Brain-Computer Interface Control of an Anthropomorphic Robotic Arm

Samuel T Clanton

CMU-RI-TR-11-21

July 21, 2011

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Andrew Schwartz, Chair
Nancy Pollard
George Stetten
Neville Hogan, Massachusetts Institute of Technology

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Brain-computer interface, BCI, brain-machine interface, neuroprosthesis, motor control, robot, shared control, virtual fixture, impedance, compliance

For Kevin

Abstract

This thesis describes a brain-computer interface (BCI) system that was developed to allow direct cortical control of 7 active degrees of freedom in a robotic arm. Two monkeys with chronic microelectrode implants in their motor cortices were able to use the arm to complete an oriented grasping task under brain control. This BCI system was created as a clinical prototype to exhibit (1) simultaneous decoding of cortical signals for control of the 3-D translation, 3-D rotation, and 1-D finger aperture of a robotic arm and hand, (2) methods for constructing cortical signal decoding models based on only observation of a moving robot, (3) a generalized method for training subjects to use complex BCI prosthetic robots using a novel form of operator-machine shared control, and (4) integrated kinematic and force control of a brain-controlled prosthetic robot through a novel impedance-based robot controller. This dissertation describes each of these features individually, how their integration enriched BCI control, and results from the monkeys operating the resulting system.

Acknowledgments

I would first like to thank my advisor, Andy Schwartz, for his uncompromising ability to propel me into tackling large problems. Without his guidance and sometimes uncanny insight, this project would never have been accomplished. I thoroughly enjoyed our frequent discussions and hope that if he had to choose somebody with whom to hold an argument, he would choose me. I would also like to thank the other members of my committee - Nancy Pollard, George Stetten, and Neville Hogan. I appreciate their willingness to serve on my committee and their suggestions, which have had a large influence on creating this dissertation. I also want to thank Takeo Kanade and the QoLT Center for supporting me during the transition into Andy's laboratory. Next, I want to thank all current and past members of Robot Team. This includes Meel Velliste, for getting me started on the project and his constant support throughout, Zohny Zohny, whose clever application of common sense solved many crucial problems, Rob Rasmussen, for taking on the role of new guy wholeheartedly, as well as Morgan Jeffries, Andrew Whitford, and Angus McMorland for their contributions to the project. I would also like to thank our interns, Charles Frantz, Bridget Endler, and Andrew Weitz, who were able to be extremely helpful in project development. I am also grateful to Steve Chase in our lab, whose insights were particularly valuable in helping me to understand when I didn't invent something, as well as Scott Kennedy who helped with the 3D renderings in this dissertation. I also want to thank Michael Palazzolo, Sagi Perel, George Fraser, Chance Spalding, and other members of our laboratory with whom I have had many valuable discussions and a lot of fun, as well as Rachelle Stopczynski for her understanding. On a personal level, I want to thank my parents for their support and flexibility throughout this process. Last but not least, I want to thank my oversized dog Kevin for literally standing at my side throughout the process of preparing this dissertation.

Contents

List of Figures

Chapter 1

Introduction

Over the past decade, research and development of robotic arms and hands has accelerated to the extent that robots that eclipse the controllable complexity of the human limb are starting to become available. While these advanced robots are intended for use as prosthetic devices, their development has not been matched by an understanding of how they can be controlled efficiently by prosthetic users. The potential clinical utility of these robots will be limited by the maximum bandwidth of the control interface with which they are operated. While existing prosthetic control schemes in use by patients are generally based upon residual patient movement, those with the least ability to move have the greatest need for rich control of sophisticated prosthetic devices.

To meet the challenge of affording people with the greatest level of disability the ability to control sophisticated prosthetics, brain-computer interface (BCI) control schemes allow the movement of a robot to be guided by signals directly recorded from the brain. Patients with conditions ranging from spinal cord injury to amyotrophic lateral sclerosis (ALS) and locked-in syndrome could potentially greatly benefit from control schemes based on BCI. Even for amputees and patients capable of a significant amount of volitional motor movement, direct brain interface may be the most intuitive way to control complex devices.

BCI experiments in which subjects control a device continuously have focused almost exclusively on controlling the linear movement of a cursor or robotic gripper in point or space partition target acquisition tasks. In contrast, human arm behaviors include a much larger range of movements by integrating rotation of the wrist with

hand translation. These movements do not take place in an empty, dynamically unconstrained environment; force interactions between the limb and its environment shape the evolution of arm movements. Both of these factors are essential features of the way humans perform activities of daily living and are important goals for designing BCI prosthetic systems to restore meaningful function to people with disabilities.

In this work, we address integrated BCI kinematic (motion) and dynamic (force) control of the translation, orientation, and finger aperture of a robotic hand. In a series of experiments, two monkeys were able to use a BCI to control a robotic arm in a task requiring simultaneous control of linear and rotational movements. This task was completed in an environment that resembled the real world in that the robot interacted directly with external objects; force interactions with these objects could be controlled via the BCI. This work represents a transition beyond point acquisition tasks for BCI, moving towards control systems for complete prosthetic limbs that can functionally replace physiologic ones.

Creation of the BCI robot control system that made this possible was not a task of knowing exactly what brain activity would allow 7-DoF control of a robot to take place then engineering the system that would effect that intended behavior physically. There is no way to know this model *a priori*; brain control of a robot is not a natural process that can be examined in a normally behaving subject. The connection between thought and robot movement must be instantiated for the subject to incorporate control of the robot into the existing motor control schema and allow the subject to embody the limb. To make adoption of control over the BCI possible, we relied on two related processes: (a) the transfer of features from the control of natural movements into a BCI decoding system that was compatible with adoption by the subject and (b) implementation of a systematic method to allow this adoption to take place without the subject becoming overwhelmed by the challenge. The components of the BCI system that support these two processes are described in this dissertation.

1.1 Contributions of this Work

While the end result of our work is a significant advance in the number of degrees of freedom (DoF) and quality of control that has been exhibited with a brain-computer interface, it is also the first demonstration of a new general model for establishing

control of complex BCI devices. This model incorporates a set of features that are desirable in BCI systems as the state of the art moves towards clinical replacement of full limb function in the disabled. These features include

Direct Control of Complex Robotic Systems

Microelectrode interface BCI control methods have generally been based on techniques originally developed for the control of the linear movement of cursors. The techniques and extraction algorithms that have been used are identical to those used for control of 3-D cursors on a display, with movement of the cursor transferred to the movement of a robot arm. In this work, methods used for cortical decoding and robot control are designed to work directly with general robotic hardware, with no intermediary vestigial layer derived from linear cursor control.

No Movement Required

Under the assumption that BCI control of prosthetics is closely related to brain activity involved in executing natural movements, control paradigms have often relied on real or intended natural subject movements in the process of building cortical decoding models. Real movements cannot be relied upon in disabled subjects, while intended movements – "imagine moving your hand" – become less useful if prosthetic devices diverge structurally from subject physiology or the representation of natural movement in the brain has diminished as the result of a motor deficit. An alternative method to build a cortical decoding model is to rely on cortical activity elicited by observation of movement of the effector device. Previous studies have used this as a basis for control, but in each case a previous motor task had been performed that established an initial connection between natural and robot movement before moving to observation. In this work, we expose naive subjects to the motor task and create the decoding model based on observation alone, with no physiologic motor task performed at any stage in the process.

Building-block structure for developing skill

After building a cortical decoding model to enable brain control of a device, it cannot be assumed that the subject will then be able to control it immediately. A useful decoding model is only one part of what is needed to develop control ability with a BCI. The other part is practice, but it may be extremely difficult to immediately control a complex robot in which the controlled DoF cannot be

selectively manipulated. Our system presents a systematic way to control the difficulty of a complex BCI task so that a subject will not be overwhelmed on the way to developing full control. Instead, the difficulty can be incrementally increased on a per-control dimension basis until competence is achieved.

Modular construction

The evolution of BCI control system design is a user-centered process of fitting the control pathway to the behavior of the user and demands of the task. The closed-loop nature of brain-computer interface systems makes offline evaluation of the effectiveness of the parts of its parts very difficult. We have created a modular system architecture for BCI that facilitates the development process by allowing the rapid interchange of highly cohesive system components in real time. Using this process, candidate modules for performing tasks such as neural extraction and robot control could be developed and evaluated rapidly.

To promote these features, a set of novel engineered systems have been created that form a large portion of this dissertation. These systems are

1. A new formulation of algorithms used for cortical decoding that extends BCI to the control of 3-D rotational velocity and to other first-order systems.
2. A novel method for operator-machine shared control that was developed to provide a substrate for observation-based decoder calibration and allow subjects to incrementally learn BCI control of a specific device. While developed for this application, it is also relevant to shared-control methods used in telerobotic and robotic surgical applications.
3. A new method for controlling robotic manipulators based on the recursive Newton-Euler equations that was developed to provide a straightforward way to implement force and kinematic control systems. Using this method, brain control of robot endpoint kinematics was integrated with a Cartesian impedance control algorithm that allowed indirect control of interaction forces between the prosthetic robot and objects in the environment.

1.2 Contents of Dissertation

- Chapter 2 contains a review of brain-computer interface control systems and an overview of the 7-DoF brain control system that is presented in this dissertation.

- Chapter 3 discusses the 6-DoF cortical decoding system that was used to translate neural activity into linear and rotational robot motion commands.

- Chapter 4 presents a novel shared-control algorithm that can be applied to telerobotic, haptic, and prosthetic systems.

- Chapter 5 contains a discussion of how a shared control system based on the one in Chapter 4 can be employed in the control of brain-computer interface prosthetic devices.

- Chapter 6 presents the robot controller that was used to provide kinematic and dynamic control of the robot.

- Chapter 7 contains a self-contained review of the complete brain-computer interface experiment with overall results.

- Chapter 8 contains detailed descriptions and methods regarding the BCI system.

- Chapter 9 concludes the dissertation of what has been accomplished and its role in how future brain-computer interface systems can be built.

Chapter 2

Background and Overview

This chapter contains a short summary of brain-computer interface (BCI) technologies and research, focusing on the control of motor prostheses. This is followed by an overview of the BCI robot arm control experiment and system that is the subject of this dissertation. Some of the main component parts of this system do not originate with BCI research and have limited shared history with it. Background material relating to each of these component parts is located within chapters of this dissertation that discuss them. Chapter 3 reviews the population vector algorithm that relates cortical activity to movement and have been used for BCIs. Chapter 4 discusses methods used in telerobotics and computer-integrated surgery to augment the control of robots with artificial control signals. Chapter 5 discusses the distributed representation of movement in the brain and learning in the context of BCIs. Chapter 6 includes a brief background on kinematic and dynamic control systems for robot manipulators.

2.1 Progress in BCI

Motor brain-computer interfaces are systems that translate neural activity measured in the brain into control signals that command an external device. Figure 2.1 shows a diagram illustrating the basic components of this type of BCI system. Electrical signals are recorded in the brain, these signals are processed to produce control command signals of some kind of computer or robotic effector system, then they are executed in the output device. Not shown is the feedback pathway that provides

visual or other feedback to the subject to close the control loop.

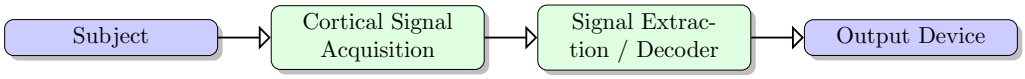


Figure 2.1: General schematic representation of brain-computer interface systems.

BCIs were first envisioned in the 1970s by Vidal while observing activity recorded by an electroencephalogram (EEG) [122]. EEGs, which measure electrical potentials from the brain at the scalp, have been used clinically for the diagnosis of epilepsy since the 1930s [111]. Attempts to use these devices for BCI had begun by 1990 [22, 69, 128] and has continued since that time to eventually enable its use by a relatively large number of human patients [71, 99, 100, 104, 129]. EEG systems are noninvasive, which has made them readily available for BCI experimentation, but the bandwidth of information that can be transmitted over an EEG interface is inherently limited by recording through layers of tissue and bone.

