \alpha_t \boldsymbol{\phi}_t - \alpha_t \gamma\lambda[\mathbf{e}_{t-1}^{\mathsf{T}} \boldsymbol{\phi}_t]\boldsymbol{\phi}_t \tag{5.3}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \delta_t \mathbf{e}_t + \alpha_t[\mathbf{w}_{t-1}^{\mathsf{T}} \boldsymbol{\phi}_t - \mathbf{w}_t^{\mathsf{T}} \boldsymbol{\phi}_t]\boldsymbol{\phi}_t \tag{5.4}$$

This algorithm learns a weight vector, $\mathbf{w}$, that is used to make the prediction P_{t+1} about the future of the pseudo-reward signal R, where the ideal prediction is the scaled sum of future rewards $\sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i}$. In our experiments, the cumulant is defined as $R_{t+1} = (1-\gamma)\theta_{t+1}$, where the $(1-\gamma)$ factor is used to select the instantaneous angle at the pseudo-termination of the prediction. The γ term specifies how much weighting is given to future rewards and, in expectation, is related to the number of timesteps (ts) used for prediction by $\gamma = 1 - \frac{1}{\text{timesteps}}$; $\gamma = 1$ looks to infinity and $\gamma = 0$ looks one timestep. At each timestep a feature vector, $\boldsymbol{\phi}$, is used to calculate the temporal-difference error in (5.2). The trace vector in (5.3) assigns credit to various features, with a decay factor λ specifying how far into the past to assign credit ($\lambda = 1$ is fully Monte Carlo, $\lambda = 0$ is full bootstrapping), and α specifying a per-timestep learning rate. Finally, (5.4) updates our weight vector, $\mathbf{w}$. Predictions, P_{t+1}, are made for a given joint, θ, in expectation, looking τ timesteps in the future, using an inner product $P_{t+1}^{(\tau)} = \mathbf{w_{t+1}}^{\mathsf{T}} \boldsymbol{\phi_{t+1}}$. GVFs were initialized to predict the minimum angle values for each joint ($\mathbf{w}[:] = \mathsf{min\_angle/num\_active\_features}$), and a fixed step size was used, $\alpha = 0.3/\mathsf{num\_active\_features}$, and $\lambda = 0.95$. Each predictor used the same fixed-length binary feature representation, $\boldsymbol{\phi}$, as input, consisting of a single bias unit and a 4-dimensional tile-coding (R. S. Sutton and Barto, 1998) of shoulder angle (θ_S), elbow angle (θ_E), and history traces of the same, as shown in (5.5), with a decay rate of 0.99. Angles were normalized over the effective joint range. To balance generalization and accuracy, 100 coarse tilings with width 1 were hashed to a memory size of 2048, for a total feature vector size of 2049 with

101 active features per step.

$$H_{t+1} = (1 - \xi)\theta_{t+1} + \xi H_t \tag{5.5}$$

where

H – the decaying history trace on the normalized joint angle

ξ – decay rate

θ – normalized joint angle

Each experiment consisted of a TPC training phase (i.e., no automated joint control) followed by DPCC, with learning kept on during all phases to allow for continued adaptation and user correction. Experiments were conducted by a non-amputee subject using the Bento Arm (Figure 5.2), a non-compliant robot arm developed in our lab (Dawson et al., 2014). For these experiments, only shoulder rotation and elbow flexion were used, with all other joints held rigid via motor commands. The words "shoulder" and "elbow" were audibly played to inform the user about the outcome of their toggling action. Joint angles were constrained to limit joint action to a range covering the effective workspace of each experiment by a small margin. The Bento Arm was desk mounted and fixed in place. Control software for the Bento Arm and all experiments ran on the Robot Operating System (Quigley et al., 2009). Sensory updates, motor commands and real time learning of the GVFs were all performed at 30 Hz. It is important to note that while the current experiments were limited to only 2 DOC, the methods employed are theoretically applicable to any number of DOC.

5.4 Experiment 1: Navigating Waypoints

The purpose of our first experiment was to investigate the behavior of DPCC during ongoing human-robot interaction. Here the user operated a joystick to move the robot arm through a series of waypoints, while also observing a visualization, as shown in Figure 5.3, which translated the joint space of the shoulder and elbow into horizontal and vertical components.

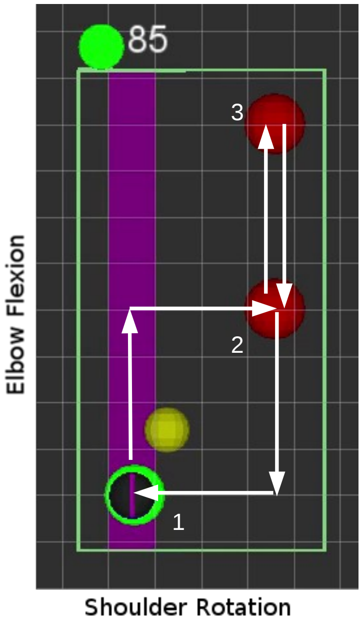
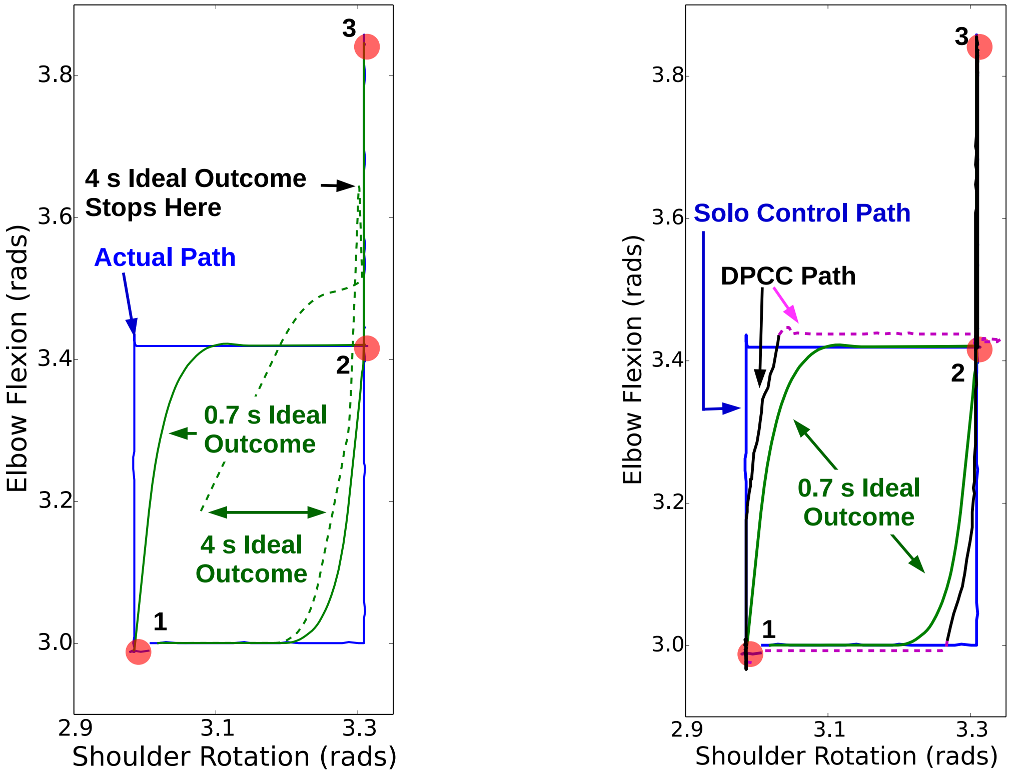


Figure 5.3: real time visualization of the waypoint navigation task. Joint bounds are shown by the green box. The current joint angle position is indicated by the black circle, while the yellow circle indicates the current prediction. Waypoints are indicated by red circles, and when the black circle is within a waypoint it turns bright green indicating the joints are within the specified tolerance. The purple strip indicates the direction the user can move. Circuit number is indicated at the top along with a task-state indicator. The path taken in this task (white arrows) starts at 1, proceeds up and right to 2, then up to 3, down to 2 and finally down and left to 1.

Waypoints were indicated by red markers. When joints were within 0.0175 rads (1 degree) of a waypoint center, the marker would turn green, a sound was played and the user attempted to hold the joints within the marker until a second sound played three seconds later. The circuit started at the lower-left waypoint, 1, moved up and then right to the midpoint, 2, up to the top-right waypoint, 3, down to the midpoint, 2, and finally down and then left to the start point, 1. These waypoints are symbolic for places in joint space where the amputee would complete another task before moving on, such as grasping or releasing an object. In typical operation of a prosthesis, an amputee may not have any conscious recognition of these waypoints and they may change over time. Joystick control mimicked the signals produced by an EMG-based TPC system (i.e., a single joystick provided a scalar signal [-1,1]) and a button press allowed the user to toggle between joints.

Figure 5.4a shows a sample circuit made during training alongside ideal outcomes. Ideal outcomes are the ideal path we would expect to follow when using GVF predictions as



(a) Ideal task outcomes

(b) DPCC performance

Figure 5.4: a) The path of a single training circuit (blue), and predicted ideal outcomes for 0.7 s or 20 timesteps (solid green) and 4 s or 120 timesteps (dashed green). b) Direct Predictive Collaborative Control. Compare a circuit made in training (blue) and its ideal outcome (green) against the path taken during a DPCC circuit (magenta and black). Magenta and black indicate user control of the shoulder and elbow, respectively.

control actions when looking ahead at a specific timescale. In this case, the ideal outcomes are defined to be the ideal predictions made by the GVFs (i.e., the computed temporally extended predictions for the observed data). Outcomes for 0.667 s or 20 timesteps ($\gamma = 1 - \frac{1}{20}$) are shown (solid green) as well as outcomes for 4 s or 120 timesteps ($\gamma = 1 - \frac{1}{120}$, dashed green). We see that these predictions produce a rounding of the upper-left and lower-right corners of the circuit. It is this rounding that we exploit in order to preemptively activate joints and achieve desired joint angles. As would be expected, the rounding of the corners for 4 s predictions is much more pronounced and it looks past the 3 s pause at the waypoints.

Training lasted 29 circuits (≈11 min), followed by 50 circuits of DPCC. Predictions of 0.667 s (20 timesteps) were used, with a scaling factor k of 0.5/#timesteps = 0.025.

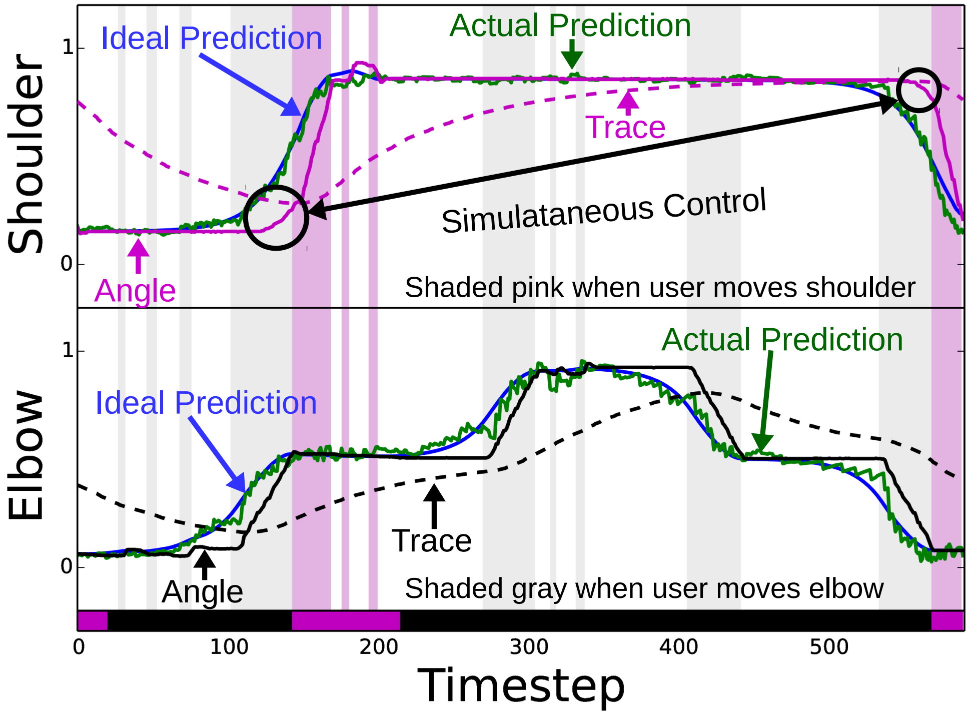


Figure 5.5: Simultaneous control of multiple prosthetic joints though a single control channel is made possible by our proposed automation. *Shoulder*: normalized angle (solid magenta), normalized angle trace (dashed magenta), ideal outcome (blue), actual prediction (green). *Elbow*: normalized angle (solid black), normalized angle trace (dashed black), ideal outcome (blue), actual prediction (green). Pink shading indicates the user was actively moving the shoulder, and gray shading indicates the user was actively moving the elbow. The strip at the bottom indicates the user had selected the shoulder (magenta) or elbow (black). Areas of simultaneous joint activation are shown inside the black circles. It may be helpful to note that a positive movement in the shoulder corresponds to a movement to the right in joint space.

Figure 5.4b compares the path taken in the final DPCC circuit against a typical circuit taken from the training set. We can see that the DPCC follows a similar path to the ideal outcome for the training circuit, albeit not as pronounced. Figure 5.5 shows the temporal behavior of the final DPCC circuit. In the gray shaded regions of this example the shoulder was moved by the system while the user controlled the elbow, as highlighted by the change in the solid magenta line in each of the two black circles. In the first circle we clearly see that the shoulder joint angle (solid magenta) is increasing (moving right) as the elbow joint (solid black) increases (moves upward). In the second circle we see that shoulder joint angle is decreasing (moving downwards) as the elbow joint decreases (moves left).

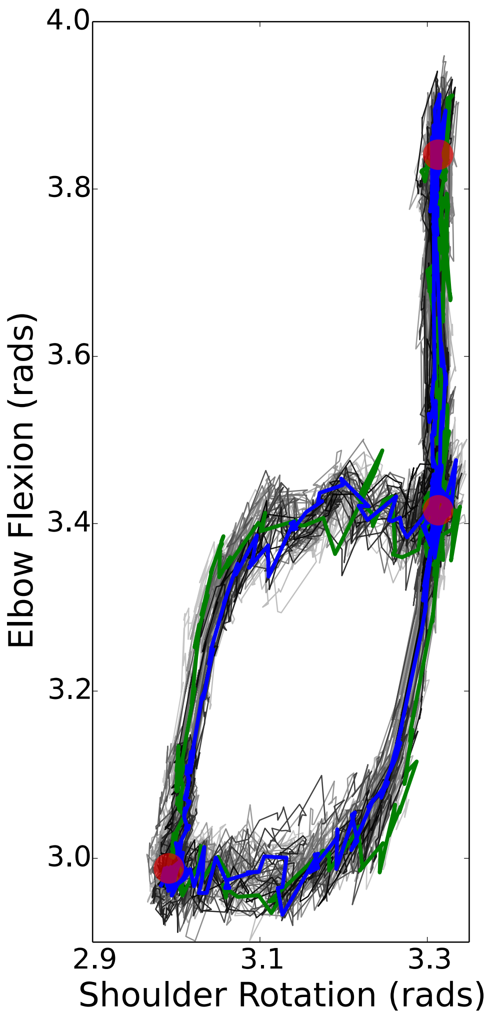


Figure 5.6: Predictions made over all DPCC circuits. The first circuit is shown in green and the last is shown in blue. Circuits in between are scaled light to dark, with earlier circuits being lighter.