To bypass this limitation, a related interface method that has been used more recently for BCI is the elecrocorticogram (ECoG). ECoG was developed by Penfield and Jasper for the identification of epileptogenic zones over 50 years ago [44], but incorporated into a BCI in 2004 [66]. ECoG arrays consist of a plastic membrane in which a matrix of electrodes are embedded. This membrane sits on top of the brain surface, under the layer of dura, and records synchronized postsynaptic potentials (local field potentials) with a spatial resolution of approximately 1 cm [6]. Its proximity to the brain and density of electrodes allows for a significantly higher bandwidth signal to be communicated. Recent ECoG BCI experiments have showed the potential of ECoG for use in continuous BCI control [72, 85, 101]. This interface method is more invasive than EEG, but it allows for a significantly higher level of control signal transmission. It is also a longstanding clinical technology that does not involve penetration of the brain.

A third BCI interface method uses microelectrodes that are placed directly in the brain to record the activity of neurons individually. The use of microelectrode recordings for BCI has its roots in early experiments that used an operant conditioning paradigm to train monkeys to modulate the activity of individual neurons [26, 28]. This paved the way for mapping real-time recordings of this modulated activity to the output of some effector device. As techniques for recording individual action potentials over longer periods in the brain were developed [51], these techniques were used

in the first intracortical BCIs [16, 43, 50]. Progress in using intracortical microelectrodes for BCI control continued to 2-D and 3-D control of cursors [9, 106, 113] and robots [13, 112] with primate subjects. A study in our laboratory enabled a monkey to control a robot arm in a 4-D self-feeding task that included control of a gripper (Figure 2.2). Because of the highly invasive nature of microelectrode implants, human studies with these devices have been limited. In the experiments that have been conducted, subjects have been able to show intentional modulation of a neural signal [50] and closed-loop control of a computer interface [15, 40, 55].

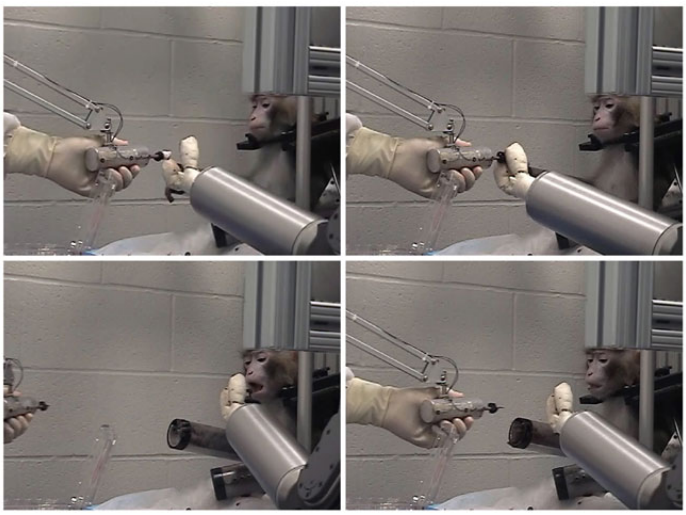


Figure 2.2: BCI self-feeding experiment of Velliste et al [121] that exhibited 4 degree-of-freedom continuous control of a robot arm and gripper using a chronic microelectrode interface.

2.2 Clinical Potential of Brain-Computer Interfaces

Brain-computer interfaces hold the potential to aid people who suffer from a wide range of neuromuscular deficits; any pathology that disrupts the normal pathway from movement intent to movement execution could potentially be remedied by a BCI. Instead of repairing the disrupted physiologic pathway, these systems allow the brain to bypass the lesion to control an artificial device. An alternative type of BCI formulation stimulates the muscles of a patient directly instead of relying on an electromechanical effector [80, 81].

The patient population for whom BCIs could provide the most benefit are those

who are paralyzed with at most one or a few voluntary functions left [8]. Described as locked-in syndrome, this condition can occur as the result of amyotrophic lateral sclerosis (ALS), Guillian-Barré syndrome, or pontine stroke. Other patients that may possess more voluntary movement but are candidates for BCI include the large population of people with spinal cord injury ($>$ 100,000 in the United States [36]), cerebral palsy, muscular dystrophy, and potentially stroke. Stroke often affects the movement of a single limb and is the main cause of long-term disability in adults. BCI may also be appropriate for prosthetic device control for amputees and people with more isolated peripheral pathologies [64, 67], especially for the control of advanced prosthetic devices that have a large number of controlled degrees of freedom.

While only a few experiments have as yet been conducted using invasive microelectrodes in humans, noninvasive BCIs have reached a stage in which a few people are using EEG systems at home on a day-to-day basis [84, 105]. EEG systems have been most effective in allowing people with little to no voluntary movement to communicate using spelling applications [10, 14], though input rates to these systems are limited to about 6 characters per minute. A higher information transmission rate of roughly 17 characters/min was achieved using an ECoG interface [12], corresponding to the higher bandwidth of information transfer using this modality. Testing of similar systems with microelectrode interfaces in primate models has not been able to produce positive results.

2.3 Models for Control of Cursors and Robots

Algorithms for the continuous control of effector devices using BCIs predict a set of kinematic variables $\boldsymbol{Y}$ from a set of neural data $\boldsymbol{X}$. Neural data usually consists of samples of spike rates from a set of neural units. Units define the electrical activity of one or more neurons that is grouped together as a single input to the model. We will briefly review two types of prediction algorithms that have been used for BCI control with microelectrode interfaces. These prediction algorithms fall into classes of discriminative and generative models, each described below.

2.3.1 Discriminative Models

Discriminative models compute predictions based on the conditional probability distribution $P(\boldsymbol{Y} = y|\boldsymbol{X} = x)$. Models of this type are not based on known features of motor neural activity, but instead on established models for prediction of time-series data [97]. The most basic filter of this type that has been used in BCI is the Wiener filter or multidimensional linear regression model [13, 40, 78, 106]. This model was originally developed during World War II for aiming antiaircraft batteries [127] and has been widely used for time series prediction since. A version of the linear filter algorithm which includes time-lagged inputs and has been used for BCI [13] is

$$\boldsymbol{y}(t) = \boldsymbol{b} + \sum_{u=-m}^{n} \boldsymbol{a}(u)\boldsymbol{x}(t-u) + \boldsymbol{\epsilon}(t)$$

In this equation, $\boldsymbol{x}(t-u)$ is an input vector of firing rates at time t and time lag u, $\boldsymbol{y}(t)$ is a vector of kinematic and dynamic variables at time t, $\boldsymbol{a}(u)$ is a vector of weights at time lag u, $\boldsymbol{b}$ is a vector of y-intercepts, and $\boldsymbol{\epsilon}(t)$ are the residual errors.

The matrix form of the filtering model is

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{A} \tag{2.1}$$

and its calibration model is

$$\boldsymbol{A} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{Y} \tag{2.2}$$

where each row in each variable is in time and each column is a data vector. Each entry in the input $\boldsymbol{X}$ holds a neural firing rate, and each column includes lagged data from each neuron so that it has a column for each lag at each neural recording channel. The calibration Equation 2.2 fits the model $\boldsymbol{A}$ to a set of observed data in $\boldsymbol{Y}$ and $\boldsymbol{X}$ so that predictions can be made using Equation 2.1.

This lag-linear prediction model is a type of moving average (MA) model or FIR filter over the input, also equivalent to a single-layer feedforward neural network. Another type of discriminative model that has been used for BCI are multilayer neural networks with feedback [16] that model nonlinear relationships between the input and output. Backpropagation learning or other methods can be used to adjust the weights

between nodes in the network to more optimally fit kinematic data to firing rates.

2.3.2 Generative Models

Generative models are movement prediction algorithms that are based on some model for reconstruction of the neural firing rates from subject behavior or environmental stimuli. Rather than determining only the conditional distribution $P(\boldsymbol{Y} = y|\boldsymbol{X} = x)$, these models first calculate some "inverse" model for generating firing rates based on observed behavior. The inverse model for a neural population is then inverted to form the "forward" model for prediction of behavior from the firing rate. While this would seem to be an extra step, this type of model has the advantage of incorporating some knowledge of state information of the firing neurons into movement predictions.

A popular type of generative model is the Population Vector Algorithm (PVA) [31, 102, 103] based on original observations of the population encoding of hand movement direction in monkeys. The prediction and calibration equations for this algorithm are described in Chapter 3 with the related Optimal Linear Estimation (OLE) algorithm that was used in our BCI system. As we will show in Chapter 3, embedded into the process of calibrating the OLE decoding model is an implicit two-step process for (a) fitting a generative model of the neural state of each unit to kinematic data, followed by (b) fitting a model to predict a kinematic state given a population of neural state estimates.

Generative encoding models similar to those used in PVA have been extended over other movement variables such as position, velocity, acceleration, and force [73, 118, 125, 126]. More sophisticated state-space models incorporate the neural or kinematic state of the system explicitly as variables and use some form of Bayesian model for state transitions in time. State-space models are based on some form of maximizing the likelihood of a model M given a set of data D. These types of models are based on the Bayes' formulation

$$P(M_i|D) = \frac{P(D|M_i)}{\sum_k P(D|M_k)P(M_k)} P(M_i) \tag{2.3}$$

where the maximally probable state-space model is found given a set of observed data, based on the fit of each candidate model M_i to the data ($P(D|M_i)$) and some prior probability of each model ($P(M_i)$). The prior is generally based on a previous

observed or calculated state. Monte-Carlo or similar methods can be used to compute the posterior probability distribution across all models for equation 2.3 [97]. Kalman filters are the primary sequential algorithms of this type that have been employed for BCI, which assume a linear relationship between input and output variables as well as Gaussian noise in the observed firing activity [3, 130]. Particle filters, which lift the linear and gaussian restrictions of the Kalman filter have also been used [11].

2.3.3 Reconstruction vs. On-line Control

An important distinction to be made when evaluating algorithms to use for BCI decoding is between the quality of reconstruction of a set of movement data from spiking activity and the performance of a decoding algorithm for on-line control. Many of the above models that incorporate a significant amount of lagged spike data, additional movement parameters, or state-space models into decoding equations show significantly better reconstructions of movement data than simpler algorithms such as those based on OLE or PVA. It is then implied that this makes these algorithms superior for BCI control. However, this ignores the role that the subject learning the decoding model plays in the development of closed-loop control skill. While these algorithms may indeed produce better reconstructions of data recorded from natural movement and potentially higher-quality offline reconstructions of BCI external effector movements, they may be difficult to learn because of their complexity. These types of algorithms have not been shown to be superior for on-line control [56].

As BCI moves towards the control of increasingly complex manipulators, correspondence between the movement of an effector device and the natural movement of the subject decreases. Because of this, some of the more complex neural decoding models that are based on reconstruction of subject movements decrease in their usefulness for BCI. Instead, the process of developing skilled BCI control becomes primarily one of learning a decoding model so that intent can be linked to action. The process of learning during development of BCI control been studied previously [13, 38]. It has already been shown that subjects can compensate for models that provide inferior kinematic reconstructions [17]. More complex models that abstract the direct connection between intent and changes in effector device output may be more difficult to control. Considering BCI as a special case of human-computer interfaces in general, it may be important to preserve a direct connection between short-term intent and device output. Intuitively, turning the steering wheel on a car is expected to

turn the wheels; superposition of a complex control model that incorporates a range of considerations into whether or not the wheels should turn would be of questionable benefit.

2.4 Brain-Computer Interface Experiment Overview

A brief overview of our brain-computer interface experiment is given here, intended to provide context for following chapters that discuss its component parts. The overall experiment is then returned to in Chapter 7, followed by a detailed explanation of the construction of each part of the BCI system in Chapter 8.

2.4.1 Brain Controlled 7-DoF Manipulator Experiment

Two monkeys were implanted with chronic cortical microelectrode arrays in their motor cortices. Signals recorded from these electrode arrays were used to drive a robotic arm in a reach-to-grasp task that required control over 7 robotic degrees of freedom (DoF) in an environment that permitted controlled collision and interaction with a target object and other objects in the experimental workspace. The robot arm used in this experiment, a Barrett Technology WAM Robot Arm, has 7 active DoF that are similar to the human arm. An attached 3-fingered Barrett Hand robot had 3 DoF of controlled individual finger flexion and 1 DoF of finger spread. In this experiment, the 3-D linear and 3-D rotational velocity of the hand in a Cartesian task space were controlled by the monkey, along with 1-D finger closure velocity of all three fingers simultaneously.