These results demonstrate that our approach was able to learn target angles for each joint without explicitly providing them to the system. Simple, but potentially beneficial, joint synergies were learned in real time purely by observing ongoing user behavior, and effected by a direct mapping of predictions to control commands for the unattended joint.

As has been stated, learning was ongoing for all joints during the entire experiment, regardless of whether the human or the robot controlled the joints. As the arm follows the predictions, those predictions are in turn being updated to reflect the new trajectory. This creates a positive feedback loop. This feedback is tempered by the use of the variable k in (5.1). Figure 5.6 shows the predictions made for all the collaborative control circuits and

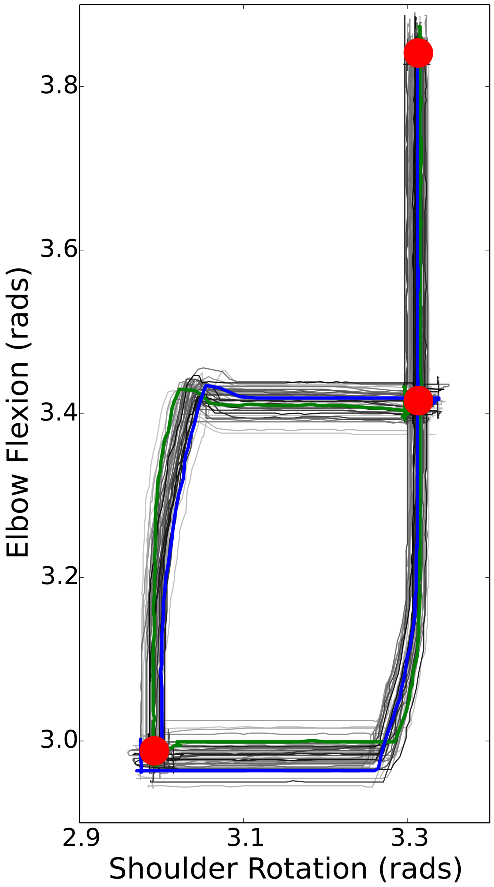


Figure 5.7: Paths taken over all DPCC circuits. The first circuit is shown in green and the last is shown in blue. Circuits in between are scaled light to dark, with earlier circuits being lighter.

Figure 5.7 shows the trajectories taken over those same circuits. The first circuit is indicated in green and the final circuit is indicated in blue. The in-between circuits are shown in shades of gray, with lighter circuits being earlier and darker circuits being later. From these, we can see that despite the positive feedback loop, the resulting prediction paths and trajectories are generally stable. This is caused by the interaction between k in (5.1) and the step size α in (5.3); when using a small k value the velocity calculation takes only a fraction of the step towards the ideal outcome, at the same time, a small α means that we only move our predictions a small step towards the new trajectories. While it cannot be said that the path has converged, the trajectory appears stable within the life of the current experiment.

Further, as can be seen in Figure 5.6, the predictions made for the elbow are noisy. The use of the scaling factor k helps reduce the effects of this noise. That is not to say that k is the best way to deal with the noise, but simply an observation on the current experiment. In future experiments it would be preferable to first improve the predictions and then smooth the resulting commands using a windowed average.

5.5 Experiment 2: Navigating an Angle Maze

As one example of the need for simultaneous multi-joint prosthetic motion, amputees operating one joint at a time must use compensatory body motions to create diagonal (off-joint-axis) movements.

We expect diagonal (off-joint-axis) movements and related motions should be straightforward to perform using DPCC. We conducted a limited test of this hypothesis by navigating an angled portion of a wire maze with the robot arm. The user controlled the arm using proportional EMG signals (TPC). A metal rod was attached to the gripper of the Bento Arm as shown in Figure 5.2. A green barrier marked the start of a circuit and a yellow barrier marked the turn-around point. Contact with each barrier was detected by electrical connection between the rod and a 2-cm-long exposed metal region at the center of each barrier. A circuit consisted of contacting the green barrier, moving to the yellow barrier, then returning to the green barrier. The system played a sound when either barrier was contacted. The user would then try to hold position on the exposed portion of the barrier until a second sound was played 5 s later. The user was to avoid contacting the walls, but they was not penalized in any way for doing so. The proportional signal was generated using EMG recorded from two sites on the user's forearms (each thresholded and normalized) at 200 Hz, calculating the mean absolute value of those signals over a 10-sample sliding window and then taking the difference. The toggle signal was produced using a third EMG signal on the user's other forearm. User controlled joint speed was limited to 0.2 rad/s for ease of control, while no limitation was placed on those controlled by automation. This task was designed to reflect real-life precision movement tasks in a constrained environment, and was inspired by a similar challenge in an upcoming competition for parathletes (Cybathlon 2016).

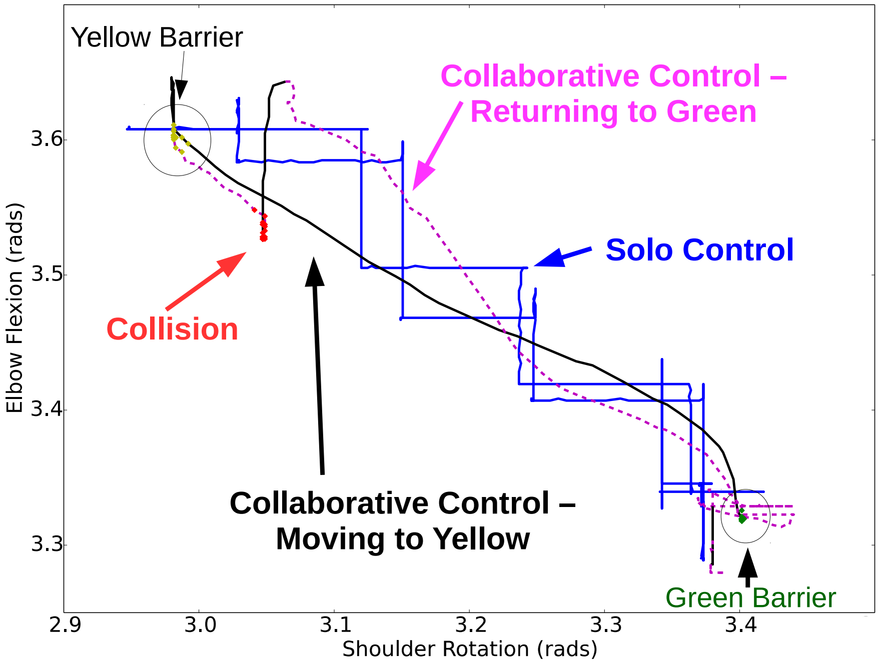


Figure 5.8: Key result: Comparison of solo and collaborative control through the angle maze. While not perfect, we see clearly that DPCC enables the user to achieve an angled trajectory not possible for a user on their own.

TPC training lasted for 30 circuits (≈16 min), followed by 53 circuits of DPCC. As expected, when the user controlled the arm alone, the path was noticeably stepped as shown in Figure 5.8 (blue). However, with DPCC based on predictions made for 20 timesteps, or 0.667 s ($\gamma = 1 - \frac{1}{20}$) in the future, the achieved trajectory was considerably smoothed due to the learned, simultaneous joint actuation (dashed magenta indicates user was controlling the shoulder, and solid black indicates user was controlling the elbow). A value of 0.1 was used for k. This figure overlays the 46th circuit of DPCC, a particularly good circuit, where we see the path of the rod move along at an angle without the stepped behavior characteristic of TPC. One point of contact with the walls occurs on the return portion of the circuit near the lower bend in the maze near the yellow barrier. The user corrected by switching to elbow, moving up and then switching back to shoulder and moving the rest of the way to the right. While angled movement was evident in most DPCC circuits, not all had so few collisions with the bounds. To accommodate the resulting slight shifts to the maze during operation, the lines in Figure 5.8 denoting the DPCC circuit were registered to the maze starting point for accurate visual comparison.

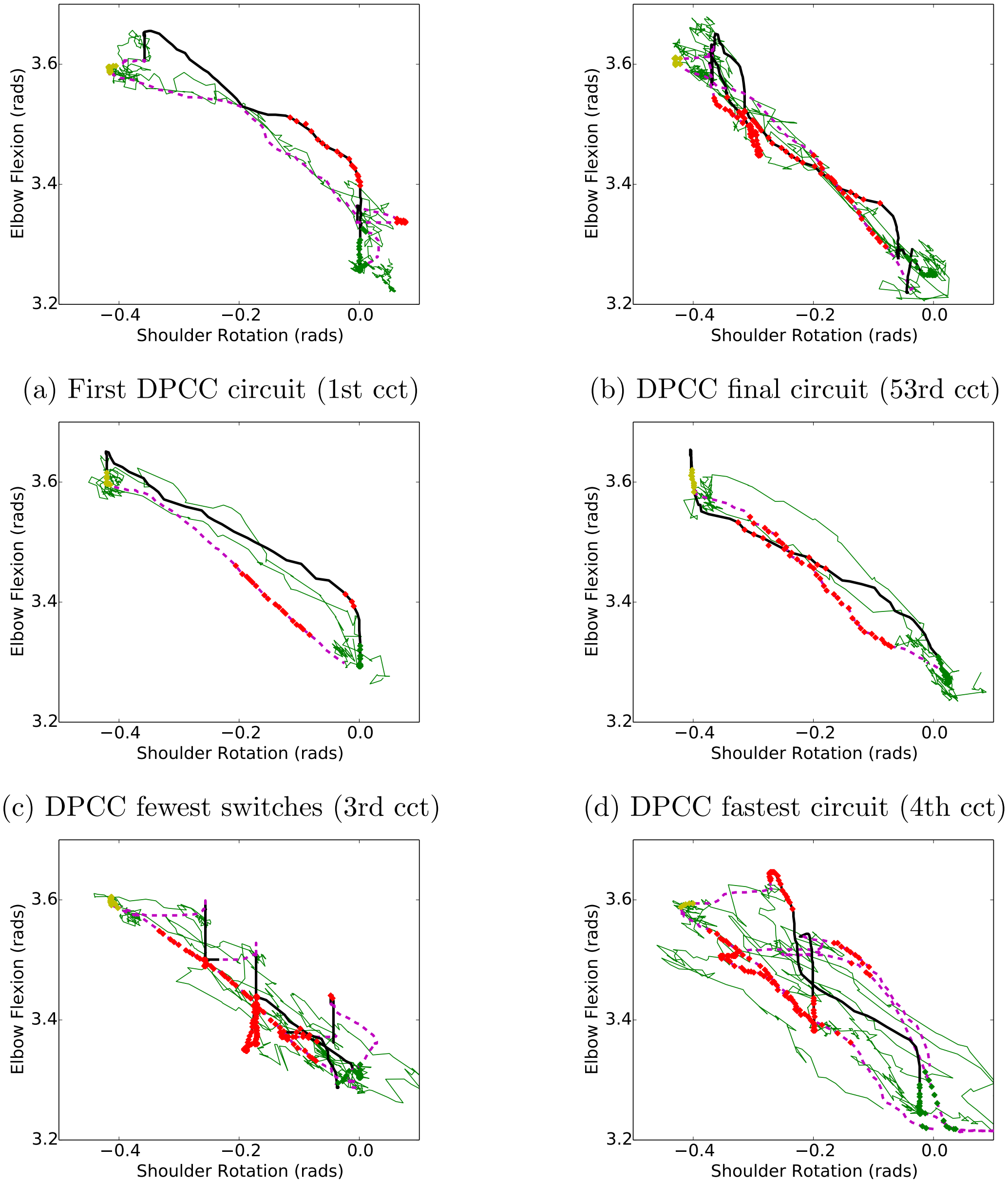
Table 5.1: Results for the angle maze experiment

Control	Toggle Count				Task Time (s)			
	AVG	STDEV	Min	Max	AVG	STDEV	Fastest	Slowest
Training (TPC)	15.1	3.85	10	22	32.3	8.13	21.9	54.7
DPCC	7.83	4.07	1	19	26.3	6.14	14.9	40.8
Theoretical Best	-	-	0	-	-	-	12.8	-

Table 5.1 compares performance between TPC and DPCC phases. Ideally a circuit could be completed with no joint toggles in 12.8 s. In our experiments the user's speed is limited, but the automation speed is not, therefore the fastest time will occur when the user controls the joint that must move the shortest distance. The shortest distance is moving the shoulder 0.28 rad up and down at a max speed of 0.2 rad/s giving a value of 2.8 s travel time plus 10 s of waiting. The fastest time achieved by DPCC was 14.9 s with a single toggle. DPCC performance showed significant improvement over solo control, with average circuit time falling by 19% from 32.3 s to 26.3 s, and average toggle counts falling nearly 50% from 15.1 to 7.83. Improvement was not due to human learning; the user had prior experience with the system and task.

Figure 5.9 shows several circuits made during DPCC, where we see that angled trajectories were generally achieved. However, we can also see that the behavior is still suboptimal; many of the circuits required numerous corrections, predictions could vary wildly, and contact with the maze was a common occurrence. Recall that avoiding contact with the maze was not a part of the task design, however, it should be considered for future experiments. Figure 5.10 shows temporal representations of the circuit with the fewest switches (see Figure 5.9d) and the circuit with the most switches (see Figure 5.9e) respectively. These figures are useful for understanding the behavior of the system. However, they are quite complex, so we will describe them in detail in the following paragraph.

Each of the different plots is a representation in time, sharing the same axis (i.e., taking a vertical line through all of them indexes the same point in time). The top two plots are for the shoulder joint, with the top-most showing the TD error for the shoulder GVF. The second plot shows the normalized shoulder angle (solid magenta) and the trace on the normalized shoulder angle (dashed magenta), along with the predicted shoulder angle at a



(a) First DPCC circuit (1st cct)

(b) DPCC final circuit (53rd cct)

(c) DPCC fewest switches (3rd cct)

(d) DPCC fastest circuit (4th cct)

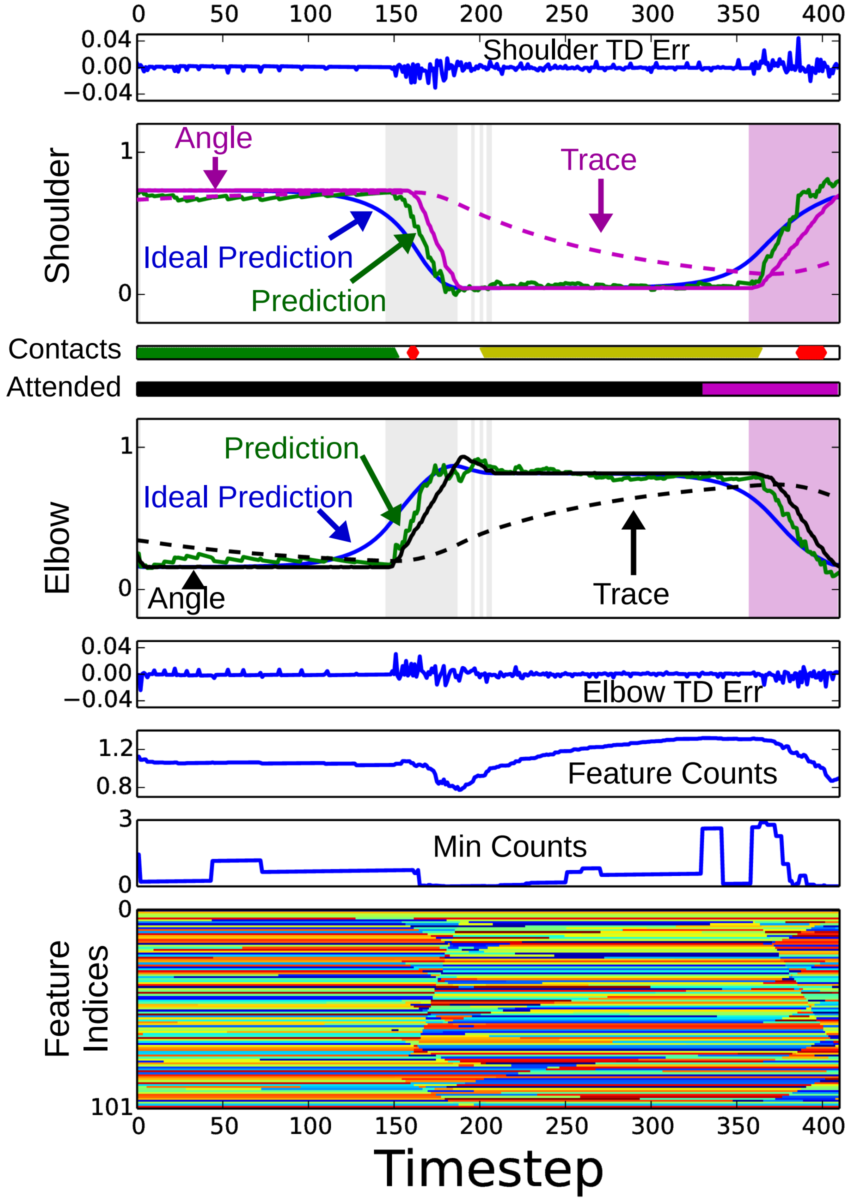
(e) DPCC most switches (11th cct)

(f) DPCC slowest circuit (34th cct)

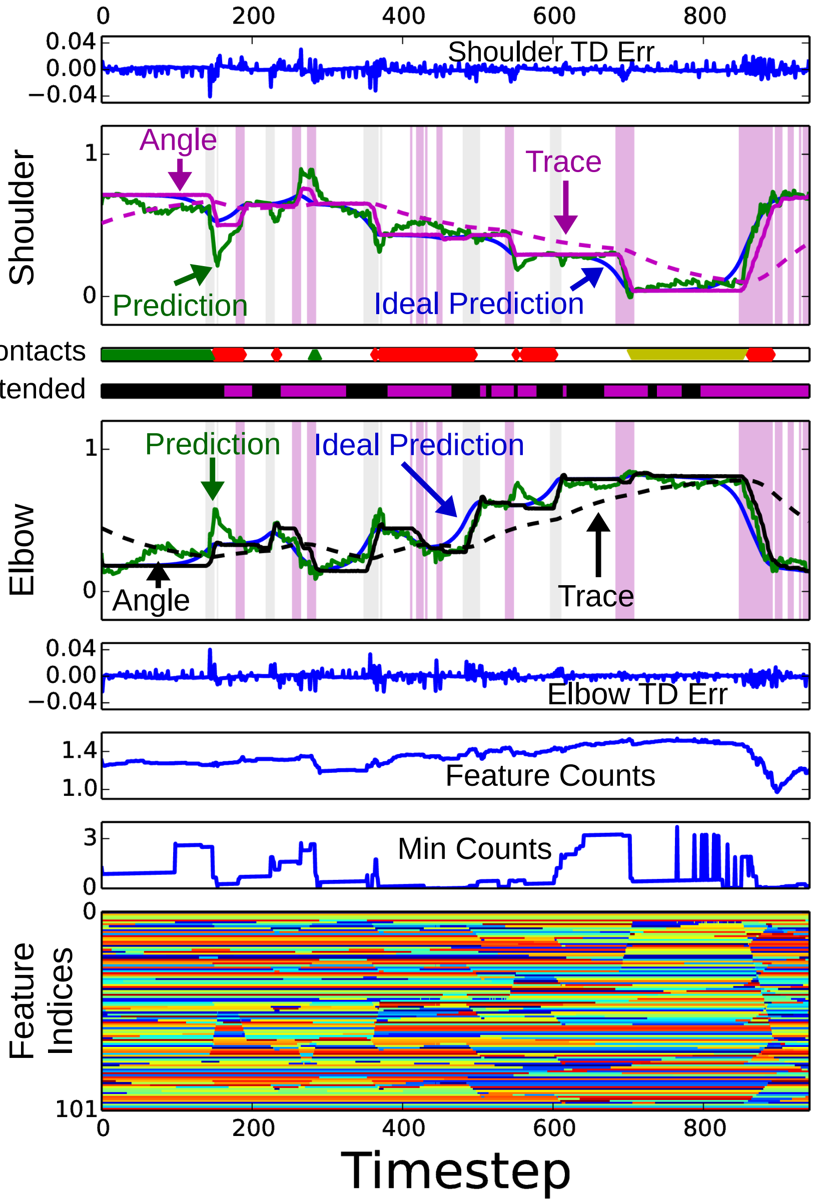
Figure 5.9: DPCC – several circuits. Circuit trajectories are shown in solid black and dashed magenta. Black is used when the user is in control of the elbow and magenta indicates the user was controlling the shoulder. Contact with the green barrier is indicated by the green dots, and yellow dots show contact with the yellow barrier. Red dots show contact with maze boundary. Predictions are shown in green. Note that the x-axis has been registered such that all the circuits start at 0.0 rads.

timescale of 0.7 s or 20 ts. Recall that the input to the tile coding is the normalized shoulder and elbow angles and traces on both. The shaded regions indicate that the user was actively moving either the shoulder joint (magenta) or the elbow joint (gray) at that time. It may be helpful to note that a positive motion corresponds to moving the arm to the right. The next plot, *Contacts*, indicates the times when the metal rod was in contact with the green barrier (green), yellow barrier (yellow), or the maze itself (red). The plot below this, *Attended*, shows which joint the user was attending (i.e., which joint was selected for control by the user) with elbow indicated in black and shoulder indicated in magenta. The next two plots are for the elbow. The first shows the normalized elbow angle (solid black), the trace of the normalized elbow angle (dashed black), and the predicted elbow angle at a timescale of 0.7 s or 20 ts. Again, the shaded regions indicate that the user was actively moving the shoulder joint (magenta) or the elbow joint (gray). The next plot indicates the TD error for the elbow GVF. The final three plots display information about the binary features used. In the first of these, *Feature Counts*, we see a plot of the feature counts. This is an implementation of the *naive* approach to state counting described in Section 6.1. Essentially we keep a count of how many times each feature has been observed. For a given timestep we then simply add all the counts from the currently active features, excluding the bias unit. The numbers given are in millions. The second plot, *Min Counts*, gives the smallest count of all the active features in thousands. If this value is very low, say around zero, then the current state is novel for at least one of the tilings. The final plot, *Feature Indices*, provides a visual representation of the values of the feature indices at each timestep. There are 101 rows corresponding to the 100 tilings produced by tilecoding and the single bias unit. Each of these tilings can have a value from 0 to 2049, which is indicated by the color. While this plot does not allow us to make a detailed analysis it does provide a way to observe changes in the representation.

Figure 5.10a shows the temporal representation of the DPCC circuit with the *fewest* switches, while Figure 5.9c shows the same circuit in joint space. In joint space the predictions seem to plot a reasonable path through the maze. However, there is noise around the barriers. This noise can also be seen in the temporal representation in the shoulder and elbow feature plots (around 0–150 ts and 180–360 ts). This noise is caused by the angle traces used to produce the feature representation. These traces continue to decay causing



(a) Temporal view of circuit with fewest switches (see Figure 5.9c)



(b) Temporal view of maze circuit with the most switches (see Figure 5.9e)

Figure 5.10: Temporal circuit view. From Top to Bottom: *Shoulder TD Err*, *Shoulder Features* — shoulder angle (solid magenta), shoulder angle trace (dashed magenta), ideal prediction (blue), actual prediction (green) with shaded areas indicating the user was actively moving the shoulder (pink) or elbow (gray), *Contacts* — rod contacts with the green barrier (green), maze (red), and yellow barrier (yellow), *Attended* — the joint currently selected by the user, black for elbow and magenta for shoulder, *Elbow Features* — elbow angle (solid black), elbow angle trace (dashed black), ideal prediction (blue), actual prediction (green), with shaded areas the same as for the shoulder features, *Elbow TD error*, *Feature Counts* — a summation of all the times the current features have been observed, in the millions, *Min Counts* — for each observation, the smallest feature count of all active features, *Feature Indices* — an image representing the current index of each of the 101 binary features. See Section 5.5 for detailed explanation.

changes in the state space, which can be observed by changes in the *Feature Indices* plot. In the feature plots we also see that the system moved its joints to track the predictions quite well. This can be seen in the shoulder feature plot in the shaded area around 150 ts and in the elbow feature plot in the shaded area around 360 ts. That being said, tracking these predictions often caused the rod to skim along the lower rail during the return portion of the circuit. Using a smaller timescale may improve behavior here by pursuing more gentle predictions. While the GVFs are indeed predicting, they are not predicting as early as the ideal predictions indicate that they should and do not appear to predict movement until the user has actually begun moving. This evidence suggests that the feature representation used is insufficient to differentiate these states.

Figure 5.10b shows the temporal representation of the DPCC circuit with the *most* switches, while Figure 5.9e shows the same circuit in joint space. This circuit had many collisions with the maze when moving from the green barrier to the yellow one, but only skimmed along the maze on the return trip. Note that as the circuit starts out the traces for both joints are much further from converging on the resting point than they were in Figure 5.10a. This appears to have a significant effect on the early predictions. Right away this results in a collision with the inner portion of the maze around 180 ts. As the user corrects this action the arm is returned to the starting point (around 180 ts) as the system now thinks that the user is moving from the left towards the green barrier. Throughout the trip from the green barrier to the yellow barrier the user makes many fast switches. Recall that after a switch there is a 0.5 s or 15 ts period during which the system returns to manual mode before reengaging DPCC. Often we can see that the user made a switch and began moving their joint before DPCC had turned back on. This is the expected way to make corrections. However, the user may have been expecting DPCC to reactivate sooner than it did as we see three spikes in Figure 5.9e where the user appears to have driven the arm straight up into the top of the maze, as if they were anticipating the system moving the arm to the left. Much of the prediction error observed in this circuit can be accounted for by the fact that the system has simply never seen a circuit that looked quite like this before. This can be seen by the many regions of low *Min Count*. The behavior of the predictions is discussed further in Section 5.6.1.

It was common to see the rod slide along the lower bound of the maze on the return portion of the circuit. The Bento Arm is not compliant, however, there is a bit of play in the linkages and in the mounting of the rod. If the rod were to press hard against a barrier the rod would deflect and we would see points of contact with the maze (red dots) with deviation from the maze bounds, like the downward spike shown in Figure 5.9e and the return trajectory in Figure 5.9d. On the other hand, if the rod just brushes the maze then we would see points of contact, but a trajectory which continues to follow the shape of the maze. Often, the touch was very light as in the return portions of the circuits shown in Figures 5.9b, 5.9c, and 5.9e. As we have seen in the temporal plots in Figure 5.10, DPCC control of joints tended to do well in following the predictions. This suggests that if the predictions had been less aggressive collisions may have been reduced. This could be accomplished by using a shorter timescale (i.e., smaller γ).

5.6 Discussion and Future Work

Several studies have shown users are willing to accept a degree of automation from assistive devices. As examples, one study examined the use of intelligent wheelchairs (Viswanathan et al., 2014) and another examined able-bodied use of prosthetic hands (Cipriani, Zaccone, et al., 2008) in a shared control system. However, it is not clear to what degree actual amputees will accept automatic control of prosthetic movements. Amputees have an intimate relationship with their artificial limbs; arm motions not felt to be self-initiated by the user may be difficult to accept, or may undermine the illusion of ownership created by a user with respect to their prosthesis. It is important that these interactions between automation and ownership be investigated and that potential methods be compared in terms of effectiveness and amputee acceptance.

5.6.1 Improved Predictions

The predictions plotted for both the waypoint experiments (see Figure 5.6) and the maze experiments (see Figures 5.9 and 5.10) clearly show us that predictions were noisy and at times quite wrong. Improving these predictions is an important next step in evaluating the use of predictions in controlling a prosthetic arm. Predictive accuracy is dependent on the featurization of the sensor data. The predictions shown for the waypoint experiments (Figure 5.6) show considerable noise in the elbow, but not much in the shoulder. This is perhaps not surprising as the elbow undergoes a more complex trajectory. The elbow predictions appear to jump up and down, which likely corresponds to switches in tile coding bins. Further, the predictions for the maze experiment (Figure 5.9) seem rather consistent except for at the barriers or when the trajectory undergoes significant deviation from the norm as in 5.9e and Figure 5.9f.

A feature representation allows a predictor to capture a sense of the state of the world. In most real-world domains it is impossible to accurately capture all the states and we therefore rely on function approximation to give us a coarser view of the world. On one hand this is a loss of information, producing conflation of states where the representation used cannot distinguish between one or more states. This can certainly be seen as one of the causes for predictive error in the current implementation of DPCC. However, on the other hand, this coarseness prevents overfitting and creates the ability to generalize, which enables an agent to learn much more rapidly than if it had to treat each new sample as being a completely new state. Additionally, this generalization smooths out predictions and allows the GVF to capture the *gist* of things.

A further source of predictive error is simply that when the observed representation is new or somewhat new there is still learning to be done. This can be observed in the dips in the *Feature Counts* and *Min Counts* plots of Figure 5.10.

The use of angle traces in the representation used by these experiments is one cause for much of the predictive error. These traces are meant to encode both joint speed and a sense of where the joint is coming from. However, they are a crude way to do so; for this particular task what we really want is to know which barrier the user is moving towards. However, we do not want to explicitly specify this information for the learner as this would only solve for this specific task and not allow the system to be a general solver. On the other hand, if the system had a way of building its own representations capable of expressing such task specific features, this would greatly improve its abilities. At this time the automatic construction of such features is still an open research question.

The use of a single trace for each joint, as is done here, means that there are many ways to arrive at the same position; a given position in joint angle space may correspond to many different states. If those states have not been trained before, then we expect their predictions to be very inaccurate.

As we have discussed, the GVFs are only able to make accurate predictions if they know what state they are in. In the case of the circuit with the fewest switches, Figure 5.10a, we observed that at the end of the rest time on each barrier, the traces had converged or nearly converged to the target signal and thus offered no new information to discriminate state. Thus, the predictors were not able to anticipate movement until the user had begun to move.