The WAM robot was mounted horizontally to approximate the joint configuration of the primate shoulder. The monkey subject was seated in a chair to the left of the arm robot. The monkey, monkey chair, experimental framing apparatus, and other objects were within the robot's reach. The left arm of the monkey was restrained during the experiment to keep it from turning around in the chair, but the right arm was left unrestrained. A cylindrical grasp target object was mounted to a 6 DoF rigid target presentation robot. The presentation robot faced the prosthetic arm such that the target could be placed at set locations and orientations within a roughly 50cm diameter workspace spanned by both robots. This setup is shown in Figure 2.3.

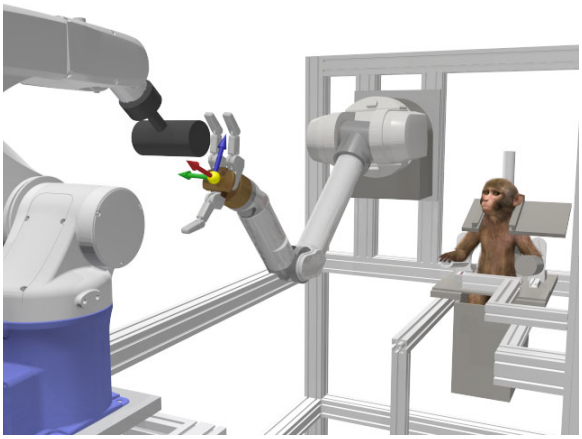


Figure 2.3: 7-DoF BCI Experimental Setup

Monkeys initiated experimental trials by pressing a button with their right hand. At that time, the presentation robot moved the cylinder to one of six target orientations while the prosthetic robot hand moved to one of six starting positions. After an audible cue, the monkey controlled the robot hand linear, rotational, and grip velocities in the grasping task. If this was completed successfully, the monkey received a liquid reward. If the trial timed out (after 12-14 seconds) or the monkey stopped trying to complete the task, a buzzer activated and the system reset for the next trial.

Experimental sessions consisted of a short brain signal decoder calibration phase followed by a subject training and practice session that lasted for as long as the monkey subject participated in the task. During the calibration period, the monkey watched the robot complete the task autonomously. Cortical activity recorded during this period was used to calibrate the decoding system. Additional "coadaptive" trials could then be performed in which the monkey's brain control signals partially affected the movement of the robot, with the cortical decoding model periodically recalibrating based on partially brain-controlled movements.

When the decoder calibration phase was complete, the system entered a training and practice phase. During this period, the monkeys received assistance from a system that adjustably reduced the amount of control error that passed from monkey brain activity to robot control commands, setting the challenge level that the monkey was exposed to in the task. Over time, the amount of assistance given was decreased so that the monkeys eventually fully controlled the robot.

2.4.2 Brain Control System Architecture

A simplified schematic of the BCI robot control experiment control pathway is shown in figure 2.4. This system was comprised of two robot control streams which were mixed to form commands that moved the robotic arm:

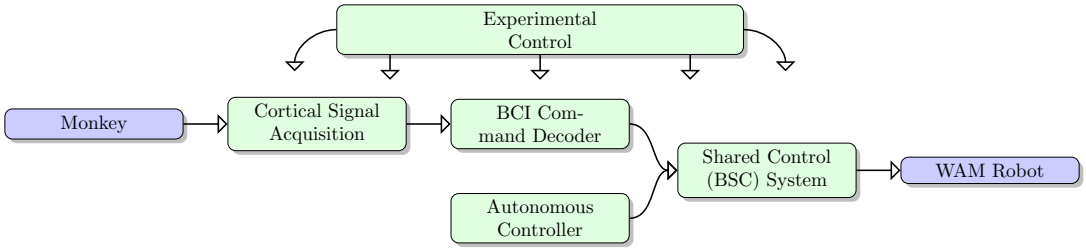


Figure 2.4: Simplified schematic of the experimental system.

Brain-Control Command Stream: In the brain-control command stream, cortical activity was recorded and decoded into robot motion commands. The cortical decoding system that translated these spike rates into robot movement commands is discussed in Chapter 3. Output commands from the cortical extraction system were emitted in 30 ms intervals and composed in the 7-DoF space of robot hand linear, rotational, and grip closure velocities.

Autonomous Control Stream: The autonomous control stream paralleled the brain-control stream, providing a set of idealized task-related robot control commands in the same 30 ms intervals. The autonomous controller generated commands based on the pose of the robot and the immediate goal of the task (reaching, orienting, gripping) to move the robot toward task completion. Because a cylindrical object can be grasped in multiple ways by the WAM robot, the autonomous controller generated multiple commands simultaneously that would each lead toward grasping.

The two command streams entered into the bimodal shared control (BSC) arbitrator module, which mixed them to form a unitary output command stream to the robot. The mixing of incoming commands was dependent on the phase of the experiment and the BCI control skill of the monkey. During the observation-based calibration phase of the experiment, the output from the arbitrator was a copy of the autonomous control stream, driving automatic completion of the reaching and

grasping trials. During coadaptive trials, robot commands were primarily automatically driven, with a small amount of control information derived from the cortical decoder mixed in. This was accomplished using an "active" shared-control method in which user and automatic control were directly combined. During the later training and practice phase of the experiment, a different "passive" shared-control method was employed in which the autonomous control was used as a template against which deviant subject control commands were treated as error and attenuated.

The passive shared control algorithm, otherwise known as "flexible fixturing", is described in Chapter 4. The role of active and passive shared control in BCI is described in Chapter 5. Composite robot commands emitted by the shared control arbitrator directed the endpoint velocity of the WAM robot. Movements of the WAM were governed by a type of kinematic-impedance control algorithm that allowed the robot to smoothly respond to a noisy stream of movement commands while controllably interacting with objects. The robot control algorithms that allowed for this behavior are discussed in Chapter 6.

Chapter 3

Cortical Decoding of Translation and Rotation

One of the most basic components of a brain-computer interface system is the cortical decoding or extraction system that translates spiking activity of neurons into device control commands. This chapter reviews the unified model for cortical decoding of 3-D rotation and 3-D translation that was used in our BCI experiment. 1-D hand aperture decoding was also performed, but this was done separately and in parallel to the decoding of whole hand movement. The use of these two decoding models together is reviewed in Chapter 7.

3.1 Neural Coding of Hand Translation

A concise and well-known generative model for the relationship between the activity of motor cortical neurons and hand movement is the "cosine tuning" model originally proposed by Georgopoulos [32]. This model describes the firing rate of individual neurons with respect to the direction of hand movements such that neurons fire at a maximal rate when the hand is moving in the "preferred direction" (PD) of the neuron. These neurons are "cosine tuned" in that this rate drops off with the cosine between the PD and the direction of movement. The PD model was originally used to describe firing rates in cortical area M1 during 2-dimensional hand translations performed by monkeys. This 2D model was later extended to translation in 3D [52].

In the PD tuning model, 3-D hand movement in direction $\boldsymbol{v}$ is related to the firing rate λ_i of neuron i by the tuning parameters $\boldsymbol{\beta}_i = [\beta_{ix}, \beta_{iy}, \beta_{iz}]$, offset by the neuron's baseline firing rate β_{i0}. The preferred direction for neuron i, the direction of movement for which its maximal firing rate occurs, is the direction indicated by the vector $\boldsymbol{\beta}_i$.

The scalar PD model is

$$\lambda_i = \beta_{ix} v_x + \beta_{iy} v_y + \beta_{iz} v_z + \beta_{i0} + \epsilon \tag{3.1}$$

or in matrix/vector form, for multiple samples of λ_i and $v_{x,y,z}$ suitable for fitting the model,

$$\boldsymbol{\lambda}_i = \boldsymbol{v} \cdot \boldsymbol{\beta}_i + \beta_{i0} + \boldsymbol{\epsilon} \tag{3.2}$$

where $\boldsymbol{v}$ is an $m \times 3$ matrix of m samples of movement direction, $\boldsymbol{\lambda}_i$ is a vector of spike rates, and $\boldsymbol{\beta}_i$ is a 3×1 preferred direction vector. $\boldsymbol{\epsilon}$ is a vector of residual values related to noise or neural activity not related to the model. $\boldsymbol{\beta}_i$ coefficients are fit by regression over a set of recorded movements and spike rates. In past studies of hand motion that have used the PD tuning model, the data used for model fitting was the averaged firing rates over whole reaches and overall reach directions, so that $m =$ the number of reaching trials.

3.1.1 Population Vector Algorithm

Relying on the PD model, the Population Vector Algorithm (PVA) [31, 103] was designed to reconstruct or predict movement directions from the firing rates of a population of directionally tuned neurons. The PVA algorithm is a voting scheme in which a predicted direction of movement $\boldsymbol{p}^*$ is formed by summing a contribution from each neuron in its normalized preferred direction, scaled by w_i^*, neuron i's activity relative to its baseline firing rate. λ_i^* is a firing rate to be used for prediction.

$$\begin{aligned} w_i^* &= \lambda_i^* - \beta_{i0} \\ \boldsymbol{p}^* &= \sum_i w_i^* \frac{\boldsymbol{\beta}_i}{||\boldsymbol{\beta}_i||} \end{aligned} \tag{3.3}$$

The use of the direction of movement alone in the PD or PVA equations constrains the representation of movement from each individual neuron to the unit sphere, so that 1-DoF of information (speed) is lost in the process. In a study by Moran [73], speed was determined to influence the PD model in both an additive way as well as with a gain component that modulated the directional tuning:

$$\lambda_i = ||\boldsymbol{w}||\,\beta_{is} + \boldsymbol{w} \cdot \boldsymbol{\beta}_i + \beta_{i0} + \epsilon \tag{3.4}$$

where $\boldsymbol{\beta}_i$ is the (non-normalized) preferred velocity, and β_{is} is an omnidirectional speed-modulated component of the neural firing rate. Using even the basic PD tuning model (Equation 3.2), speed information will be contained in the reconstruction of movement because speed-tuned neurons modulate the w_i^* of PVA (Equation 3.3). Thus, even though the 3-D PD calibration process produces 2-D preferred directions, a representation of the speed component is recovered during assembly of the movement vector using PVA. However, a gain factor must still be applied to match the velocity output of PVA to the original scale of the movement input velocity.

PVA in Brain-Computer Interfaces

In BCI experiments using the PD model and PVA, model calibration was at first performed as in natural movement experiments using hand movement recordings [113]. Later, a robot gripper was used as a substitute for the subject hand so that the direction of robot movements and averaged firing rates over movement periods were used in the PD model calibration, as in in Velliste et al [121]. In this experiment, the movement of a robotic gripper was also controlled using a 4-D version of the PD model. The distance from the starting position to a target in each movement segment (reaching, gripping, bringing the hand to the mouth) were regressed against the average neural firing rates during that segment. Each of these components (x-y-z-grip) were multiplied by a scaling factor so that the magnitude of the movement direction was about -1 to 1 in each component ([121] Supplemental Material). The velocity of the robot arm during later control was then derived from neural population activity using PVA with different scaling parameters assigned to the linear and grip DoF.

3.2 6 Dimensional Movement Coding

We now extend the 3-D PD directional tuning model to 6-D linear and rotational velocities. In our BCI prosthetic experiment, we describe the PD equation as the "inverse" model relating movement to firing rates. The calibrated inverse model is then used to obtain the "forward" model for translation of firing rates to movements in Section 3.3.2. Our model for the encoding of 6-D movements supposes that the firing rate of individual motor neurons is related to linear and rotational velocity of the hand using a 6-element movement vector

$$\begin{aligned} \lambda_i &= \boldsymbol{u} \cdot \boldsymbol{b}_i + b_{i0} + \epsilon \\ \boldsymbol{u} &= [\ \boldsymbol{v} \quad \boldsymbol{\omega}\ ] \\ \boldsymbol{b}_i &= [\ \boldsymbol{\beta}_i \quad \boldsymbol{\beta}'_i\ ] \end{aligned} \tag{3.5}$$

where $\boldsymbol{u}$ contains the 3-D linear ($\boldsymbol{v}$) and rotational ($\boldsymbol{\omega}$) movement velocities, and the preferred direction $\boldsymbol{b}_i$ for neuron i is composed of its linear $\boldsymbol{\beta}_i$ and rotational $\boldsymbol{\beta}'_i$ preferred directions. b_{i0} is the baseline firing rate of neuron i, and ϵ is residual activity. Actual movement velocity is used here instead of direction alone to simplify the relationship between linear and rotational movement parameters in the model. This takes advantage of the speed gain modulation of PD equation (3.4), preserving the full 6-DoF of the input during PD model calibration. Complications that would occur during normalization between types of movement (linear, rotational) are avoided; if linear and rotational motion are normalized together, the apparent magnitude of movement in one type of movement can change based on movement of the other type. Instead, we fit a simpler model that simply relates measured robot velocities to neural firing rates. A side effect of this is that it facilitates comparison of the regression coefficients between experiments and conditions; assuming the independence of regressors (in the brain), regression coefficient magnitudes are unaffected by the addition of other DoF representing distinct types of movement into the model.