Finally, the representation used in these experiments consisted of 100 coarse tilings. Coarse tilings provide broad generalizations which smooth out the predictions, removing a good deal of spiking behavior. By including many such tilings we maintain good generalization while providing resolution to discriminate many different states. However, having higher resolution encodings, at least in specific regions of the state space may be beneficial to the accuracy of the system.

Identifying improved feature representations for use with controlling a prosthetic arm is a necessary step for future research.

5.6.2 Reinforcement and Learning

Assuming that some form of blended autonomy is beneficial, there are many ways that one might approach multi-joint coordination during the use of a robotic arm. The DPCC approach is a fairly straightforward one—given predictions about where a joint angle will be in the future we simply move toward that angle. One potential benefit of the DPCC method, as compared to others we might imagine, is that the user operates the arm in the same way regardless of whether or not they receive automation assistance. Further, by keeping learning on during all phases of operation the system is able to continually update predictions based on user behavior and adapt to user correction. It effectively demonstrates that *use can be its own form of reinforcement*. However, it should be noted that the present approach has the potential to reinforce both good and bad behavior, as has been observed in biological learning (Jang, 2013).

In terms of limitations, DPCC makes the assumption that it is beneficial to move towards where the system predicts the user will be, with predictions being made at a single level of temporal abstraction (i.e., one time scale). We can easily identify situations where a single time scale is not the best basis for control. A chosen time scale may be a strong choice for some aspect of a task, as in the diagonal portion of the angle maze experiment. However, looking a fixed distance into the future can also lead to collisions with objects from cutting a corner (we see this in Figure 5.9), or bypassing important locations in space, as was demonstrated in the 4 s predictions in our first experiment (see Figure 5.4a). One potential solution is to choose the predictive distance based on some aspect of the system's state, allowing us to look anywhere from immediate to remote predictions. This state-dependent behavior seems desirable in general. However, it is non-trivial to select or learn the correct temporal distance to use in each state.

While typical RL methods involve the more complex process of learning a policy based on a reward signal, we have first explored what can be achieved with the simpler direct method described here. In our scheme, system and user behavior reinforce the predictions and behavior without the use of a reward signal. However, a reward signal would open up additional options for control improvement, one of which might be a mechanism for learning how far in the future to look at any given instant. Reinforcement might be used to select from a list of predictors at different timescales, to blend predictions from multiple timescales, or to select a state-dependent γ value for a single GVF. One reward signal already present in the system is the toggle event. A reinforcement learning algorithm could seek to minimize the number of toggle signals required from the user to complete a task. This is an interesting area for future work.

5.6.3 Confidence

A measure of confidence in the system's ability to behave appropriately would address many issues in the current work. Confidence might consist of several measures including: accuracy of recent predictions (A. White et al., 2014), how often the system has seen the current state before, accuracy of past predictions when in the current state, predictive convergence (M. White and A. White, 2010), risk level of the current situation, and a measure of the user's confidence in the system.

First, confidence could remove the need for the user to explicitly turn DPCC on and off. If the system gave a high enough confidence value then it could gradually engage in DPCC; if confidence drops it would return to TPC.

Second, being able to automatically move in and out of DPCC could further improve the system when it encounters previously unseen states. While the trajectories shown by the ideal prediction lines in Figure5.4a appear straightforward for us to follow, in reality, once the system begins to follow these trajectories it moves the arm out of state spaces it has seen before and its predictions suffer as a result. Some of this can be accounted for by choosing a representation that generalizes well, which is the reason why many very coarse tilings were used in these experiments. However, this is not always sufficient. When using confidence measures, the system could disengage from DPCC when it encounters a new state and simply watch the user in order to build up its predictive certainty until its confidence is restored.

5.6.4 Time and State-Space

The results achieved here should be reproducible on any commercial prostheses that provides joint angle feedback. This is possible because, unlike many alternative approaches, GVFs allow us to effectively use time as a signal. The temporal nature of GVF predictions enables automation to recognize and leverage patterns of usage, adapt to changing conditions, and capture how joints are temporally related to one another. It seems appropriate to design control systems that do not waste the user's time. While our results do not depend on modifying existing hardware, improvements should be possible by increasing the amount of information available to the machine learner from the user, from the system, and from the environment. Our methods are ideally suited to efficient implementation in the face of increasing state-space size, being linear in computation and memory (something of great importance for learning on devices intended for wearable operation).

5.6.5 Predicting EMG Signals Instead of Joint Angles

The control schemes described in Section 5.3 assumed that predicting the output of user interaction, that is the joint angles, was the right approach. Alternatively, one could instead predict the expected EMG signals from the user. This has the benefit of using the same mapping from input to output as the user's own signals. This may in fact be a viable approach to take and warrants comparison against the DPCC approach taken here. However, in the broader context of robotic control, in which we are interested, it should be clear that such an approach is tied to the problem of prosthetic control with EMG input and would not lend itself well to other teaching approaches which may be desirable. Such alternatives could include kinesthetic teaching, where the user physically moves the arm through the desired trajectory, or training through pre-programmed trajectories, or by observing movement demonstrations. In short, predicting the outputs rather than inputs allows us to develop systems which are much more flexible with broader applications.

5.6.6 Disorientation and Feedback

Anecdotally, the operator for the experiments here reported that during collaborative control they were at times confused about which joint they were in control of. This occurred when the user and the arm were working together and the arm was following the user's expected trajectory, but then the user wanted to make a correction. At this point the operator reported often not being sure about which joint they were controlling since they could no longer depend on visual feedback to inform them. As a result, the user would often initiate joint toggles that were unnecessary. These extra toggles are reflected in the Table 5.1. Ongoing feedback, as opposed to just providing feedback when a joint toggle is made, may provide a way to correct this and further reduce the number of toggles required to complete the task. This sense of disorientation echoes Parasuraman et al. (2000), “Humans tend to be less aware of changes in environmental or system states when those changes are under the control of another agent (whether that agent is automation or another human) than when they make the changes themselves”. Improving feedback for the prosthetic user may remedy this, and is an ongoing area of research.

5.7 Conclusion

We have described and tested a new collaborative control approach, denoted Direct Predictive Collaborative Control, whereby a user, limited to controlling a single joint at a time, can effect multi-joint synergies in conjunction with an intelligent prosthetic control system. The control system used here *learns predictions directly from user action*, without the need for predefined target angles to be specified, and maps these predictions directly into command signals. Unlike many other approaches, the system’s *predictions can be learned in real time* during ongoing, uninterrupted use of the device. Our preliminary results from a single able-bodied participant on a simple angle-maze experiment demonstrate that our methods enable coordinated multi-joint movements and are able to improve task performance by reducing the number of switches and the time needed to complete the task. While our approach was demonstrated for the control of a prosthetic arm, the same approach should be applicable to many domains involving assistive devices where the numbers of functions exceed a user’s ability to attend to them. To our knowledge, the present study is also the first demonstration of the combined use of the True Online TD(λ) with general value functions for online control. While larger user studies are needed to confirm the improvements seen here, these results support the continued exploration of our approach.

Chapter 6

Extensions

While there are numerous extensions proposed in each of the chapters, here I focus on those which either have not yet been discussed or are deemed to be most significant. I propose four extensions to the work already described in this thesis. The first discusses how the Direct Predictive Collaborative Control method described in Chapter 5 could be further improved if a measure of the GVF's predictive confidence were available. This section looks at several existing confidence methods and proposes an additional one, which might be effective. The second extension looks at how traditional pattern recognition, as described in Chapter 3 might be combined with the GVF based predictive approaches used in Chapters 4 and 5 in order to improve classifications during real-world use by learning to recognize patterns of behavior and incorporate additional sensory streams. The third extension examines how the DPCC method could be made applicable beyond the limited tasks already demonstrate by using state-dependent predictions of varying timescales and proposes several methods by which this might be accomplished. Finally, the fourth extension discusses alternative ways of implementing collaborative control on a prosthetic arm.

6.1 Confidence

The *Direct Predictive Collaborative Control* method, as implemented in Section 5, is either on or off, which creates certain behavior complications. The GVFs predict certain trajectories through the state-space. However, as soon as the arm starts to follow these trajectories it moves into regions of state-space it has never seen before. This reduces predictive accuracy and can result in undesired trajectories, negatively impacting task performance. The user must then actively correct the system until predictions are sufficiently improved. I have proposed, in Section 5.6.3, that having a measure of a GVF's certainty or *confidence* in the GVF's predictions would provide a way to smoothly handle situations where previously unseen states are encountered, or where the user's behavior deviates from that predicted. If confidence were to fall below a certain threshold DPCC could simply disengage control and observe the user's behavior until confidence once again exceeds this threshold.

The use of confidence measures in machine learning techniques is not new. Many classification methods inherently provide a measure of likelihood that a given sample is of a particular class, which is then used to make the classification decision. This can be thought of as one form of a confidence measure and can be used in control decisions. For example, in pattern recognition with EMG, L. J. Hargrove, E. J. Scheme, et al. (2010) found that taking no action is better than taking the wrong action. In their system a confidence threshold was placed on each class in the classification scheme. If the threshold was exceeded then the class was selected as the output. If multiple classes were selected then no action was taken. This was shown to reduce classification accuracy, but improve actual prosthetic usage.

While there have been a number of approaches to identify confidence measures for GVFs, at present there is no singular definition which is entirely satisfactory. This should not be seen as a failure, but rather an indicator that the definition of such a measure is not clear. Indeed, there are several aspects one might consider in a confidence measure for GVFs:

1. How accurate have predictions been lately?

2. Have the predictions of the GVF converged?

3. How often has the current state been visited?

4. How accurate were past predictions when in the current state?

5. How long has it been since this state was lasted visited?

These aspects include both time-dependent considerations, or *recency*, and state-dependent ones. Explorations of the first two have been performed as follows.

How accurate have predictions been lately? A. White et al. (2014) developed a measure, Z, shown in (6.1), which uses an exponentially weighted average of the TD error, $\bar{\delta}$, to produce what is essentially a trace on the predictive error of a given GVF. If this measure is high then it is said that the agent experiences “unexpected prediction error” or *surprise*. This could also be thought of as a measure of how well the predictions have been lately. If surprise is low then our recent experiences have matched our predictions and confidence is thus higher. On the other hand, if surprise is high then the agent has encountered something unexpected and is less confident or more uncertain. As it stands, the approach taken is largely independent of state. However, modifying (6.1) to use a state-based measure of variance may provide the desired state dependence. It should be noted that the concept of surprise is closely related to the study of anomaly detection.

$$Z_t = \frac{\bar{\delta}}{\sqrt{var[\delta]}} \tag{6.1}$$

Have the predictions of the GVF converged? M. White and A. White (2010), proposed a method for measuring confidence of GVF predictions using interval domains. Their method involves keeping a memory of the last n weight vectors. These weight vectors, along with the current state vector (note that their paper uses state-action vectors, but it is just as relevant for state vectors), are then used to make predictions. This set of predictions is then statistically evaluated to calculate the confidence interval, essentially calculating how much the predictor has converged over the given sample memory. While this approach takes into account recency and state-dependence, it assumes that the variation in the weight vectors over the window is indicative of the convergence of the weights for the current measured state, i.e., the method assumes samples are dependent and that there is a reasonable amount of crossover of features in the representation used. Both assumptions are reasonable in the function approximation case for real-world applications. However, one limitation of the method is that it does not account for history of predictions beyond the window. In the implementation presented this information is lost. A further limitation of this method is that it does not indicate whether or not what the GVF has converged to is good. Finally, this method is computationally expensive and requires a large amount of memory.

6.1.1 Proposed Confidence Estimator

Beyond the two approaches discussed it would be beneficial to more explicitly incorporate measures of confidence that are state dependent, capture past history, and are computationally efficient. A first approach is to simply look at state visitation. If an agent has never seen the current state before, then we should expect its confidence to be low in that state. As the state is visited more we could expect the confidence to grow. This approach is fairly straightforward in the tabular case. However, when using function approximation, as I have throughout this thesis, instead of counting states, we count features. Some measure of the visitations could then be made using an inner product between the feature counts vector and the feature vector itself. However, care must be taken in the counting so as not to over count features which do not contribute to state discrimination, such as the bias feature or other static or slowly changing features. An appropriate normalization may be sufficient to address this.

Simply counting state or feature visitations is a naive approach; what we really want is to know how well our past predictions have been when in the current state, tempered by how often the state has been observed. Gehring and Precup (2013) evaluated an approach to "smart exploration" by using what amounts to a state based trace on the absolute TD error to produce a measure of controllability. Here they defined states to be more controllable if they are more predictable, which corresponds to lower absolute TD error. I propose that using the absolute TD error in a similar way could be used to produce a confidence measure which captures state dependency and many of the other desirable aspects already discussed. Equation (6.2) shows a proposed method for tracking the uncertainty of a GVF, as measured by the absolute TD error, using gradient descent and a linear function approximator with step size $0 < \eta < 1$ and weight vector $\zeta \in \mathbb{R}^{\mathrm{n}}$, which is the same length as the feature vector $\boldsymbol{\phi} \in \mathbb{R}^{\mathrm{n}}$ used by the GVF. Note that we could also choose to use δ^2 as the target of our estimate rather than the absolute value. Additionally, (6.2) can be produced using a GVF with $\gamma = 0$. Uncertainty is then calculated using an inner product as shown in (6.3).

$$\boldsymbol{\zeta}_{t+1} = \boldsymbol{\zeta}_t + \eta(|\delta| - \boldsymbol{\zeta}_t^\top \boldsymbol{\phi}_t)\boldsymbol{\phi}_t \tag{6.2}$$

$$U = \boldsymbol{\zeta}^\top \boldsymbol{\phi} \tag{6.3}$$

When the GVF encounters a state for the first time its absolute TD error will be high and as it improves its predictions the error will decrease. In practice this error will never reduce to zero as our feature representation is unlikely to discriminate all states or because of the use of fixed step size. If the range of the target signal being predicted is limited then we have a practical normalizer on the TD error. In the case of the DPCC experiments, the joint angle is bounded by a minimum and maximum angle, measured in radians. If we use the range to normalize the TD error δ in (6.2) $(\frac{|\delta|}{\text{Range}})$ then we can calculate confidence as $C = 1 - U$ with $C <= 0$ when there is no confidence and $C = 1$ when the predictor is completely confident, i.e., confidence is effectively normalized. Further, if we initialize $\boldsymbol{\zeta}$ as in (6.4) then for all possible feature vectors, $U_{t=0} = 1$ and therefore $C_{t=0} = 0$.

$$\boldsymbol{\zeta}[:]_{t=0} = \frac{1}{\#\text{ Active Features}} \tag{6.4}$$

It is also important that we consider how long it has been since the current state was observed. If it has been a long time since we last observed a state, then the confidence of that state should be low, regardless of how confident the predictor was the last time it observed that state. Thus, we might incrementally increase the uncertainty of non-active states by modifying (6.2) according to (6.5), where $\boldsymbol{\zeta}_0$ is the weight vector initialized as in (6.4) and $0 < \sigma \leq 1$ is the rate of return to the initial uncertainty. It should be clear that σ should be much smaller than η. Further, this increment would be applied to all elements of the estimator, not just those which are not currently active in $\boldsymbol{\phi}$. Practically, this may be acceptable, given the relationship between η and σ. However, it may be preferable to selectively apply this decay to just those features which are not presently active.

$$\boldsymbol{\zeta}_{t+1} = \boldsymbol{\zeta}_t + \eta(\frac{|\delta|}{\text{Range}} - \boldsymbol{\zeta}_t^\top \boldsymbol{\phi}_t)\boldsymbol{\phi}_t + \sigma(\boldsymbol{\zeta}_0 - \boldsymbol{\zeta}_t) \tag{6.5}$$

The proposed approach to confidence has several benefits: it gives a state-based measure, it captures a sense of state visitation and state recency, it captures a sense of convergence, it uses function approximation, it has linear computational complexity and finally the measure produced is useful for expressing whether or not the convergence values are good. That is, as the predictions converge, they converge to a level of TD error, which tells us how good the predictions are, not just that they converge. Combining this approach with the surprise measure implemented by A. White et al. (2014) should allow us to capture all of the desirable aspects for a predictive convergence measure as put forth in the beginning of this section.

Thus far, I have only considered confidence measures for predictions, beyond this, we may also want to combine measures of the agent's confidence in its actions, not just its predictions and temper actions based on these confidences by the level of perceived risk.

Further, I have only considered confidence measures from the agent's perspective of itself. An additional measure of the user's confidence, or trust in the robot would be beneficial for managing human/robot interactions. One can imagine that the agent could gradually increase the amount of action it took on the system as the user's trust in its abilities increases or back off if it senses that the user is not confident. The agent could use these measures to tentatively explore its action space in conjunction with the user.

The case for using confidence measures for the DPCC method is just one example of how control might be improved by their use. However, more generally, such confidence measures, from all these different aspects, offer ways of managing the interaction between human and robot, with potential benefits to increasing safety, effectiveness and user acceptance.

6.2 Time and Pattern Recognition

Most of the pattern recognition approaches used with EMG do not consider the effects of time in their predictions. That is, they simply take in a window of data and make a classification of it in isolation, with samples considered largely independent of one another. One approach has included a running trace of classification decisions so as to smooth classification outputs (L. J. Hargrove, E. J. Scheme, et al., 2010), but these methods are still ignorant to patterns of usage; including time would allow us to leverage such patterns. Equation (6.6) proposes a method by which predictions from a pattern recognition classifier, with likelihood L, might be combined with the prediction from a GVF, P, at time t for class i to produce a weighted likelihood. The scalar β is used to control weighting between the GVF and the pattern recognition likelihood. The equation shown incorporates a measure of confidence in GVF prediction as C_t, which is used to modify the weighting of the two different predictors; when confidence is low little weighting would be given to the GVF prediction and more would be given as confidence grows. A classification decision would then be made using a threshold on the weighted likelihood. This approach allows us to use the offline trained classifier as a baseline predictor while the output of the GVF is used to improve upon these predictions using ongoing experience.

$$Likelihood_{i;t} = \beta * C_t * P_{i;t} + (1 - \beta * C_t) * L_{i;t} \tag{6.6}$$

However it is not yet clear how errors in such a system could be identified in real time and how the GVF should be updated so as to appropriately capture those errors. Admittedly, this is a recurring theme in much of the work that I have done when using GVFs within positive feedback loops, for which I do not yet have a general solution.

6.3 Gamma selection

As described in Section 5.6.2 the *Direct Predictive Collaborative Control* method implemented in Chapter 5 used only a single GVF with a fixed γ value for predicting the target angle of each joint, meaning that the algorithm was always using predictions from a fixed timescale. In practice this is not ideal and could in fact make task performance worse; in some states the best behavior will come from looking only a short distance into the future, while in others it is better to look far ahead. Thus, having the ability to look various distances in the future and to learn which distance is appropriate for each state would be extremely beneficial. I outline three possible methods for addressing this.

Picking from several GVFs. The simplest method might be to have several GVFs of various γ ranging from 0 to 1. An algorithm such as Sarsa (R. S. Sutton and Barto, 1998) might then be used with function approximation to select the most appropriate GVF for each state. In order to have fine grained control it would be necessary to have large numbers of GVFs. However, this would effectively increase the number of actions the agent could take and therefore increase the difficulty of learning appropriate behavior as well as increase memory and total computation per timestep.

GVF blending. An alternative method might be to use only a handful of GVFs with γ ranging from 0 to 1 and weight the output of each to produce a final prediction. A possible method for doing this is to use some parametric function applied as a filter across the range. Figure 6.1 shows an example of this using a Gaussian weighting function. In this example there are five evenly spaced GVFs with γ values [0, 0.25, 0.5, 0.75, 1.0]. A Gaussian function is then used with these values to select the weights for each GVF, which are then normalized by the total of all the values. In our example, the weight for the GVF with $\gamma = 0.25$ would

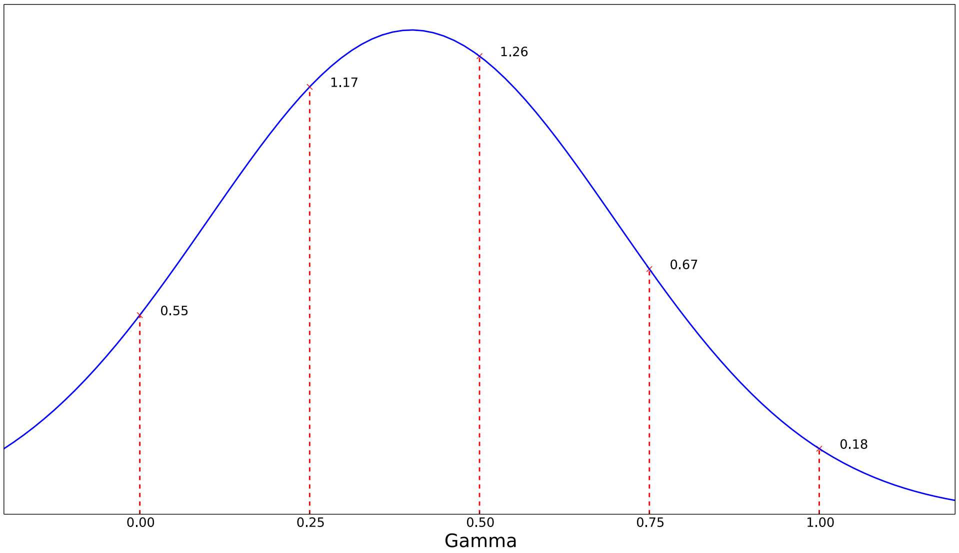


Figure 6.1: Weighting different GVFs. Five GVFs, with $\gamma = [0, 0.25, 0.5, 0.75, 1.0]$ are weighted using a Gaussian function.

thus be $\frac{1.17}{0.55+1.17+1.26+0.67+0.18} = 0.31$. The final output would then be an inner product between a vector of the weights and a vector of the GVF outputs. The parameters for the weighting function, here the mean and standard deviation, might be learned by means of an actor-critic algorithm (R. S. Sutton and Barto, 1998). This approach has the benefit of limiting the number of GVFs being learned, while providing a way of increasing resolution of the predictions of interest. However, there are several potential draw backs to this approach. First, it must have an independent learner for each parameter of the blending function. Second, while a Gaussian seems appropriate in that it is uni-modal, it is not clear what the best blending function might be. Finally, this approach assumes that blending the outputs of the various GVFs is a good thing, which is unknown.

State-dependent γ with a single GVF. Finally, a single GVF might be employed which used state dependent γ values. The difficulty then comes in the form of selecting the appropriate γ for each state. Again, we might imagine using an actor-critic algorithm for selecting γ.

Another method might involve using an additional GVF to produce an estimate of the appropriate value of γ. The cumulant for this predictor would be some measure of how much the system has achieved the desired output — essentially this is just the reward signal. The output of this GVF could then be passed through a squashing function such as the logistic function shown in (6.7), which will constrain output between [0,1]. Some consideration must then be given to selecting the scaling value k and the offset x_0. γ is then computed as $\gamma(s) = \sigma(\boldsymbol{\phi}(s)^\top \boldsymbol{w})$. If the system is performing worse than predicted then the GVF will move towards predicting lower value, which will reduce γ. On the other hand, if it performs better than predicted then the value of γ will increase. This approach may be the most direct way to determining a state-based γ value, but requires that the desired output be known and that it is possible to express a reward function relative to that output.

$$\sigma(x) = \frac{1}{1 + e^{-k(x - x_0)}} \tag{6.7}$$

All of the methods described require some way of saying whether the choice of timescale was good or not, which is where reward signals could play an important role. The most obvious reward signal presently available in the toggling proportional control system is the joint toggle itself. If the automated system worked perfectly, always predicting exactly what the user wanted and behaving appropriately to complete the user's goals, then we should expect no joint toggles. On the other hand, if the system is behaving poorly we can expect that the user will have to correct frequently, producing high rates of toggling. Thus, a toggle or toggle rate could provide a negative reward.

Explicit negative or positive signals from the amputee might also be incorporated. A user could provide feedback using a button push to indicate good or bad as done by Pilarski, Dawson, Degris, Fahimi, et al. (2011), or by giving verbal feedback by saying "good", or "bad."

Additionally, there are implicit signals that an amputee might give. These might include verbal cues, such as grunting, sighing or cursing. Or, perhaps an electroencephalogram (EEG) could be used to detect a persons sense of satisfaction with the arm's behavior directly from their brain waves (Esfahani and Sundararajan, 2011). Facial expressions might also offer clues as to the arm's performance. Tension in the other muscles of the torso or the other arm might indicate that the amputee is unhappy with the arm in general or is expecting to have to compensate for it.

Finally, there are reward signals that might come from the environment itself. For example, detecting collisions could be used to indicate poor performance. High overall current draws across the whole arm might indicate that arm was behaving in ways that are not energy efficient, providing yet another signal to use as feedback.

As the arm explores, using predictions at various timescales as target signals, it is important that it do so safely. One heuristic approach is to start out with $\gamma = 0$ and slowly increase it as confidence grows. When a negative reward is encountered, the arm might then retreat to smaller γ. As described in Section 6.1, Gehring and Precup (2013) implemented a method of safe exploration based on the idea of controllability, that is, a state is said to be more controllable if its predictive error is low. This approach may also be appropriate here.

6.4 Alternative Approaches to Collaborative Control

The main question explored in this research is "How can we make controlling prosthetic arms easier and more functional for amputees?" In Chapter 5, we sought to do this in the context of learning directly from user behavior and without modifying the existing hardware or adding additional sensory or input modalities to the system. That is, we have limited ourselves to two channels of information: a scalar channel, and a toggle channel. The following list proposes several desirable attributes for an intelligent control system for a prosthetic arm.

- User behavior should inform system behavior. That is, there are numerous signals, both explicit and implicit, coming from the user, which should inform the arm about its behavior and a user's intentions.
- The system should adapt to user behavior and changing circumstances.
- Control should be as natural and consistent as possible.

- Smooth transitions between shared and solo control are desired, both from the perspective of the user's experience as well as from the perspective of produced trajectories.
- The user must have a way to correct errors in learned behavior.
- The system should be easy to use, minimizing cognitive burden as much as possible.
- The user should maintain as much of a sense of control and embodiment over the arm as possible, while maximizing the practical functionality of the arm.
- The system **must** behave safely and predictably.

There are several solutions I will consider to this problem. In all cases, the system will attempt to predict what actions would be beneficial to the user and let us assume that an amputee's usage of the limb is informative to making those predictions. Further, let us assume that these coordinated predictions can be represented as joint trajectories, i.e., temporally coordinated joint movements.

A first approach might be to simply have the arm memorize trajectories and add those trajectories directly to the toggle list. Unfortunately, this actually has the potential to increase the disparity between the user input signals and the DOC of the system by constantly increasing the number of options the user has to toggle between. For this reason, this approach is rejected.

Consider two views to cooperative control. In the first view, a user and their helper both actively work together to accomplish the task. An example of this would be two people moving a piece of plywood. Each person is responsible for their portion of the work load. The helper takes cues from the leader, and likely has a good guess or explicit knowledge of the goal. A second view would have a *boss* showing the workers what to do and then letting them do it. The DPCC method in Chapter 5 takes the first view. It was my belief that this would allow the user to maintain a greater sense of control and embodiment. However, the delegation view, in which the arm would simply do what the user wanted, is a reasonable alternative, presenting a clear picture for implementation.

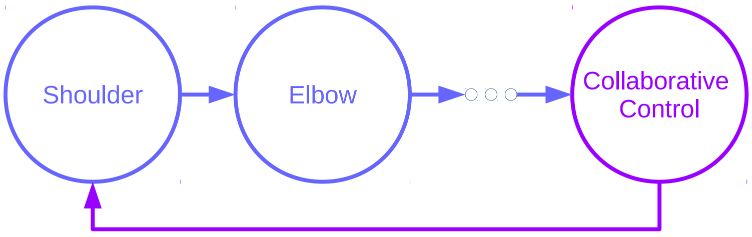


Figure 6.2: Collaborative control added to the toggle list.

Here I discuss a possible implementation of the delegation view to collaborative control, which I will call the *trajectory* approach. Rather than controlling individual joints, the user pushes and pulls all joints of the arm along the trajectory. In this case the system predicts the trajectory, and the user tells the system how fast and in what direction to move along that trajectory. The user can still have a toggle list as before, with all joints listed. Additionally, the option to enter collaborative control is added to the list as shown in Figure 6.2. The user can then enter and exit collaborative control simply by toggling. This provides a straight-forward way for the user to correct the system. Like the DPCC method, the fallback behavior for such a system would be the usual toggling proportional control. Therefore, the system only adds to performance.

Further, this approach can be readily combined with the adaptive switching work of Edwards et al. (2014). Thus, the system could learn to predict which joint was most likely needed when switching out of collaborative control, and conversely, when the user might want to select collaborative control. This provides an indirect way for the user's trust of the system to be integrated into control; if the user does not trust the system they will not select collaborative control.

Additionally, there is a role for predictive confidence, as described in Section 6.1, to play in the system as well. First, if the system is not sufficiently confident it should simply not offer the collaborative control option in the toggle list. Secondly, this confidence could be combined with some sort of feedback to inform the user when the system is confident enough that it can take over; essentially indicating to the user that the collaborative control option was available.

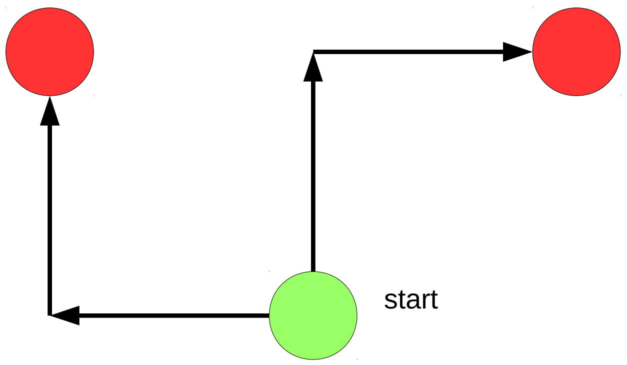


Figure 6.3: Two possible trajectories from the same starting position.