Neural Coding of Hand Rotational Velocity

While the cortical basis of linear hand movements has been studied extensively, hand rotation has not. In a study by Kakei [46], cortical activity was found to be more related to the Cartesian direction of hand movement in space than joint movements, but particular movements at the wrist joint in this study correlated with the ori-

entation of the hand. In a more recent study by Wang et al [126], rotation of the hand in the frontal plane was studied in a 3-D reaching task. A gain-field (nonlinear multiplicative) relationship was found in the effect on neural firing rates between linear and rotational coordinates in monkey hand movements. The potential nonlinear relationship between linear and rotational motions is dealt with by a sequential PD model calibration process that is outlined in Section 3.4.

Cosine Tuning in 3D is Inner Product Tuning in 6D

In describing control models for translation, the relationship between movement directions and PDs is often described with respect to the angle between them. As rotational velocity is added to the model, it is simpler to think of the preferred direction model as simply relating the equivalent inner product of the the coefficient vector and the direction of movement with some offset b_0. Otherwise, angles between vectors representing different movement types (linear, rotational) will make little sense. Accordingly, angles derived from the inner product projection (with the inverse cosine) in spaces that relate different types of movement are dependent on the units of measurement used. Therefore, it may be more useful to think of the PD tuning model as one that relates firing rates to the inner product of a preferred direction vector and a movement vector. This is mathematically equivalent to the cosine tuning model in 3D, but much easier to understand moving to higher-dimensional decoding models.

3.3 Calibration of the 6-D Prediction Model

We now discuss the method that was used to calibrate the inverse model relating 6-D movements to single-unit firing rates, followed by the derivation of the forward model from a population of calibrated PD models. This section includes a concise matrix description of the PD and OLE calibration process, which may provide a simplified way of looking at how these algorithms function. At the very least, it facilitates direct comparison to other decoding algorithms that have been used for BCI.

3.3.1 Optimal Linear Estimation

Because of the many-to-one relationship of movement parameters to firing rates in the PD model (Equation 3.5), using the activity of a single neuron to predict 6-D movement is insufficient. Given a single neuron's firing rate, the possible encoded movement vector has 5 remaining degrees of freedom in the control space of the experiment. An equivalent way of looking at this is that the activity of each neuron is the scalar quantity describing the projection of the movement along its PD vector. PVA sums these projections over a neural population to form an estimate of the desired movement vector in the control space. It is easy to see that when the basis (PD) vectors are unevenly distributed, the resulting vector sums will be biased towards more densely represented movement directions. This problem exists in 2-D and 3-D decoding, and is only compounded for 6-D decoding.

Optimal Linear Estimation (OLE) [49, 96] is used to reduce bias in the reconstruction of the movement from the population of projections. OLE is equivalent to PVA for uniform distributions of preferred directions, but eliminates bias due to uneven ones. From Salinas et al [96], Optimal Linear Estimation estimates a control vector $\boldsymbol{v}_{est}$ from a set of firing rates $\boldsymbol{\lambda}$ as:

$$\boldsymbol{v}_{est} = \sum_i \lambda_i \, \boldsymbol{p}_i$$

where

$$\boldsymbol{p}_i = \sum_j Q_{ij}^{-1} \boldsymbol{k}_j$$
$$\boldsymbol{k}_j = \int f_j(\boldsymbol{v}) \, \boldsymbol{v} \, d\boldsymbol{v}$$

Q_{ij} is the entry in a correlation matrix $\boldsymbol{Q}$ of calibration firing rate data. The "true" movement vector corresponding to the given set of firing rates from the calibration is $\boldsymbol{v}$. Neurons are assumed to have Gaussian distributions of firing rates over $\boldsymbol{v}$, with average firing rate $f_j(\boldsymbol{v})$ when the movement vector takes on the value $\boldsymbol{v}$. The end result of OLE is that $\boldsymbol{v}_{est}$ is the best linear estimator for the desired movement direction given a set of firing rate data and movement vectors in the calibration. This is equivalent to the ordinary least-squares fit of a population of PD-modeled neurons to movements, which we will show in the next section.

3.3.2 Matrix formulations for decoding model calibration and OLE inversion

We now present a matrix formulation for the PD regression and OLE steps of decoder calibration. This is equivalent to the unit-by-unit calibration model that has been commonly used (Equation 3.1) combined with various descriptions of OLE [17, 96].

A preferred direction matrix representing $\boldsymbol{b}_i$ (in Equation 3.5) for an entire population of neurons can be determined with a single matrix inversion by arranging the problem as follows:

$$\boldsymbol{\lambda} = \boldsymbol{U} \cdot \boldsymbol{B}$$

$$\begin{bmatrix} \lambda_{11} & \lambda_{12} & \dots & \lambda_{1m} \\ \lambda_{21} & \lambda_{22} & \dots & \lambda_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_{s1} & \lambda_{s2} & \dots & \lambda_{sm} \end{bmatrix} = \begin{bmatrix} 1 & u_{11} & u_{12} & \dots & u_{1n} \\ 1 & u_{21} & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & u_{s1} & u_{s2} & \dots & u_{sn} \end{bmatrix} \begin{bmatrix} b_{01} & b_{02} & \dots & b_{0m} \\ b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix}$$

$$\boldsymbol{B} = \boldsymbol{U}^{+}\boldsymbol{\lambda} \tag{3.6}$$

where λ are measured firing rates of m units over s samples, elements u are kinematic data of n dimensions over s samples, and, b are the PD model coefficients to be fit. $\boldsymbol{U}^{+}$ denotes the generalized matrix of matrix $\boldsymbol{U}$. As $\boldsymbol{U}$ is not a square matrix, the Moore-Penrose pseudoinverse gives the equivalent to the least-squares regression over each unit. Calibration of the $\boldsymbol{B}$ matrix establishes the inverse model relating firing rate to movement. To drive the kinematics of the robot using neural activity, we derive the forward model that uses PD models of the neural population to estimate desired kinematics, based on the generalized inverse of $\boldsymbol{B}$:

$$\boldsymbol{D}^{*} = \boldsymbol{\lambda}^{*}\boldsymbol{B}^{+}$$

where $\boldsymbol{D}^{*}$ are the predicted kinematic quantities for a set of firing rates $\boldsymbol{\lambda}^{*}$ for prediction. The Moore-Penrose pseudoinverse gives

$$\boldsymbol{X}^{+} = (\boldsymbol{X}^{T}\boldsymbol{X})^{-1}\boldsymbol{X}^{T}$$

which is the "minimal OLE" solution as described by Chase [17]. This is a least-squares estimate of the kinematic data given the modeled state of the population

determined in Equation 3.6 [56] . There is no need to apply additional scaling factors in this step if movement velocities were used for the samples in $\boldsymbol{U}$.

Comparing the OLE model to the type of linear filter described in Section 2.3.1, the difference in the two approaches can be described as a difference in how error is minimized during model calibration. In the filter method, the model error is minimized over the prediction variables directly based on samples of raw spike rates from a population. In the PD/OLE calculation, error is first minimized in the generative model of firing rates during the PD calculation over the raw recorded data. In the OLE step, the least-squares error is then minimized over the prediction variables based on the modeled population of neurons.

As noted in [17], the assumption for using ordinary least squares for multiple units is that the neural activity is homoscedastic and there are no correlations between the residuals and the regressors. As these assumptions often do not hold for recorded neural data, we can perform a version of weighted least squares where we transform the data such that the assumptions of least squares hold. Given a model $\boldsymbol{B}$, then a set of movement and spike data $\boldsymbol{U}$ and $\boldsymbol{\lambda}$.

First, the residuals are formed into $\boldsymbol{R}$

$$\begin{aligned} \boldsymbol{\lambda}_P &= \boldsymbol{U}\boldsymbol{B} \\ \boldsymbol{R} &= \boldsymbol{\lambda} - \boldsymbol{\lambda}_P \end{aligned}$$

then the covariance matrix of the residuals is calculated

$$\boldsymbol{\Omega} = cov(\boldsymbol{R})$$

and generalized least-squares regression [45] is applied.

$$\boldsymbol{P}^D = \alpha(\boldsymbol{B}^T\Omega^{-1}\boldsymbol{B})^{-1}\boldsymbol{B}^T\Omega^{-1}$$

which is the "full OLE" solution from Chase et al [17], and is equivalent to a linear transformation of the data to standardize the scale of and decorrelate the residuals. The "variance only" OLE method uses only the diagonal entries in $\boldsymbol{\Omega}$, corresponding to standardizing the scale of residuals to adjust for mean firing rate differences between units.

3.4 Observation-based Model Calibration

In our BCI experiment, an observation-based paradigm was used to collect data on which the PD coefficients in Equation (3.5) were calibrated. This process was similar to that used by Velliste [121] and Wahnoun et al [123] previously in BCI, except that in our work the monkey was never trained in any physical task that originally caused the robot to move. At the beginning of a daily experimental session, the robot autonomously completed grasping trials as samples of filtered spike rates and filtered robot velocities were buffered for use in fitting the coefficients in Equation 3.5. Sampled spike rates and Cartesian robot velocities were used for model calibration instead of averaged distances to target and averaged firing rates over task periods. This simplified the process of acquiring calibration data to fit the relationship between the two observed processes, especially for experiments with multiple periods of different movements (reaching, orienting, grasping) occurring during individual trials.

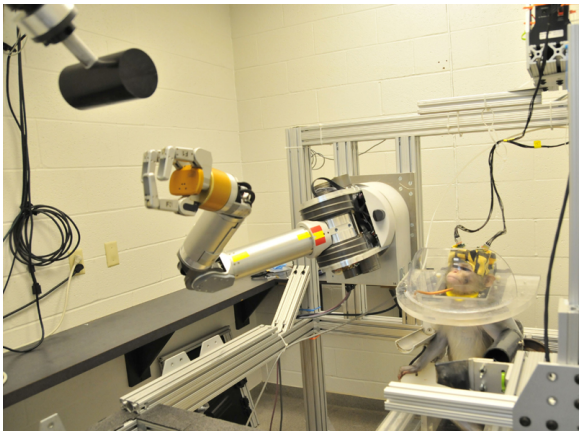


Figure 3.1: A monkey observing motion of the robotic arm.

The use of observed movements of the robot directly instead of goal-related proxies in calibration supports the notion that cortical activity used for model calibration is based upon the congruence of neural activity between action and the observation of action. Tkach et al have observed this phenomenon when a well-trained monkey observes replay of a motor task involving a robot [115, 116]. The difference in our experiment is that the monkey observes the arm moving with no learned connection to the monkey's own motor movement, so that any cortical activity is elicited by a natural response to viewing the robot alone.

Observation-based activity may be related to a mirroring phenomenon first de-

scribed by Rizzolatti et al in cortical area F5 [30]. Congruent single-unit activation was observed while a monkey performed a manipulation task and a human experimenter performed a similar task. Rizzolatti and others propose that mirroring processes are responsible for cortical activation during observation of other motor and non-motor process, including emotion [24]. These conclusions for areas outside of cortical area F5 were drawn mainly through studies using noninvasive recording methods. In one of these studies, similar cortical areas are activated when humans observe humans or robots grasp an object [74].