In the case of DPCC, having the user control one of the joints provides the system with information it needs to discriminate between available predictive options. This is important for predicting what should be done in situations like the one shown in Figure 6.3. Here we start in a position from which we have learned two trajectories, up and right, and left and up. If the user controls one joint they can make motions in either direction, providing additional state information to predict the appropriate trajectory. In the case where the user controls movement along the trajectory, we do not have the information necessary to yet distinguish the appropriate trajectory (assuming we do not add additional sensory modalities to our state space which might give us sufficient information in other forms other than joint position). This simply changes when the user has the option to switch to collaborative control; if the user starts in manual control, they must first make motion towards one of the trajectories before switching to collaborative control.

The trajectory approach is appealing as it is simpler than the DPCC method and presents a somewhat clearer picture. However, the DPCC approach offers the theoretical advantage of having the user's behavior in manual or collaborative control modes be the same — they simply control the joint they have selected. However, as alluded to by the discussion of disorientation in Section 5.6.6, this may not actually be realized in the real system. Further, the DPCC approach offers incremental assistance, which may be more beneficial than the trajectory based approach. Finally, from a user's perspective it is unclear which approach would allow for the greatest sense of embodiment. In conclusion, the trajectory approach presents an appealing picture and is worth investigating.

Chapter 7

Conclusion

The overall goal of my research has been to improve the control of powered prosthetic arms — increasing functionality and reducing the effort for amputees. I have pursued this with the philosophy that treating prosthetic arms as intelligent agents will improve control beyond what can be achieved by only thinking of them as electro-mechanical tools. I defined an *intelligent agent* as one which is able to learn about itself, its environment and the interaction between the two and adapt its behavior to improve its ability to accomplish a goal. While my long-term goals include integrating all of these aspects of intelligence, this thesis primarily focused on the use of predictions to control prosthetic arms.

Current methods of controlling prosthetic arms are tedious for amputees and many reject prosthetic arms citing difficulty of control and lack of functionality as significant factors. While mechanical functionality is rapidly increasing due to advances in hardware, control interfaces have not managed to keep up. The key problem is the gap between the number of available functions in the arm and the user's ability to attend to all that are needed or desired at any given instant, i.e., the communication interface between the user and the arm is too limited. This deficit is not unique to the prosthetic domain but can be found in many human-machine interfaces, including smart phones, computers and teleoperation of robots.

Prediction is believed to be a key component in making good control decisions for humans and other animals (Wolpert et al., 2001). From this, one might theorize, as I do throughout this thesis, that prediction is a beneficial tool to intelligent agents in general and may be useful in developing an intelligent prosthetic arm. This theory is bolstered by several demonstrations showing the benefits of predictions in robotic control (R. S. Sutton, Modayil, et al., 2011; Modayil, A. White, and R. S. Sutton, 2014; Edwards et al., 2014).

This thesis looked at several different aspects of prediction in prosthetic control. Chapter 3 identified one context under which current predictive pattern recognition methods for EMG-based prosthetic control do not generalize. This experiment reinforces similar findings made by others in the field including Cipriani, Sassu, et al. (2011) and Tkach et al. (2010). Further, this finding raises concern over the validity of evaluating pattern recognition methods without the use of actual amputees wearing a prosthetic and performing real-world tasks, a concern that has been similarly expressed by others in the field (L. Hargrove et al., 2007). Chapters 4 and 5 employed a different type of predictor known as a general value function, which allows online learning of any measurable signal. Chapter 4 demonstrated that it was possible to layer general value functions such that the output of one or more could be used as input for another general value function. Such an arrangement might allow us to make a great variety of compound predictions and provide robust state representations. These experiments also showed that arranging general value functions in layers could produce a boosting like behavior where the secondary layer is able to make highly accurate predictions of its target signal even though the primary layer is not. Such approaches may be of benefit in the combination of various predictors and improve upon such things as transfer learning across domains. Chapter 5, which is arguably the most important contribution of this thesis, used general value predictions in a novel control algorithm called *Direct Predictive Collaborative Control*, which allowed the arm to move joints on its own, in collaboration with the amputee, so as to achieve the amputee's movement goals. The algorithm did this by learning predictions about intended joint angles from observing the user's behavior during toggling proportional control. When the algorithm was then allowed to assist the user it did so by directing each joint towards the predicted joint angles prior to the user's own ability to execute those movements, effectively achieving helpful simultaneous multi-joint control in conjunction with the user's own direction. The algorithm was demonstrated on a 2 DOC robot arm with an able-bodied user guiding the arm through an angled wire maze. This successfully demonstrated that *Direct Predictive Collaborative Control* was able to improve the user's performance on the task, by reducing the time to complete the task, and reduce user effort, by reducing the number of toggles required. These results would benefit from repeated experiments with more subjects. Additionally, there are still many implementation details which can be improved, such as the selection of the appropriate prediction timescale at each timestep. Nonetheless, these early results are promising and warrant further investigation.

However, despite the functional improvements that might result from intelligent prosthetics, it is a big assumption to think that a prosthetic arm taking action on its own, as in the work I have presented with the *Direct Predictive Collaborative Control* approach, will be acceptable to prosthetic users at all. As has been mentioned, amputees have an intimate relationship with their arm, and even slight motion or action not perceived to be self-initiated by the user may be completely unacceptable to most amputees. This assumption needs to be tested. Unfortunately, it will be difficult to test this assumption without first building the control system. Wizard-of-Oz studies, like those performed with autonomous wheelchairs (Viswanathan et al., 2014), would be incredibly helpful, but it is not clear how such studies might be performed in a convincing way for amputees. Conducting such a study is a good opportunity for future research.

Before closing, I would like to give my opinion on what I presently believe to be the best approach to prosthetic arm control. First, the ultimate interface is a bi-directional neural interface where motor commands come directly from the nervous system and sensor signals return directly to the nervous system. This is an ongoing area of research and has been heavily funded by DARPA (Miranda et al., 2015). In my opinion, this is now a technically plausible approach within reach.

However, if instead we look only at what is clinically available *today* then the next best approach would involve targeted muscle reinnervation in combination with improvements on a pattern recognition methods. Admittedly, this does require additional surgery, which may not be an option for some. However, such surgery need not be considered frivolous as targeted muscle reinnervation has the added benefit of reducing issues with painful neuromas that can occur after amputation (Souza et al., 2014). Despite the shortcomings of the pattern recognition techniques described in Chapter 3, the classifications they make are valuable signals providing a great deal of very specific information about the user's intent over numerous joint synergies. Enhancing the use of these signals is a great place for reinforcement learning techniques to step in. Reinforcement learning allows us to continually adapt as the environment changes and to incorporate numerous sensory streams in ways that are computationally tractable in real time. This is a key point; the inclusion of additional sensory information such as gaze tracking, joint angles, tactile sensation, inertial measurement units, etc., could greatly enhance the predictive ability of these systems immensely and could be effectively integrated using reinforcement learning techniques.

Finally, viewing prosthetic arms as wearable intelligent robots implies that they have their own sense of agency — their own ability to learn from the world, and adapt their actions so as to improve their ability to meet goals. In this way, the arm becomes a partner or helper and is not directly tied to the user's commands, providing a flexibility that goes well beyond hand-coded or static solutions. This is a natural way to compensate for the disparity in the control interface which exists between the amputee and their arm. Indeed, we are rarely aware of the minute details of the motor commands we give over our natural limbs. Rather, our higher consciousness appears to delegate such details to lower level cognitive agents, simply providing goals. An intelligent prosthetic arm simply seeks to replicate this situation outside of the body. Ultimately, this is the largest contribution of this thesis: demonstrating that it is possible to construct a system that learns about a user's intentions and that such a system can assist a user in achieving their goals.

Bibliography

Ameri, A., Scheme, E., Kamavuako, E., Englehart, K., & Parker, P. (2013, September). Real time, simultaneous myoelectric control using force and position based training paradigms. *IEEE Transactions on Biomedical Engineering*, *61*(2), 279–287. doi:10.1109/TBME.2013.2281595

Amsuss, S., Gobel, P., Graimann, B., & Farina, D. (2014, August 18–22). Simultaneous, proportional wrist and hand control for natural, dexterous movements of a physical prosthesis by amputees. In *Myoelectric Controls Symposium (MEC)* (pp. 2–5). Fredericton, New Brunswick.

Biddiss, E. a. & Chau, T. T. (2007). Upper limb prosthesis use and abandonment: a survey of the last 25 years. *Prosthetics and Orthotics International*, *31*(3), 236–257. doi:10.1080/03093640600994581

Bishop, C. M. (2006). *Pattern recognition and machine learning.* New York, New York, USA: Springer Science+Business Media, LLC. doi:10.1117/1.2819119

Boostani, R. & Moradi, M. H. (2003, May). Evaluation of the forearm EMG signal features for the control of a prosthetic hand. *Physiological Measurement*, *24*(2), 309–19.

Castellini, C., Artemiadis, P., Wininger, M., Ajoudani, A., Alimusaj, M., Bicchi, A., Caputo, B., Craelius, W., Dosen, S., Englehart, K., Farina, D., Gijsberts, A., Godfrey, S. B., Hargrove, L., Ison, M., Kuiken, T., Marković, M., Pilarski, P. M., Rupp, R., & Scheme, E. (2014, August). Proceedings of the first workshop on peripheral machine interfaces: going beyond traditional surface electromyography. *Frontiers in Neurorobotics*, *8*(22), 1–17. doi:10.3389/fnbot.2014.00022

Castellini, C. & Nowak, M. (2014, August). EMG-based prediction of multi-DOF activations using single-DOF training: a preliminary result. In *Myoelectric Controls Symposium (MEC)* (pp. 45–49). Fredericton, New Brunswick.

Childress, D. S. & Weir, R. F. (2004). Control of limb prostheses. In D. G. Smith, J. W. Michael, & J. H. Bowker (Eds.), *Atlas of amputations and limb deficiencies* (Vol. 3, pp. 173–196). Rosemont, IL: American Academy of Orthopaedic Surgeons.

Cipriani, C., Sassu, R., Controzzi, M., Kanitz, G., & Carrozza, M. C. (2011, August). Preliminary study on the influence of inertia and weight of the prosthesis on the EMG pattern recognition robustness. In *Myoelectric Controls Symposium (MEC)*. Fredericton, New Brunswick.

Cipriani, C., Zaccone, F., Micera, S., & Carrozza, M. C. (2008). On the shared control of an EMG-controlled prosthetic hand: analysis of user-prosthesis interaction. *IEEE Transactions on Robotics*, *24*(1), 170–184. doi:10.1109/TRO.2007.910708

Collinger, J. L., Wodlinger, B., Downey, J. E., Wang, W., Tyler-Kabara, E. C., Weber, D. J., McMorland, A. J. C., Velliste, M., Boninger, M. L., & Schwartz, A. B. (2013). High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, *381*, 557–564. doi:10.1016/S0140-6736(12)61816-9

Dawson, M. R., Sherstan, C., Carey, J. P., Hebert, J. S., & Pilarski, P. M. (2014). Development of the Bento Arm: an improved robotic arm for myoelectric training and research. In *Myoelectric Controls Symposium (MEC)* (pp. 60–64).

Dumanian, G. A., Ko, J. H., O'Shaughnessy, K. D., Kim, P. S., Wilson, C. J., & Kuiken, T. A. (2009). Targeted reinnervation for transhumeral amputees: current surgical technique and update on results. *Plastic and Reconstructive Surgery*, *124*(3), 863–869. doi:10.1097/PRS.0b013e3181b038c9

Edwards, A. L., Dawson, M. R., Hebert, J. S., Sutton, R. S., Chan, K. M., & Pilarski, P. M. (2014, August). Adaptive switching in practice : improving myoelectric prosthesis performance through reinforcement learning. In *Myoelectric Controls Symposium (MEC)* (pp. 69–73). Fredericton, New Brunswick.

Englehart, K., Hudgins, B., Parker, P. a., & Stevenson, M. (1999). Classification of the myoelectric signal using time-frequency based representations. *Medical Engineering & Physics*, *21*(6-7), 431–8.

Esfahani, E. T. & Sundararajan, V. (2011). Using brain–computer interfaces to detect human satisfaction in human–robot interaction. *International Journal of Humanoid Robotics*, *08*(01), 87–101. doi:10.1142/S0219843611002356

Finley, F. R. & Wirta, R. W. (1967). Myocoder studies of multiple myopotential response. *Archives of Physical Medicine and Rehabilitation*, *48*(11), 598–601.

Fischer, H. (2014). *A guide to U.S. military casualty statistics: Operation Inherent Resolve, Operation New Dawn, Operation Iraqi Freedom, and Operation Enduring Freedom.*

Gehring, C. & Precup, D. (2013, May). Smart exploration in reinforcement learning using absolute temporal difference errors. In *Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 1037–1044). Saint Paul, MN.

Hargrove, L., Losier, Y., Lock, B., Englehart, K., & Hudgins, B. (2007, August). A real-time pattern recognition based myoelectric control usability study implemented in a virtual environment. (pp. 4842–4845). Cité Internationale, Lyon, France. doi:10.1109/IEMBS.2007.4353424

Hargrove, L. J., Englehart, K., & Hudgins, B. (2007, May). A comparison of surface and intramuscular myoelectric signal classification. *IEEE Transactions on Biomedical Engineering*, *54*(5), 847–53. doi:10.1109/TBME.2006.889192

Hargrove, L. J., Scheme, E. J., Englehart, K. B., & Hudgins, B. S. (2010, February). Multiple binary classifications via linear discriminant analysis for improved controllability of a powered prosthesis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *18*(1), 49–57. doi:10.1109/TNSRE.2009.2039590

Hebert, J. S., Elzinga, K., Chan, K. M., Olson, J., & Morhart, M. (2014). Updates in targeted sensory reinnervation for upper limb amputation. *Current Surgery Reports*, *2*(3), 1–9. doi:10.1007/s40137-013-0045-7

Hefftner, G., Zucchini, W., & Jaros, G. G. (1988). The electromyogram (EMG) as a control signal for functional neuromuscular stimulation - Part I: autoregressive modeling as a means of EMG signature discrimination. *IEEE Transactions on Biomedical Engineering*, *35*(4), 230–237. doi:10.1109/10.1370

Hudgins, B., Parker, P., & Scott, R. N. (1993, January). A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, *40*(1), 82–94. doi:10.1109/10.204774

Jang, S. H. (2013). Motor function-related maladaptive plasticity in stroke: a review. *NeuroRehabilitation*, *32*(2), 311–316. doi:10.3233/NRE-130849

Kormushev, P., Calinon, S., & Caldwell, D. (2013). Reinforcement learning in robotics: applications and real-world challenges. *Robotics*, *2*, 122–148. doi:10.3390/robotics2030122

Krueger, C. A., Wenke, J. C., & Ficke, J. R. (2012). Ten years at war: comprehensive analysis of amputation trends. *The Journal of Trauma and Acute Care Surgery*, *73*(6), 438–44. doi:10.1097/TA.0b013e318275469c

Lawrence, P., Herberts, P., & Kadefors, R. (1972). Experiences with a multifunctional hand prosthesis controlled by myoelectric patterns. In *The Fourth International Symposium on External Control of Human Extremities* (pp. 47–65). Dubrovnik, Croatia.

Light, C. M., Chappell, P. H., & Kyberd, P. J. (2002). Establishing a standardized clinical assessment tool of pathologic and prosthetic hand function: normative data, reliability, and validity. *Archives of Physical Medicine and Rehabilitation*, *83*, 776–783. doi:10.1053/apmr.2002.32737

Linden, D. J. (2003). From molecules to memory in the cerebellum. *Science*, *301*, 1682–1685. doi:10.1126/science.1090462

Littman, M., Sutton, R., & Singh, S. (2002). Predictive representations of state. *Advances in Neural Information Processing Systems (NIPS)*, *14*, 1555–1562.

Lyman, J. H., Freedy, A., & Prior, R. (1976). Fundamental and applied research related to the design and development of upper-limb externally powered prostheses. *Bulletin of Prosthetics Research*, *13*, 184–95.

McDonnall, D., Hiatt, S., Smith, C., & Guillory, K. S. (2012, August). Implantable multichannel wireless electromyography for prosthesis control. In *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)* (pp. 1350–1353). San Diego, California, USA. doi:10.1109/EMBC.2012.6346188

Miranda, R. A., Casebeer, W. D., Hein, A. M., Judy, J. W., Krotkov, E. P., Laabs, T. L., Manzo, J. E., Pankratz, K. G., Pratt, G. A., Sanchez, J. C., Weber, D. J., Wheeler, T. L., & Ling, G. S. F. (2015). Darpa-funded efforts in the development of novel brain-computer interface technologies. *Journal of Neuroscience Methods*, *244*, 52–67. doi:10.1016/j.jneumeth.2014.07.019

Modayil, J., White, A., Pilarski, P. M., & Sutton, R. S. (2012, October). Acquiring diverse predictive knowledge in real time by temporal-difference learning. In *IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1903–1910). Seoul, Korea.

Modayil, J. & Sutton, R. S. (2014, July). Prediction driven behavior: learning predictions that drive fixed responses. In *AAAI Workshop on AI and Robotics*. Québec City, Québec, Canada.

Modayil, J., White, A., & Sutton, R. S. (2014). Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior*, *22*(2), 146–160.

Novak, D. & Riener, R. (2014). A survey of sensor fusion methods in wearable robotics. *Robotics and Autonomous Systems*. doi:10.1016/j.robot.2014.08.012

Ohnishi, K., Weir, R. F., & Kuiken, T. a. (2007). Neural machine interfaces for controlling multifunctional powered upper-limb prostheses. *Expert Review of Medical Devices*, *4*(1), 43–53. doi:10.1586/17434440.4.1.43

Nitime. (2015). Version 0.5. Retrieved from http://nipy.org/nitime/

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, *30*(3), 286–297. doi:10.1109/3468.844354

Parker, P., Englehart, K., & Hudgins, B. (2006, December). Myoelectric signal processing for control of powered limb prostheses. *Journal of Electromyography and Kinesiology*, *16*(6), 541–8. doi:10.1016/j.jelekin.2006.08.006

Pattichis, C. S. & Elia, A. G. (1999). Autoregressive and cepstral analyses of motor unit action potentials. *Medical Engineering & Physics*, *21*(6-7), 405–19.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peerdeman, B., Boere, D., Witteveen, H., Huis in ‘tVeld, R., Hermens, H., Stramigioli, S., Rietman, H., Veltink, P., & Misra, S. (2011). Myoelectric forearm prostheses: state of the art from a user-centered perspective. *The Journal of Rehabilitation Research and Development*, *48*(6), 719–738. doi:10.1682/JRRD.2010.08.0161

Pilarski, P. M., Dawson, M. R., Degris, T., Fahimi, F., Carey, J. P., & Sutton, R. S. (2011, June). Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *IEEE International Conference on Rehabilitation Robotics* (pp. 134–140). Zurich, Switzerland. doi:10.1109/ICORR.2011.5975338

Pilarski, P. M., Dawson, M. R., Degris, T., Carey, J. P., Chan, K. M., Hebert, J. S., & Sutton, R. S. (2013, June). Adaptive artificial limbs. *Robotics and Automation Magazine*, *20*(1), 53–64.

Pilarski, P. M., Dawson, M. R., Degris, T., Carey, J. P., & Sutton, R. S. (2012, June). Dynamic switching and real-time machine learning for improved human control of assistive biomedical robots. In *IEEE International Conference on Biomedical Robotics and Biomechatronics* (pp. 296–302). Roma, Italy.

Pilarski, P. M., Dick, T. B., & Sutton, R. S. (2013). Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints. *IEEE International Conference on Rehabilitation Robotics*, 1–8. doi:10.1109/ICORR.2013.6650435

Quigley, M., Conley, K., Gerkey, B., FAust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Mg, A. (2009, May). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software* (Vol. 3, *3.2*, p. 5). Kobe, Japan. doi:http://www.willowgarage.com/papers/ros-open-source-robot-operating-system

Redish, A. D. (2013). *The mind within the brain: how we make decisions and how those decisions go wrong*. New York, New York, USA: Oxford University Press.

Resnik, L., Lieberman Klinger, S., Etter, K., & Fantini, C. (2014, July). Controlling a multi-degree of freedom upper limb prosthesis using foot controls: user experience. *Disability and Rehabilitation Assistive Technology*, *9*(4), 318–29. doi:10.3109/17483107.2013.822024

Resnik, L., Meucci, M. R., Lieberman-Klinger, S., Fantini, C., Kelty, D. L., Disla, R., & Sasson, N. (2012, April). Advanced upper limb prosthetic devices: implications for upper limb prosthetic rehabilitation. *Archives of Physical Medicine and Rehabilitation*, *93*(4), 710–717. doi:10.1016/j.apmr.2011.11.010

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.

Scheme, E., Lock, B., Hargrove, L., Hill, W., Kuraganti, U., & Englehart, K. (2013, March). Motion normalized proportional control for improved pattern recognition based myoelectric control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *22*(1), 149–157. doi:10.1109/TNSRE.2013.2247421

Scheme, E. & Englehart, K. (2011). Electromyogram pattern recognition for control of powered upper-limb prostheses: state of the art and challenges for clinical use. *The Journal of Rehabilitation Research and Development*, *48*(6), 643–659. doi:10.1682/JRRD.2010.09.0177

Smith, L. H., Hargrove, L. J., Lock, B. A., & Kuiken, T. A. (2011). Determining the optimal window length for pattern recognition-based myoelectric control: balancing the competing effects of classification error and controller delay. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *19*(2), 186–192. doi:10.1109/TNSRE.2010.2100828

Souza, J. M., Cheesborough, J. E., Ko, J. H., Cho, M. S., Kuiken, T. A., & Dumanian, G. A. (2014). Targeted muscle reinnervation: a novel approach to postamputation neuroma pain. *Clinical Orthopaedics and Related Research*, *472*(10), 2984–2990. doi:10.1007/s11999-014-3528-7

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement learning: an introduction.* Cambridge, MA: MIT Press. doi:10.1109/TNN.1998.712192

Sutton, R. (2005). Tiles (version 2.1). Retrieved from http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/RLtoolkit/tilecoding.html

Sutton, R. S. (1988, August). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44. doi:10.1007/BF00115009

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., & Precup, D. (2011). Horde : a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction categories and subject descriptors. In *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (Vol. 2, pp. 761–768). doi:10.1037/a0023964

Tkach, D., Huang, H., & Kuiken, T. A. (2010). Study of stability of time-domain features for electromyographic pattern recognition. *Journal of Neuroengineering and Rehabilitation*, *7*, 21. doi:10.1186/1743-0003-7-21

van Seijen, H. & Sutton, R. S. (2014, June). True online TD (λ). In *International Conference on Machine Learning (ICML)* (Vol. 32, pp. 692–700). Beijing, China.

Viswanathan, P., Bell, J. L., Wang, R. H., Adhikari, B., Mackworth, A. K., Mihailidis, A., Miller, W. C., & Mitchell, I. M. (2014, September). A Wizard-of-Oz intelligent wheelchair study with cognitively-impaired older adults : attitudes toward user control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Assistive Robotics for Individuals with Disabilties: HRI Issues and Beyond.* Chicago, IL, USA.

Wang, W., Collinger, J. L., Degenhart, A. D., Tyler-Kabara, E. C., Schwartz, A. B., Moran, D. W., Weber, D. J., Wodlinger, B., Vinjamuri, R. K., Ashmore, R. C., Kelly, J. W., & Boninger, M. L. (2013). An electrocorticographic brain interface in an individual with tetraplegia. *PLoS ONE*, *8*(2), 1–8. doi:10.1371/journal.pone.0055344

White, A. (2015). *Developing a predictive approach to knowledge* (Doctoral dissertation, University of Alberta).

White, A., Modayil, J., & Sutton, R. S. (2014, July). Surprise and curiosity for big data robotics. In *AAAI Workshop on Sequential Decision Making with Big Data* (pp. 19–23). Québec City, Québec, Canada.

White, M. & White, A. (2010). Interval estimation for reinforcement-learning algorithms in continuous-state domains. In *Advances in Neural Information Processing Systems (NIPS).*

Wolpert, D. M., Ghahramani, Z., & Flanagan, J. R. (2001, November). Perspectives and problems in motor learning. *Trends in Cognitive Sciences*, *5*(11), 487–494. doi:10.1016/S1364-6613(00)01773-3

Yatsenko, D., McDonnall, D., & Guillory, K. S. (2007, August). Simultaneous, proportional, multi-axis prosthesis control using multichannel surface EMG. (pp. 6134–7). Cité Internationale, Lyon, France. doi:10.1109/IEMBS.2007.4353749

Young, A. J., Smith, L. H., Rouse, E. J., & Hargrove, L. J. (2012, June). A new hierarchical approach for simultaneous control of multi-joint powered prostheses. In *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics* (pp. 514–520). Roma, Italy. doi:10.1109/BioRob.2012.6290709

Zardoshti-Kermani, M., Wheeler, B., Badie, K., & Hashemi, R. (1995). EMG feature evaluation for movement control of upper extremity prostheses. *IEEE Transactions on Rehabilitation Engineering*, *3*(4), 324–333. doi:10.1109/86.481972

Ziegler-Graham, K., MacKenzie, E. J., Ephraim, P. L., Travison, T. G., & Brookmeyer, R. (2008). Estimating the prevalence of limb loss in the united states: 2005 to 2050. *Archives of Physical Medicine and Rehabilitation*, *89*, 422–429. doi:10.1016/j.apmr.2007.11.005

Appendix A

Bento Arm

The Bento Arm, which was used for the main body of research in Chapter 5, is a custom arm designed and built in the Bionic Limbs for Improved Natural Control Lab (BLINC) as part of a project by the Alberta Innovates Centre for Machine Learning (AICML). I have, thus far, been solely responsible for developing and maintaining the control software used for operating and interfacing with the Bento Arm. The Bento Arm has become a significant piece of research equipment for our lab, not just for my own experiments, with several more copies of the arm in production at this time.

All of the control software for the Bento Arm was implemented using the Robot Operating System (ROS), an open source robotics platform. ROS is used widely in research and is even moving into the industrial automation space with the ROS-Industrial initiative (http://rosindustrial.org/). Despite the name, ROS is not an operating system, but rather a meta-operating system, which runs, primarily, on Ubuntu, Linux. ROS facilitates communication between various programmatic modules, called *nodes*, over standard network communication channels, either locally or across multiple machines; it provides a common communication infrastructure which allows code to be modular and reusable.

ROS uses three forms of communication between software components:

1. *topics* - Topics are asynchronous messages built on a publisher/subscriber model. They are the primary way in which communication is done in ROS.

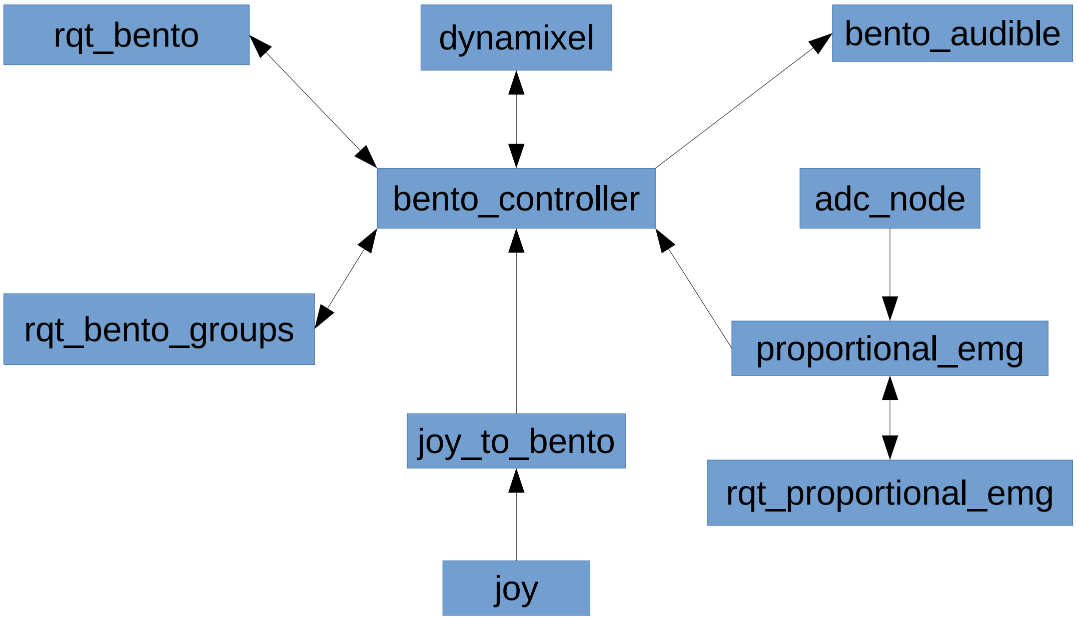


Figure A.1: Bento architecture

2. *services* - Services are synchronous, i.e., the caller of a service waits for a response from the service being called. These are useful for making calls which must be completed before other functions are performed, such as changing internal state of a particular node. An example might be enabling torque control.

3. *actions* - Actions are long-running goals, which are not completed in a single call. The caller of an action specifies the desired output, receives updates about progress, and finally receives notification when the action has been completed. Actions can be interrupted. An example of an action might be a command to go to home position.

This appendix will briefly describe the software, but is not a substitute or duplicate for the actual documentation, which is currently available on corresponding wikis which are presently private to the Pilarski lab. Figure A.1 presents a high level view of the architecture, showing the various packages involved. The architecture is grouped by functionality and explained in the following sections.