In our experiment, observation trials were completed in a sequence of each movement type individually; the robot first moved in the linear DoF to a point near the target. Next, it rotated to the target with no translation. This was done for two reasons: (a) isolation of movement types eliminated a potential confound if the cortical representation of linear and rotational motion interact in a nonlinear fashion and (b) isolation of movement types allowed us to evenly sample the space of targets with fewer trials. In the study previously mentioned of Wang et al [126], the gain-field relationship between cortical activity and linear and 1-D rotational hand movements would be difficult to calibrate in a timely manner for use in BCI. By calibrating each movement type individually, we removed potential interaction effects that would skew the linear model. On the other hand, we did not need to collect data over the large number of combinations of movements needed for nonlinear model calibration. While we wished to afford the monkey a natural basis for movement control, a factor that simplifies BCI is that the designer of the system implements the actual model for translating cortical activity that is applied to the robot. Thus, we could estimate the independent parameters in the model by presenting a degenerate case in which interaction between linear and rotational DoF did not take place. The resulting model could then ignore the nonlinear aspect of the relevant natural movement tuning equation that would be much more difficult to use for control of a robot.

3.5 BCI Rotational Control of a Robot Manipulator

This section reviews how control commands from the decoder were applied to movement of the robot. This section is motivated by some particular methods for controlling robot rotation that seem to arise in control schemes by default which would

make BCI robot control more difficult.

3.5.1 Parameterization of Rotation

There are multiple ways to represent 3D rotations, one of which must be chosen for use in the study of motor control and in brain-computer interfaces. There are significant issues with some of these rotation representations that are commonly underrepresented when orientation or rotation are discussed. Parameterizations of rotation such as Euler angles (yaw-pitch-roll) are not independent (Figure 3.2), and arrays comprised of Euler angle values have a nonlinear relationship with the magnitude of the rotation that they represent. Other rotation representations such as quaternions and rotation matrices each have 3 degrees of freedom but require more than 3 parameters, making them harder to integrate into straightforward control models. Because of its dependence on a reference pose, axis-angle representations (using 3-element arrays) of orientation cannot be compared using vector arithmetic. This representation of orientation also has a discontinuity at magnitudes greater than 180 degrees, further complicating large rotations (Figure 3.3).

A representation of rotation that does not suffer from these problems is the simple 3-component angular velocity vector. This representation identifies a 3 dimensional rotation command space that operates like linear velocity in that it is a consistent vector space. Its relationship to hand orientation, however, is unlike the relationship between linear velocity and position. 3-component representations of orientation cannot be subtracted to find meaningful difference vectors that represent a direction of movement. A rotational velocity that would bring a robot from one particular orientation toward another must be calculated using methods specific to each parameterization for orientation. For these reasons, the angular velocity vector, without reference to orientation, is used in the 6-D decoding model.

3.5.2 Choice of Control Coordinate Frames

The exact definition of translation and rotation to be used in the movement of a robot arm is an important factor in reducing the complexity of a BCI control system capable of producing rotational and linear movement simultaneously. We take a common point for which hand translation and orientation are controlled to be at the

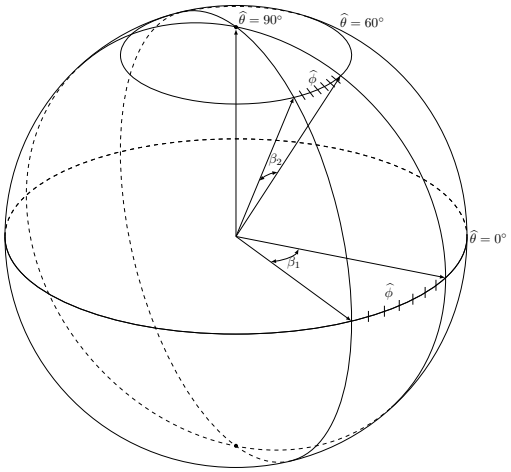


Figure 3.2: Rotations subtended by individual Euler angles are not independent. The rotation of β_1 is different than β_2 for identical values of $\hat{\phi}$ "longitude" at different $\hat{\theta}$ "latitudes".

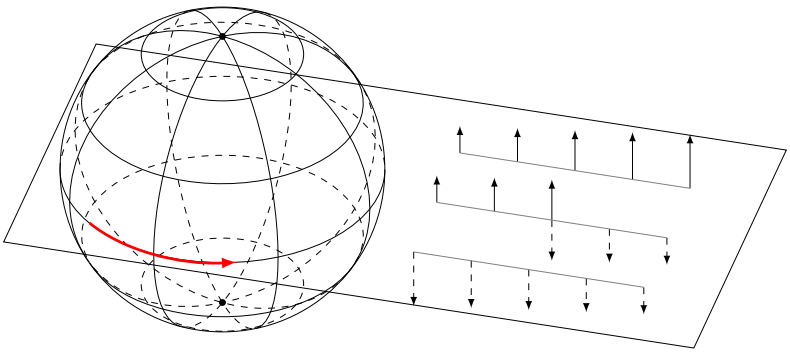
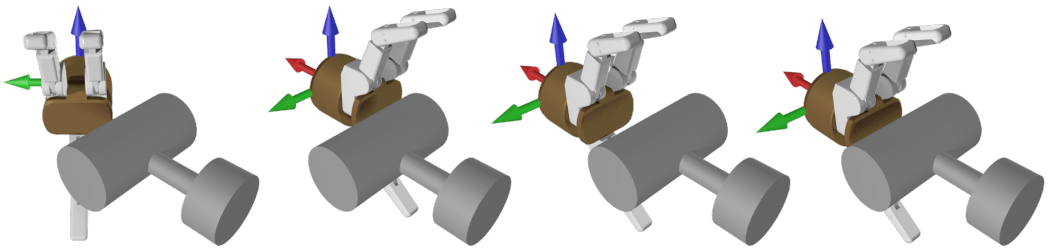
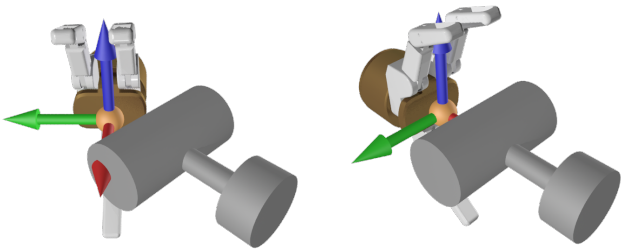


Figure 3.3: Multiple representations of a single rotation exist for 3-D rotation arrays using an axis representation. Changes in heading plotted on the unit sphere (red arc) are represented with 3 sets of arrows in scaled Euler axis representations of orientation (right). On the top and bottom are arrows of increasing and decreasing magnitudes representing the same orientations achieved during the rotation. If a boundary condition is used (middle, if the red arrow represents a $> 180°$ rotation from the origin), there is a discontinuity at the boundary.



(a) Rotation at the robot wrist requires translation to reapproximate the center of the target.



(b) Rotation near the object preserves the linear relationship.

Figure 3.4: A comparison of two methods of applying rotation to a robot, showing how rotation near the object being grasped can help to maintain independence between linear and rotational DoF.

terminal portion of the robot hand; this is a point at the intersection of the hand and an object. While the use of a single point at the effector for application of control is not a unique concept in robotics, it may have special significance for brain-controlled interfaces. In this object centered coordinate system, linear and rotational degrees-of-freedom approach independence during interaction between robot and object to be grasped; rotation about the object requires minimal linear movement to maintain approximation to it. This property has been useful in allowing isolated rotational and linear movement during the observation-based calibration phase of our primate BCI experiment. Otherwise, the use of separate places on the robot to control rotational and linear movement would create interdependence between the two reference frames. As an example, consider an arm control system where linear movement acts on a point at the base of the wrist and rotation is applied distal to it in extrinsic or intrinsic (joint angle) coordinates. If the arm has approximated an object for grasping, application of a rotation command would require additional linear movement to bring the hand back to its original position with respect to the object (Figure 3.4). While this is easily compensated for by a robot trajectory planner or controller, it may complicate BCI control of the same device.

An alternative method to providing full control over hand translation and rotation that avoids interaction between component DoFs is to control the joints of the arm directly. Intrinsic-space arm control using brain-computer interfaces in particular may present unique sets of challenges based on the specific design of the robot used. Processes such as cortical activity evoked by observation of robot arm movement used here for model calibration and underlie embodiment of the arm during control learning (Chapter 5) may be hampered when using intrinsic control schemes if the mechanical design of the robot arm differs from a natural one. In the case of directly controlling the joint angles of the 7-DoF robotic arm used in our primate experiment, primates have an abduction/adduction joint at the wrist, while the robot has an additional pronation/supination joint at the same point in the kinematic chain. While the same orientations of the robot endpoint can be achieved using either joint configuration, there is limited correspondence between the robotic and natural joint-space poses for achieving these orientations. Even though cortical activity and joint angular velocities are correlated during natural movement [93], extension of this to robotic prostheses of arbitrary design is potentially problematic.

Chapter 4

Shaped Fixtures for Shared-Control Manipulation

4.1 Introduction

The "Virtual Fixture" (VF) was introduced by Rosenburg et al [95] to describe the concept of embedded guidance or constraint within human-machine interactive systems. In a system that uses virtual fixturing, a human controls the movement of an effector robot but the virtual fixturing algorithm modulates the movement to enhance task performance. Current virtual fixturing methods have been used to guide the control of a robot towards or away from certain areas of a workspace [1, 95] or along trajectories or paths [57]. These methods are intended for applications in robotic surgery and telerobotic micromanipulation where movement of a robot into certain poses could potentially injure a patient [79] or more generally where precise or stable control is needed but where task requirements can exceed human abilities [88]. In haptic experiments, force feedback can be used to communicate the effect of the fixture to the user. Haptic virtual fixtures have been used for training users in a 2 dimensional control task by reinforcing desired movement behaviors [75].

Existing virtual fixturing applications have been defined in 2-D or 3-D translational movement tasks, in which movement attenuation or resistance occurs when the endpoint of a manipulator enters certain regions in the workspace of the robot. The interactions between the robot and fixtures are generally computed in specific ways depending on the geometry of the task. In Bettini et al [7], virtual fixturing methods

were applied across a set of geometric primitive types. In other studies, motion was constrained to a plane or line [57] depending on the relationship between the robot endpoint and task geometry.

In this chapter, we present a novel type of "flexible fixturing" (FF) algorithm that generalizes virtual fixturing over convex submanifolds of robot configuration spaces. Using FF, the robot can be constrained to movement towards or along any convex shape in pose space with a unified algorithm. In this chapter, the FF control law will be presented. Next, we will discuss the application of flexible fixturing to existing virtual fixturing control schemes. This will be followed by a simulated task that shows the operation of the FF system in a "tube-following" paradigm. Finally, we will present results from an experiment in which subjects used a joystick to control an arm robot in both linear and rotational movements with FF assistance applied.

4.2 Flexible Fixturing Algorithm

Haptic virtual fixtures have been described as a control law governing the relationship between a vector of control inputs $\boldsymbol{u}$ and a vector of output robot control commands $\boldsymbol{v}$ [57]. The relationship between the input modality and robot movement in examples of Virtual Fixturing teleoperation experiments has been admittance-type, relating forces applied to an interface joystick to output robot velocities [114], or impedance-type, where input velocity of a manipulandum is transmitted to output force of a robot [75]. In either type of control, scaled directional motions are transmitted from input to output. For instance, pushing a joystick to the right produces rightward motion in a robot, with some gain or scaling parameter between input and output systems. Control and output degrees of freedom (DoF) have been generally described in terms of linear Cartesian forces and velocities.

In the FF system, the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are n-dimensional vectors of any directional control signal. That is, they represent movement commands to the system such as forces, torques, linear or rotational velocities, or changes in joint angles. The FF control law is agnostic towards the admittance or impedance nature of the specific robot-operator system being used, while it could be easily integrated into either. Instead, we view FF as a method of adaptively gating user control signals; the relationship and scaling between the input and output movement types is assumed to be an independent problem and defined outside of the fixturing algorithm.

A matrix $\boldsymbol{D}$ is defined whose columns contain unit vectors of desired directions. Desired directions point from the current robot state towards the boundaries of a desired set of robot poses or over a range of desired robot commands. The FF system computes $\boldsymbol{v}$ from $\boldsymbol{u}$ and $\boldsymbol{D}$ as

Algorithm 1 Flexible Fixturing Algorithm

$$\boldsymbol{v} = (S(\boldsymbol{D}, \boldsymbol{u}) + \boldsymbol{C}_p K(\boldsymbol{D}, \boldsymbol{u}))\boldsymbol{u} \tag{4.1}$$

$$\mathrm{S}(\boldsymbol{D}, \boldsymbol{u}) = \widehat{\boldsymbol{D}}(\widehat{\boldsymbol{D}}^T \widehat{\boldsymbol{D}})^+ \widehat{\boldsymbol{D}}^T$$

$$\widehat{\boldsymbol{D}} = \mathrm{MaxPosSpanBasis}(\boldsymbol{D}, \boldsymbol{u}) \tag{4.2}$$

$$\mathrm{K}(\boldsymbol{D}, \boldsymbol{u}) = \boldsymbol{I} - \mathrm{S}(\boldsymbol{D}, \boldsymbol{u})$$

where

$\boldsymbol{u}$ is an $n \times 1$ vector of commands supplied as input, where n is the number of controlled degrees of freedom.

$\boldsymbol{C}_p$ is a $n \times n$ diagonal matrix of attenuation parameters in the range $0 \ldots 1$

$\boldsymbol{v}$ is an $n \times 1$ vector of output commands.

$\boldsymbol{D}$ is a $n \times p$ matrix of p desired command column vectors.

S is the projection of $\boldsymbol{u}$ onto the positive desired command vectors.

K is the portion of $\boldsymbol{u}$ that is orthogonal to $\boldsymbol{S}$. $\boldsymbol{D}$ can be of any rank ($\widehat{\boldsymbol{D}}$ will always be returned full rank). An empty $\boldsymbol{D}$ matrix is taken as $\boldsymbol{D} = \boldsymbol{0}$.

Algorithm 2 Maximum Positive Span Basis Algorithm

function MaxPosSpanBasis($\boldsymbol{D}, \boldsymbol{u}$)
  if $\boldsymbol{D}$ is empty **then**
    return $\varnothing$
  end if
  $\boldsymbol{P} \leftarrow \boldsymbol{D}^T \boldsymbol{u}$
  $q, index_q \leftarrow max(\boldsymbol{P})$
  if $q > 0$ **then**
    $\boldsymbol{D}_q, \boldsymbol{D}_{-q} \leftarrow$ column $index_q$ in $\boldsymbol{D}$, remaining cols in $\boldsymbol{D}$
    $\boldsymbol{u}_{-q} \leftarrow \boldsymbol{u} - q\,\boldsymbol{D}_q$
    return append($\boldsymbol{D}_q$, $\boldsymbol{MaxPosSpanBasis}(\boldsymbol{D_{-q}}, \boldsymbol{u}_{-q})$)
  else
    return $\varnothing$
  end if
end function

The FF algorithm is derived from existing virtual fixturing algorithms [7, 57], but replaces the projection of a command vector onto the *span* of a set of vectors with the projection to the *maximal positive span* of these vectors. The positive span, as originally described by Mason [70], is $[\boldsymbol{D}] = \sum a_i \boldsymbol{z}_i \mid a_i \geq 0, \boldsymbol{z}_i \in \boldsymbol{D}$. The maximal positive span takes into account the non-orthogonal component column vectors of $\boldsymbol{D}$ so that the fewest possible vectors from $\boldsymbol{D}$ are used to describe the overall projection. That is, $\widehat{\boldsymbol{D}}$ is the set of column vectors from $\boldsymbol{D}$ with the largest magnitude of unique positive projections onto $\boldsymbol{u}$. The *MaxPosSpanBasis* function (Algorithm 2) consists of a novel recursive positive span vector determination algorithm that works by finding the $\boldsymbol{D}$ vector that has the greatest positive projection onto $\boldsymbol{u}$, subtracting that projection from the input, then recursively calling the function on the residual and the set of unselected vectors. The algorithm ends when the remainder of $\boldsymbol{u}$ no longer has a positive projection on any vectors remaining in the set, at which time all vectors found by the algorithm are returned.

The FF algorithm separates the command vector $\boldsymbol{u}$ into two parts, (1) The projection of the command onto the vectors in $\boldsymbol{D}$ that maximally positively span the input, and (2) The portion of the command orthogonal to the first part. The positively spanned portion of the input is preserved in the output, while the remainder of $\boldsymbol{u}$ is attenuated by the matrix $\boldsymbol{C}_p$ with diagonal elements $c_{p,i}$, $1 \leq i \leq n$, which vary between 0 and 1 to control the amount of error that can propagate from the user command signal to the output.

As long as $\boldsymbol{D}$ does not positively span $\mathbb{R}_n$, motion away from the convex subspace bounded by the columns of $\boldsymbol{D}$ will be attenuated. In practice, $\boldsymbol{D}$ vectors can trace the outline of a target command set for robot motion. The target command set can guide motion towards or along arbitrary shapes and paths in the configuration space of the robot, some examples of which will be illustrated in Section (4.2.1).

All of the inputs to the FF algorithm are assumed to change over time. In addition to changes in the command vector $\boldsymbol{u}$ each time that the algorithm is evaluated, $\boldsymbol{D}$ and $\boldsymbol{C}_p$ can change as a function of robot state, task parameters, and time. Thus, $\boldsymbol{D}$ here is really $\boldsymbol{D}(\boldsymbol{\Omega})$ and $\boldsymbol{C}_p$ is actually $\boldsymbol{C}_p(\boldsymbol{\Omega})$, where $\boldsymbol{\Omega}$ is the set of robot and task parameters used in a particular application of FF at a particular time.

Some notes about use of the FF algorithm:

1. Rotational velocity and torque vectors can be used in the FF algorithm as easily

as linear velocities or forces. As long as some vector space is defined on the $\boldsymbol{u}$ and $\boldsymbol{D}$, the FF algorithm is valid.

2. The algorithm will not work as written if the set of desired commands is not the smallest region bounded by the vectors in $\boldsymbol{D}$ (e.g. spread > 180 degrees in 2-D); in this case, the definitions of positive span and kernel can simply be interchanged (see Section 4.2.1).
3. The speed of the algorithm is dependent on the size of p and n, but is generally fast enough for real-time computation. An unoptimized C language implementation of the FF algorithm tested with $n = 8$ for many sets of randomly generated $\boldsymbol{u}$ and $\boldsymbol{D}$ matrices executes in under 5 ms for $p = 100$ on contemporary PC hardware.

4.2.1 Selection of D

The effect of the FF algorithm on incoming control commands is highly dependent on the selection of the $\boldsymbol{D}$ matrix. Here, we show a variety of examples of constructing $\boldsymbol{D}$ matrices for different purposes and their effect on robot control output.

Point Targeting

For the simple case of point targeting, as in Figure 4.1, $m = 1$ so that $\boldsymbol{D}$ has a single column; control commands are filtered through an anisotropic compliance such that portions of commands not pointed towards the target are attenuated by factors in the matrix $\boldsymbol{C}_p$. If $m = 2$ and $\boldsymbol{D}_2 = -\boldsymbol{D}_1$, movement both towards and away from the target point can occur freely. We say that the point target is a "control point", such that the $\boldsymbol{D}$ matrix vector points from the current robot configuration towards that control point.

Region Targeting

For region pursuit towards convex manifolds, control points can be determined at vertices or prominent points on the boundary of the shape, and p is set to some number that sufficiently captures the precision with which fixturing is desired. Figure 4.2 shows one method of determining control points for a cube-shaped spatial target.

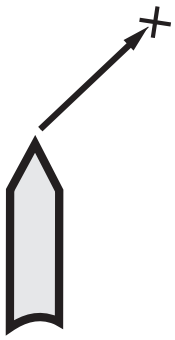


Figure 4.1: Illustration of a $\boldsymbol{D}$ matrix consisting of a single column vector directed from the end of the manipulator to a point

Directional command vectors are then formed from the current robot state to the target to form the column vectors in $\boldsymbol{D}$. There are no requirements for what points are taken; points in the interior of the target shape or proximal to each other work just as well as any other, they are just less descriptive of the target region bounds. Convex hull algorithms work well within this context for finding a good set of target points on the surface of a set of target robot poses ("target cloud"). If the robot is in a configuration on the interior of a target region, the robot will move freely if enough spread out control points are chosen such that $\boldsymbol{D}$ spans the control DoF, as in Figure 4.3.

Separation of Subspaces

Special considerations may apply when FF is applied over distinct types of movement, for instance both linear translation and rotation. Consider a case analogous to Fig. 1 of point target pursuit in the 6-D control space of both linear and rotational velocities. A vector from the current robot state to the target control point is formed. In the linear DoF, those 3 components of $\boldsymbol{D}$ are formed using the difference between the current and target location. Some method of finding the vector (axis of rotation) rotational direction to the target is also defined; this method is dependent on the representation of orientation that is used. A velocity command aimed toward the target in any of the 6 controlled dimensions will project onto that 6-D vector, so that for instance a linear velocity command alone will produce a rotational velocity command directed towards the target, at some relative scale dependent on the units

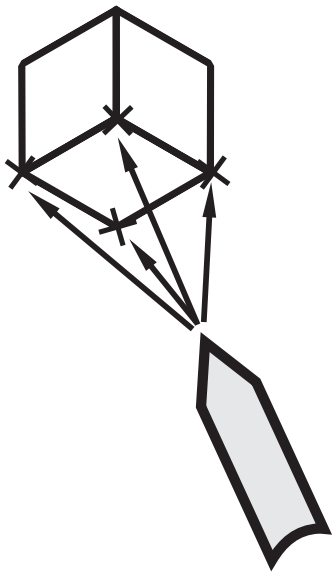


Figure 4.2: Illustration of a 4-column $\boldsymbol{D}$ matrix directing motion towards a cube. Vectors from the manipulator to the control points (X) define the range of desired directions.

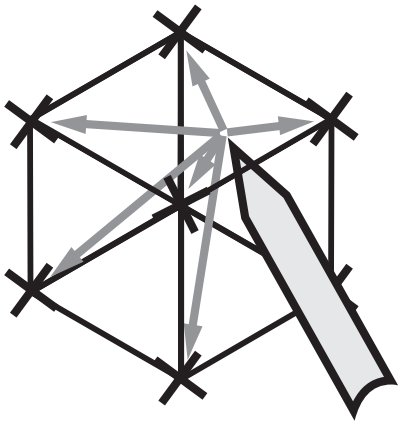


Figure 4.3: Column vectors of $\boldsymbol{D}$ point at the corners of the cube, allowing free motion in its interior.

of measurement used for each type of movement. This may or may not be the desired behavior of the system, depending on the application. If it is not, $\boldsymbol{u}$ and $\boldsymbol{V}$ can be separated into K subspaces, each of ν_k dimensions so that $\sum \nu_k = n$. The number of vectors in $\boldsymbol{D}$ can then be expanded such that $p' = Kp$. Elements of the p vectors corresponding to each k are set to zero except for the rows corresponding to the control dimensions of ν_k. This causes the FF algorithm to handle each "domain" of movement individually, so that for instance in the 6-D case commands directed at the target in linear dimensions does not generate command velocities towards the target in the rotational dimensions. If coupling is desired between the different types of movement, then the motion domains need not be separated.

Path Following

To place a fixture that constrains the robot to follow a curved path in space, columns in $\boldsymbol{D}$ can be set to point towards the nearest point on the path and a point further along it in the desired direction of motion. Bidirectional freedom is gained by setting $\boldsymbol{D}$ vectors to point to control points forward and backward on the path. For both types of path following, if the robot leaves the path these control points lead the robot back to it while allowing some ability to progress along it simultaneously. The tightness with which the path is followed is dictated by the proximity of the control points to the closest point on the path (Figure 4.4). Additional control points between the closest robot point and outer control points may be needed if the path is sufficiently curved (Figure 4.5).

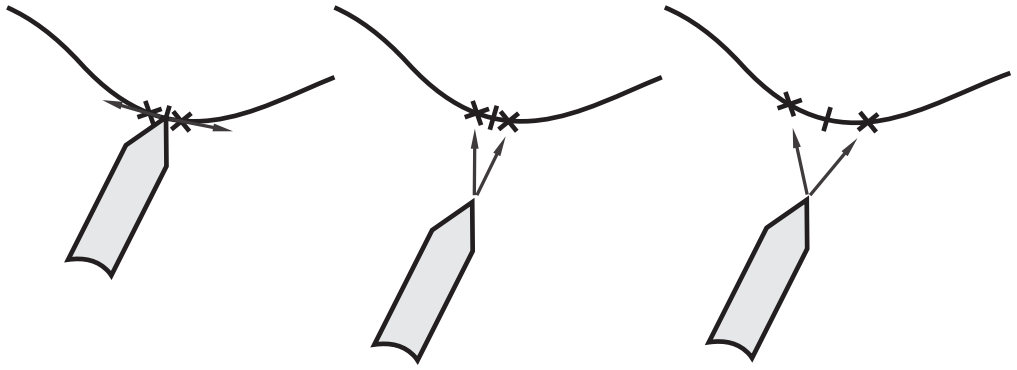


Figure 4.4: (a) Tight bidirectional path-following with the manipulator on the path, (b) the same control points guide the robot when the manipulator leaves the path, and (c) loose path following.