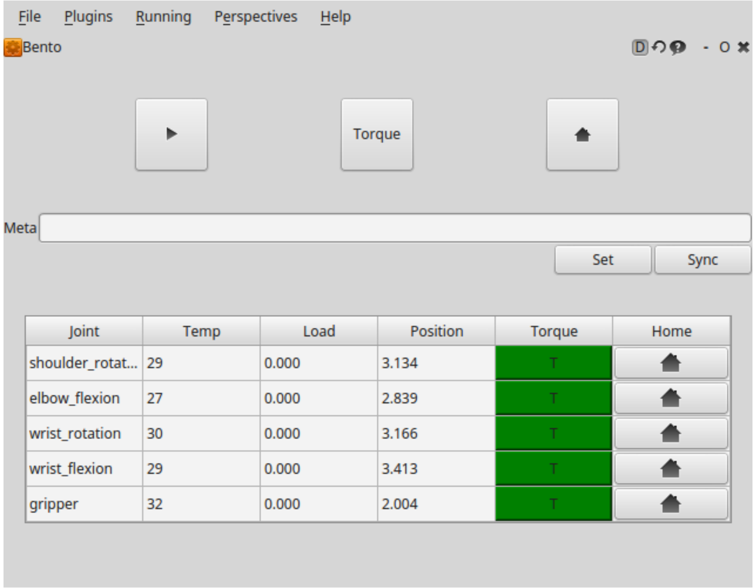
A.1 Functionality

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. Additionally, joints can be grouped and controlled as described in the next section. State information is published at regular intervals.

The bento_controller also provides several service calls. The most important are shown below (for a complete list see the source code repository).

- *pause* - Pause and unpause the arm. When the arm is paused it ignores all incoming commands to move the arm. Each individual joint can be paused or the arm as a whole.
- *torque_enable* - Enables or disables torque. When torque is disabled the joint provides no current and goes limp. Torque can be switched on the arm as a whole, or on individual joints. A side effect of enabling or disabling torque is that the arm is paused.
- *go_home* - Pauses the arm and sends it to a home position. This is presently implemented as a service call, but would be better implemented as an action.
- *set_meta* - The state messages published by bento_controller have a field called *meta*, which allows an experimenter to add meta information to the messages. This service call allows this value to be set.

A graphical interface called rqt_bento, as shown in Figure A.2 allows the user to pause the arm, toggle torque, send the arm home, and set the meta information of the state messages.



Joint	Temp	Load	Position	Torque	Home
shoulder_rotat...	29	0.000	3.134	T	⌂
elbow_flexion	27	0.000	2.839	T	⌂
wrist_rotation	30	0.000	3.166	T	⌂
wrist_flexion	29	0.000	3.413	T	⌂
gripper	32	0.000	2.004	T	⌂

Figure A.2: rqt_bento: basic GUI for controlling the Bento Arm

A.1.1 Joint Groups

Joint groups are implemented in order to facilitate control of the arm in the same way as an amputee would control the arm using the standard toggled proportional control method. Joint groups are simply lists of joints defined by the developer. For a given group there is only ever one active joint in the list. Any movement commands that are given for the group are routed only to the active joint. Specific joints can be set as the active joint. Additionally, the group allows the active joint to be toggled forward or backward through the list one joint at a time. Joint groups are presently implemented by the bento_controller node. However, it is my recommendation that this code be split off into its own package.

Control of joints in a joint group are based on velocity commands. Incoming commands are in the range [-1,1]. The incoming value is then scaled across the max speed configured for the currently active joint.

When the active joint of a group is changed a topic is published to alert subscribers of the change. The bento_audible node allows a developer to map those messages to play audio files. For example, if a group is changed to use the elbow joint an audio file saying "elbow" might be played.

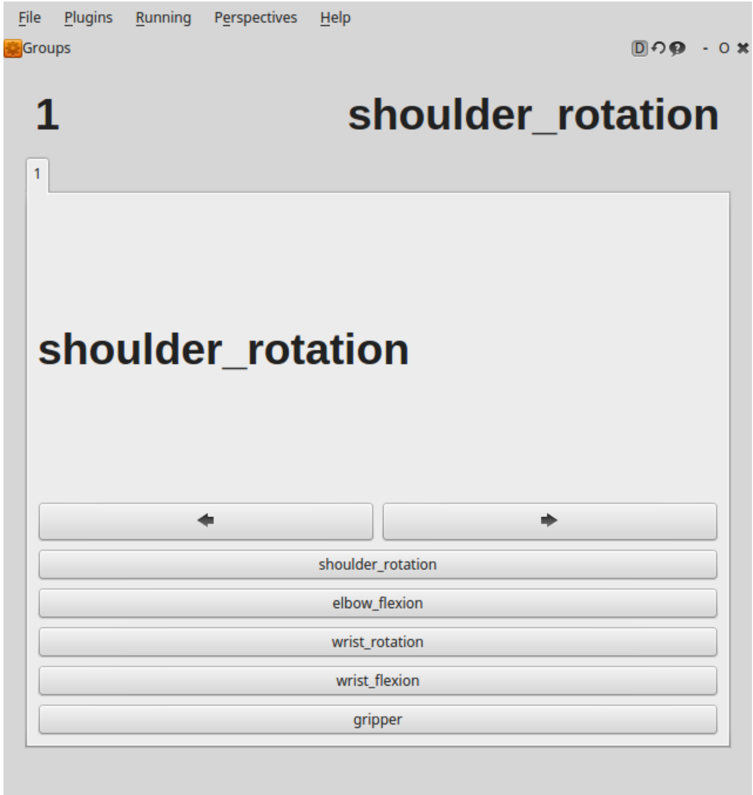


Figure A.3: rqt_bento_groups: Joint Groups Visualization

A graphical interface is provided for managing groups, through rqt_bento_groups as shown in Figure A.3, which lists each group available, the list of joints in each group and the currently active joint. It allows the user to manually set any joint active or toggle forward or backward through the list.

A.1.2 Toggling Proportional EMG Control

Presently EMG signals are read in through a National Instruments NI USB-6216 data acquisition device (DAQ). Unfortunately, this has presented several difficulties as the driver availability for this device on Ubuntu based platforms is far behind the current long term support kernels available. As a result, computers using this node must be running 32-bit Ubuntu Raring, or Mint 15, which in turn means that these computers are limited to the ROS Hydro release. Till now, no difficulty has been observed in connecting a Hydro based computer with an Indigo (the subsequent release) based one.

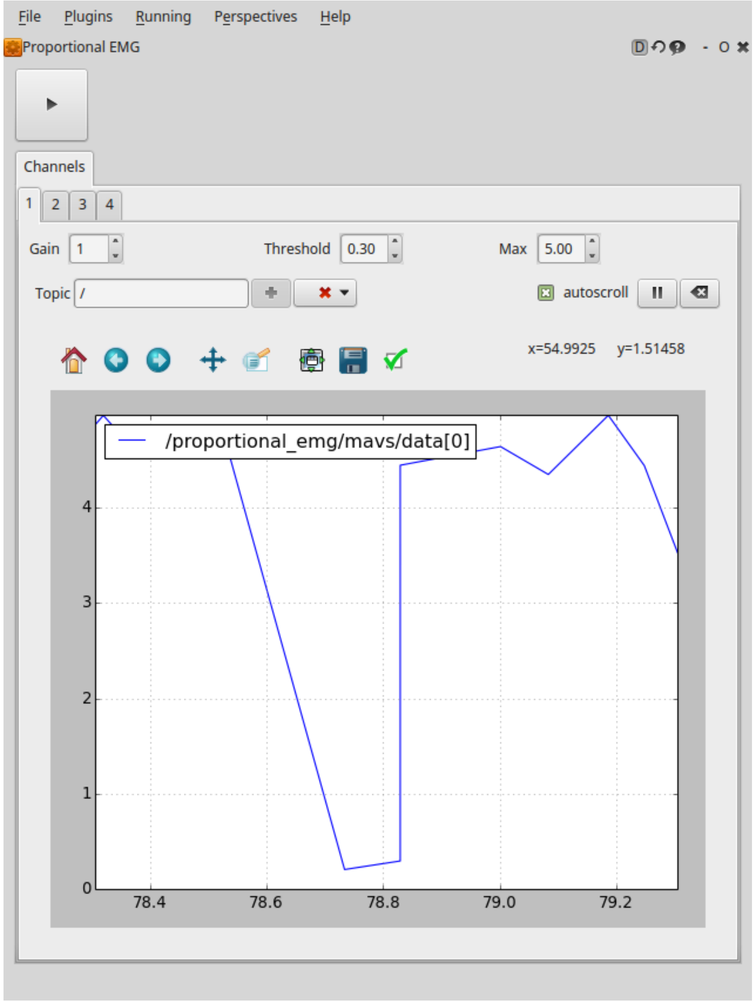
The adc_node is responsible for reading in values from the DAQ and publishing them as a ROS topic, which are then read by the proportional_emg node. The proportional_emg node allows pairs of channels to be configured to send proportional control messages to a single joint group. Individual channels can be configured to be used as toggle signals for a group. At present, the use of cocontractions to toggle a joint group is not implemented.

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. For a toggling channel, the *threshold* value is used as the point at which a toggling command will be sent, and *max* is ignored.

The rqt_proportional_emg node provides a graphical interface for controlling the proportional controller as shown in Figure A.4. It shows each of the channels available and the signals coming in for that channel. The operator is able to directly adjust *gain*, *threshold*, and *max* on the screen. The interface also provides a large pause/play button, which controls whether or not the proportional_emg node is currently sending group commands to bento_controller.

A.1.3 Joystick Control

Joystick control is accomplished through the use of the standard ROS joy node and a custom joy_to_bento node. The joy_to_bento node listens for incoming messages from a joystick and converts them into the correct commands for controlling the Bento arm. A joystick can be used to send joint movement commands as well as joint group commands and allows the user to make service calls to bento_controller for toggling pausing, turning torque on and off, and



Gain 1

Threshold 0.30

Max 5.00

Topic /

☒ autoscroll

x=54.9925 y=1.51458

Figure A.4: rqt_proportional_emg: Proportional EMG GUI

sending the arm to its home position. The joy_to_bento node is written in such a way as to take in a configuration file which creates a mapping between buttons and axis commands from the joystick and the desired Bento commands. The mappings can be configured in many different ways. The joystick can be configured in such a way as to operate the arm in the same was as the proportional EMG controller. Finally, it should be noted that using the joystick and the proportional EMG controller at the same time causes interference. Therefore a button on the joystick should be configured to toggle joy_to_bento's own internal paused state, which enables and disables sending control messages.

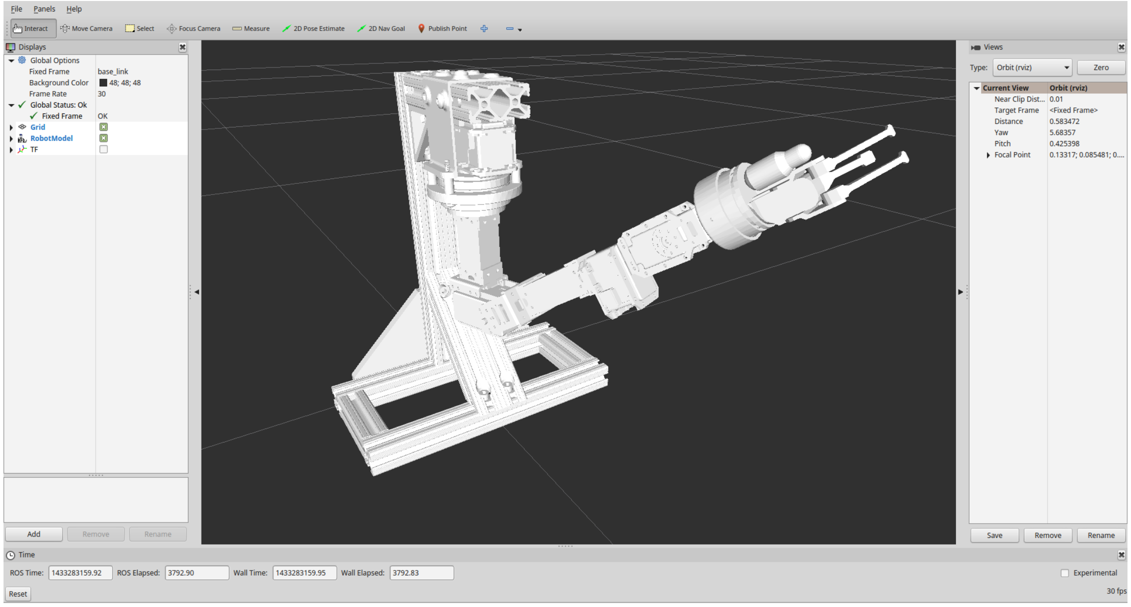
A.2 Dynamixel Control

While the bento_controller node is written to allow many different configurations and support various grippers, only support for Dynamixel servo motors has thus far been implemented.

The existing Dynamixel package for ROS implements several different types of servo motor control including position and torque based control. However, we desired both velocity and position based control, which required that I write my own controller to support velocity.

Velocity based control on Dynamixel motor presents an awkward challenge. The Dynamixel API can operate in two modes (some servos can also operate in torque based control modes, but these were not yet used): wheel mode, and joint mode. Wheel mode allows the servo to spin continuously and uses velocity based control. Joint mode allows a developer to set limits from 0 to 360 degrees. While joint mode operates using position based control, it allows you to set a velocity to use to approach the target position. However, if you set a zero velocity, it does not mean stop, instead it is interpreted as go as fast as possible. While the joint limits control does not consider the whole kinematics of the arm only the limits of a single joint, it is still desirable as a first level safety measure. Thus, the velocity based controller I have implemented operates in joint mode, so as to maintain the internal joint limit functionality of the servos. To send a velocity command, first the desired speed is set on the servo and then the limit in the direction of desired travel is given as the target position. This creates a bit of difficulty in stopping the arm as we cannot send a zero velocity command, but must, instead, estimate where the joint will be at the time it actually stops. The reason why we must estimate this is that there is delay in the state updates as well as in the time it takes for the command to be executed. At present the estimation is overly simplified, simply adding the difference between the current position and previous position to the current position. The result is that when you try to stop when traveling at higher speeds the arm bounces back as the stop target position is not far enough ahead.

Another downside to the current implementation is that it currently depends on a custom fork I made of the official repository, which means that I must keep it up to date or fall behind.



Displays	
Global Options	
Fixed Frame	base_link
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	☒
RobotModel	☒
TF	□

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip Dist...	0.01
Target Frame	<Fixed Frame>
Distance	0.583472
Yaw	5.68357
Pitch	0.425398
Focal Point	0.13317; 0.085481; 0....

ROS Time: 1433283159.92 ROS Elapsed: 3792.90 Wall Time: 1433283159.95 Wall Elapsed: 3792.83 □ Experimental

Figure A.5: RViz model of the Bento Arm

I consider improving the Dynamixel control to be the top priority for improvement. Specifically, the stopping ability must be improved as well as reverting to the official repository so that we can stay in sync rather than maintaining a separate fork.

A.3 Visualization and Simulation

ROS provides a package for visualization called RViz, which provides a framework for viewing a robot, and how the robot perceives the world. I have created an initial model, based on the CAD drawings of the Bento Arm, for the RViz package, as shown in Figure A.5, which simply allows the user to view the arm orientation in real time.

Additionally, ROS integrates with a simulation package called Gazebo, which allows the simulation of environments, robots, objects, and sensors. I have also implemented a basic simulation of the Bento Arm based on the CAD drawings. However, the correct inertial values for each joint have not yet been specified. Additionally, the simulator is presently limited to driving the RViz visualization as the bento_controller node does not yet support integration with the simulator. In order to do that, is my recommendation that the new ros_control design principles be evaluated.