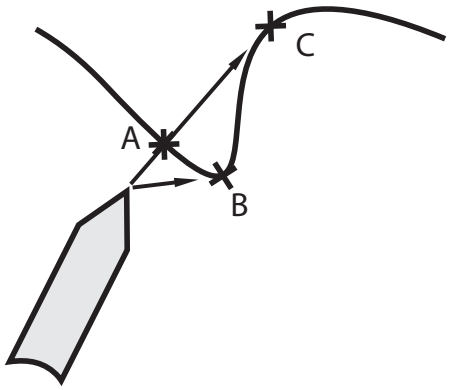


Figure 4.5: A situation where intermediate control points should be used in a path following task. The closest point on the path to the manipulator is point A. If point C is the only additional control point selected for path following, the manipulator cannot follow along the curve. To account for the high curvature of the path, an additional control point (B) is added.

It is straightforward to combine region pursuit and path following if it is desired that the robot stay within a "tube" leading to the target. Control points are taken as usual at the boundaries of the target tube (Figure 4.6). The distance along the target tube at which control points are determined controls the tightness with which staying inside the tube is enforced, as in the case of the curved line path. Movement away from the tube segment inside the $\boldsymbol{D}$ vectors pointed toward the control points is attenuated. A simulation of tube following is contained in Section 4.3.1.

Region Avoidance

Region avoidance is the opposite problem of region pursuit, where the manipulator is directed to avoid a particular region. Like region pursuit, control points are found that span the allowed area of the command space. One potential difference is that this application may more commonly require reversing the positive span and kernel definitions if more directional commands are allowed than restricted, as in Figure 4.7. Another is that for finite forbidden regions, once the manipulator enters the region it will allow all movement, because movement in any direction will point toward the region boundary. To enforce movement out of the region in a particular direction, a chosen subset of the control points must be used to direct the manipulator out in a

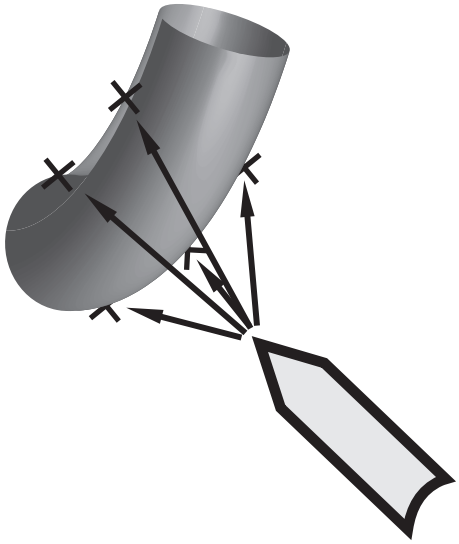


Figure 4.6: Illustration of control points and $\boldsymbol{D}$ vectors for a tube following application.

specific way.

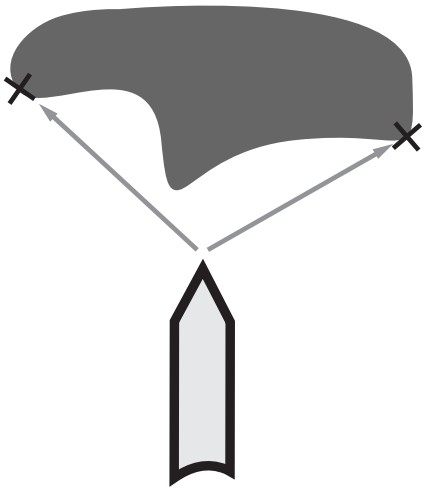


Figure 4.7: The definitions of Span and Kernel can be swapped in this case to support the avoidance of small regions.

4.2.2 Selection of C_p

In the FF system, $\boldsymbol{C}_p$ controls the per-dimension error admittance of the system. $\boldsymbol{C}_p$ is comprised of diagonal elements c_p. While each element can be set individually, for most applications they will be set to the same value within each movement domain (e.g. each subspace of the control degrees-of-freedom governing for example linear or rotational movement). We write $c_p = .5$ here to mean all of the relevant diagonal elements of $\boldsymbol{C}_p = 0.5$. Between domains, however, differentially setting $\boldsymbol{C}_p$ allows different levels of assistance based on the nature of the different types of movement in a task.

In applications where FF is used to keep a robotic manipulator in a specific region for safety, $\boldsymbol{C}_p$ values will be set to "hard" fixturing, at or near 0. For collaborative applications where FF guides the system without fully dictating movement trajectories, a "soft" virtual fixture can be set with each $c_p > 0$. Because $\boldsymbol{C}_p$ itself does not govern admittance inside a target region, these parameters can stay constant throughout execution of a task without other methods of detecting entry of the robot into an allowed or forbidden region. If needed, however, c_p values can be adjusted dynamically. If c_p values remain relatively constant throughout task execution, it represents a constant level at which error in the control signal is discarded. One application of the FF algorithm is as a training mechanism to help a user to complete a task using a robot. In these applications, the highest c_p level at which the user can complete the task can be used as an estimate of control skill; when an operator can complete a task with no assistance from FF, they are proficient in using the device to complete the task.

4.3 Validation

4.3.1 Tubular Path Following Simulation

A simulation of a tube following task was created to validate the operation of the FF algorithm. In this simulation, a curved tube admittance region was created, centered on a path following the equation $y = x^{0.5}, z = x^{0.3}, 0 \leq x \leq 0.3$ with radius of 0.05. A simulated robot endpoint was initially placed at a location outside the tube. At each time step in the simulation, the closest point on the tube to the position of the

robot was found and a set of control points were set on the circumference of the tube at 15 degree intervals around the tube at that location (Figure 4.8). Another set of control points were added at circumferential locations at a distance of 0.01 further along the tube, forming a unidirectional tube following application of FF. Columns of $\boldsymbol{D}$ were set to vectors leading from the current robot position to each control point. At each time step t, a simulated user issued a velocity command. This command was modulated by the FF system and the resulting velocity was added to the position to obtain the new position at time step $t+1$, so that the change in the robot position followed the update rule:

$$p(t+1) = p(t) + FF(\boldsymbol{D}, \boldsymbol{u}(t))\delta t \tag{4.3}$$

$$FF(u) = (S(\boldsymbol{D}, \boldsymbol{u}) + K(\boldsymbol{D}, \boldsymbol{u}))\boldsymbol{u} \tag{4.4}$$

where FF is the application of the flexible fixturing algorithm with a current input and $\boldsymbol{D}$ matrix, and p is the position.

Three sets of simulation results are shown using different types of control inputs to the FF algorithm. The robot starts at the same position in each figure. Control is evaluated for 250 time steps in each simulation. Three trajectories are shown in each figure, corresponding to identical command sequences but different c_p entry levels.

In Figure 4.8, the control input to each trajectory was a random walk; commands at each time step in each dimension were selected from a uniform distribution in the range $[-0.01 \ldots 0.01]$. The control points and vectors generating the columns in $\boldsymbol{D}$ at the first time step are indicated in this figure. The path of the trajectory resulting directly from the input commands ($c_p = 1$) wanders away from the target path. The path corresponding to $c_p = 0$, which preserved only movements towards the target region proceeds more or less directly towards it while allowing some variability of movement within the range of the target target directions. Once entering the target region, the target wanders in the tube but with a bias towards progression to the right. If the bidirectional path following command points were used, this would have been a random walk bounded by the tube. The path corresponding to $c_p = 0.5$ discards half of the vector sum of movements not directed toward the target region. Accordingly, it displays more randomness than the red trajectory but is still biased towards the target tube.

Figure 4.9 illustrates trajectories resulting from a command issued at each t with

a similar random walk, plus a bias vector leading from the current position towards the mean of the control points at a magnitude of 0.005. This kind of control input would result from an operator issuing correct commands, but using a very noisy input system. Because the on-target bias vector is always preserved in FF output, each path makes it into the target region during the simulation. However, deviation from the direct path is decreased in proportion to FF attenuation levels.

In Figure 4.10, a bias vector is added to the command directed at the control point centroid. This vector is in the direction of the cross product between the mean $\boldsymbol{D}$ vector and the tube central path tangent vector. The FF system reduces the systematic deviation from the direct path to target in this case. Once inside the target region the movements were not attenuated and followed an identical corkscrew pattern due to the bias applied.

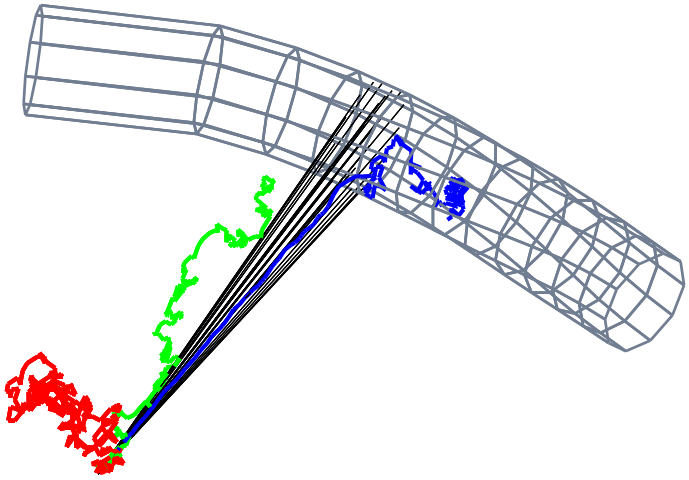


Figure 4.8: Results from simulation of a tube pursuit application with random walk input. The mesh tube is the desired region. Vectors leading from the starting position to the tube define the columns of $\boldsymbol{D}$ used in at that time step. The red trace is the integrated raw command signal and output if entries $c_p = 1$. The green trace is the output for $c_p = 0.5$, and blue shows $c_p = 0$.

4.3.2 Region Pursuit - Subject Control Experiment

The fixturing system was used in a prototype prosthetic arm control experiment in which people controlled a robot arm in 6-D with a hand-held controller. The overall

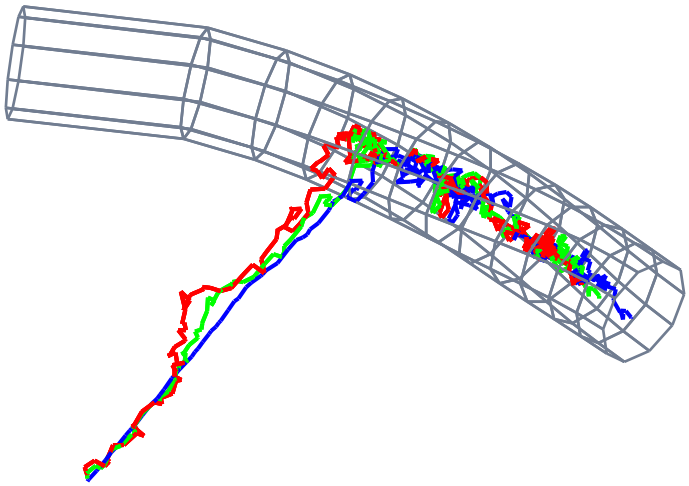


Figure 4.9: Results from simulation with directed input plus random error. The simulation is identical to 4.8 besides the control input. Bias vs. simulation amounts.

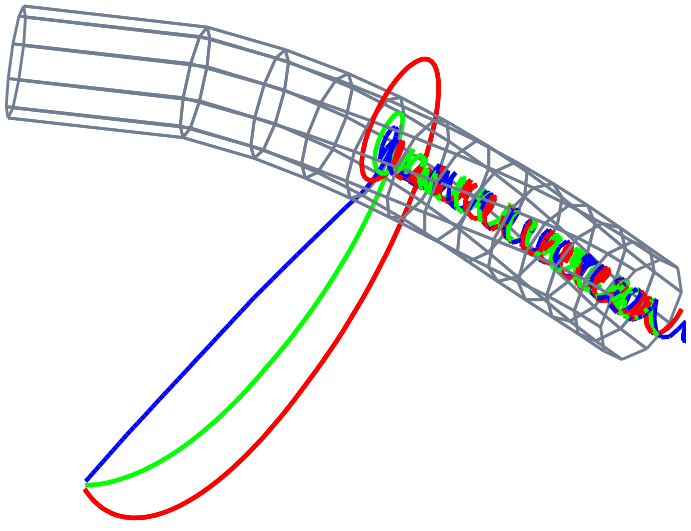


Figure 4.10: Results from simulation with ideal command perpendicular bias applied to control at each time step.

purpose of this experiment was to compare the control of a robot by humans with hand-held controllers with control of the same robot performed by monkeys using brain signals for control. Here we examine the performance enhancement characteristics of the flexible fixturing control algorithm. The experimental trials were approved by the Institutional Review Boards at the university of Pittsburgh and Carnegie Mellon University.

A Barrett WAM robotic arm (Barrett Technologies, Cambridge, Mass., USA) was mounted across from a DENSO industrial-style 6-DoF arm (Figure 4.11). A cylindrical object was mounted at the end-effector of the DENSO robot to serve as a control target. Subjects sat in the room with the robots and controlled the WAM using a handheld game controller (Sony Sixaxis Controller). 6 proportional controls were output from the controller. 4 DoF input came from the movement of two thumb-controlled joysticks, and an additional 2 inputs were provided by two sets of pressure-sensitive buttons. The control space of the robot was the set of 3-D linear and 3-D rotational velocities of a Barrett hand mounted at the end of the WAM. The fingers of the Barrett hand were not controlled in this experiment. One of two mappings was applied to relate the input and output control DoF. Each input control was mapped to a command that spanned either the 3 linear or 3 rotational DoF, so that movement in 1 input DoF produced a linear or rotational control vector pointed diagonally with respect to the "floating" coordinate axis in Figure 4.11.

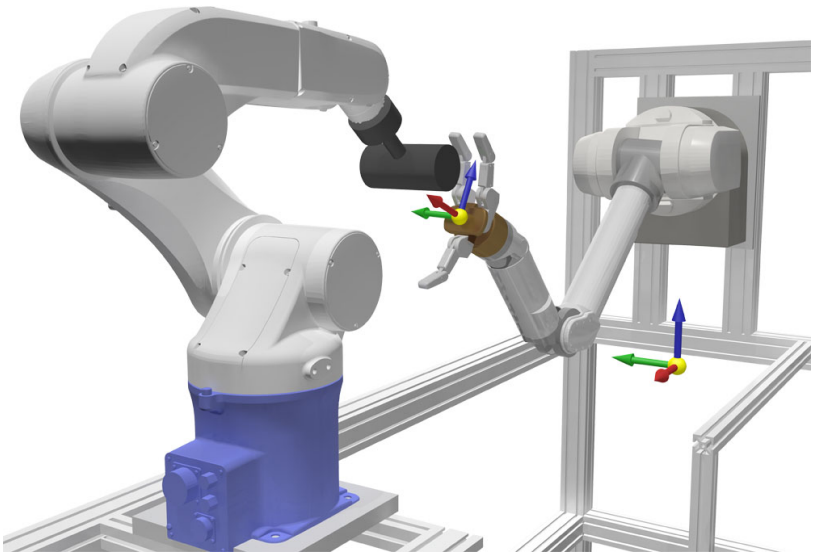


Figure 4.11: Robotic setup for human subject control task.

At the beginning of each experimental trial, the WAM robot moved automatically to one of six starting positions on the sides of a ≈ 40 cm cube in a forward facing orientation (Figure 7.2). The targeting robot simultaneously moved the cylinder to one of six orientation targets near the center of the workspace. Each orientation target was rotated to 40 degrees away from direct opposition of the forward-facing hand. When the robots had moved into their respective start positions, the user was given control of the WAM robot. Subject movement commands from the controller were sampled at 30 ms intervals and were passed to a software module running the FF algorithm. Output from the fixturing module was passed to a low-level robot controller that implemented a control system that output the desired linear and rotational velocities of the robot hand.

Each trial consisted of the subject attempting to move the WAM hand from its starting position to a point in front of the cylinder center, rotated to match the orientation of the cylinder. If the hand endpoint entered within 12 cm of this point and rotated within 40 degrees of the correct rotation, the trial was successful. After a 5-7 second timer elapsed, the trial was registered as a failure and the robots reset for the next trial.

At the beginning of each experimental trial, a software module generated a set of FF control points for the current target. These control points were a set of 6-D points on the boundary of the target range of successful robot poses. Points in the linear DoF were generated within 12 cm of the target, and points within the rotational DoF were within 40 degrees of the target using a yaw-pitch-roll parameterization of the target orientation. While it is possible to optimize the spread of control points around the target sphere, we used a set of random points on the sphere surface to better simulate how irregular shapes are compatible with the FF system.

At each 30 ms iteration of the FF system, vectors representing commands leading towards these control points were generated by the FF module and used as the columns in the $\boldsymbol{D}$ matrix. Linear command vectors were generated to point from the current robot position to the control point positions. Rotational command vectors were generated by converting yaw-pitch-roll control points to a 3x3 orientation matrix, then finding the axis-angle difference between the 3x3 robot orientation and the control point orientation using Rodrigues' rotation formula. Linear and rotational evaluation of FF occurred in separate subspaces as described in Section 4.2.1.

An example of the target space and randomly generated control points used

for linear FF evaluation are shown with the corresponding vectors that form the $\boldsymbol{D}$ matrix in Figure 4.12. FF output from to a sample input at different c_p levels is also shown. The equivalent figure for evaluation of FF over rotational DoF is in Figure 4.13. In this figure, the rotational $\boldsymbol{D}$ vectors are shown that outline a desired set of rotation commands.

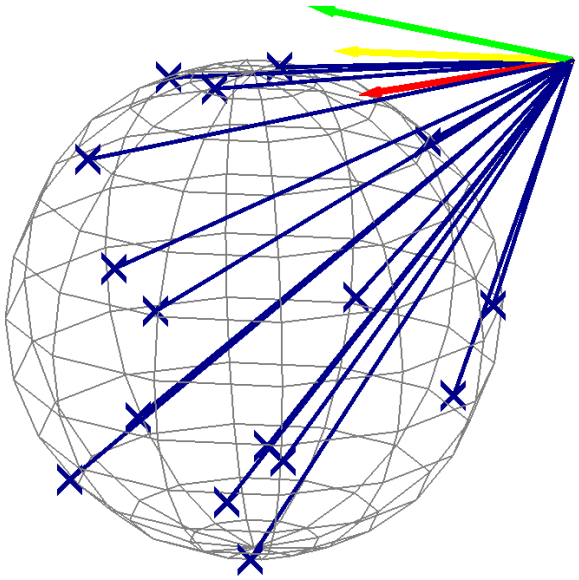


Figure 4.12: Illustration of region pursuit with spherical target in experimental task. Vectors lead from the current position of the robot to control points on the sphere (X). The green vector shows a control input vector and control output if entries in $\boldsymbol{C}_p$ are 1, yellow the output vector if entries in $\boldsymbol{C}_p = 0.5$ and red for entries in $\boldsymbol{C}_p = 0$.

4.3.3 Results

9 subjects who had not previously controlled the robot performed 80 experimental trials with the FF-enabled robot control system. c_p values for all 6 DoF were initially set to 0.2 such that 80% of control error was attenuated (very low $\boldsymbol{C}_p$ values were found to be not helpful because the robot did not move much even though the subject was providing input). During the first 60 trials, $\boldsymbol{C}_p$ was increased at a constant rate per trial until the robot was fully controlled by the subject on the 61st trial.

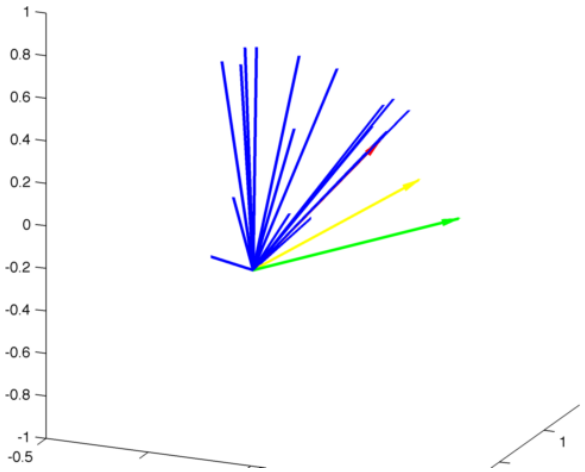


Figure 4.13: Illustration of region pursuit over a space of rotations. Control points on the target are in the space of yaw-pitch-roll and are not shown, but form a spherical range of targets in that space. The green vector shows a control input vector and control output if entries in $\boldsymbol{C}_p = 1$, yellow for $c_p = 0.5$ and red for $c_p = 0$.

A robot simulator was developed that models the dynamic behavior of the 7-DoF robot in response to velocity commands. Simulated linear robot trajectories formed from the starting position and velocity command sequences in an actual trial is shown with solid red lines in Figure 4.14, along with simulated resulting trajectories from application of FF at c_p values set to 0.5 and 0, with the target space (sphere). While the recorded trajectory and the simulations cannot be compared fairly due to the closed- vs. open-loop nature of the control taking place, this illustrates how FF reduces the error during trials.

An error measure applied to linear control trajectories was the deviation from the direct line from the starting position to the target. Error in rotational trajectories was generated by generating a direct path from starting position to target using the Slerp algorithm [107], then finding the rotational deviation from the closest rotation on the path. This application of FF did not attempt to return the robot to this direct path once leaving; however, it does give an idea of how far from ideal each trajectory was. Example linear and rotational error profiles from individual trials in two sessions are shown in Figure 4.15 for different levels of c_p values. While FF reduced the error applied to the robot in each time step, the actual trajectory error overall was still

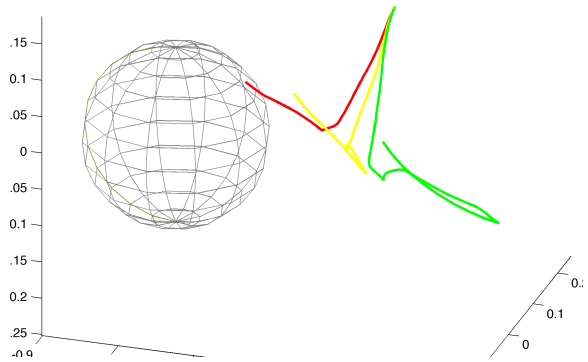
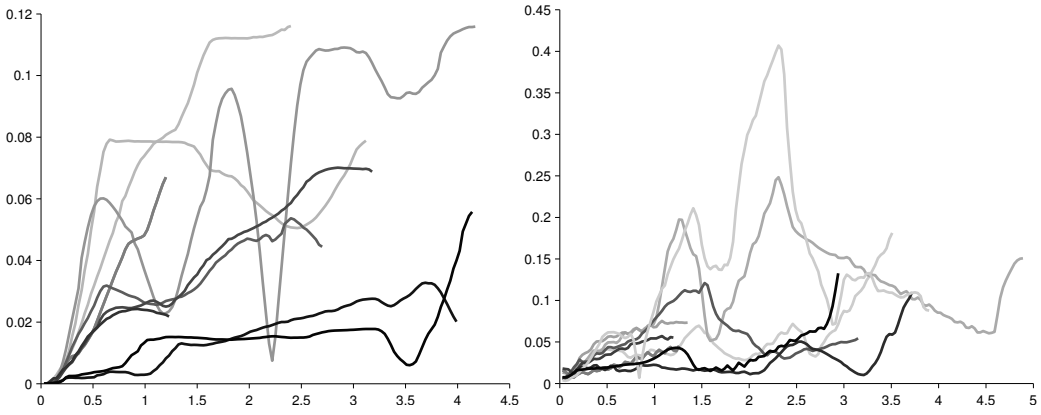


Figure 4.14: Robot trajectories generated from user commands with different levels of assistance. The mesh sphere shows the target region. Trajectories with admittance values of (c_p) of 1 (green), 0.5 (yellow), and 0 (red) are shown.

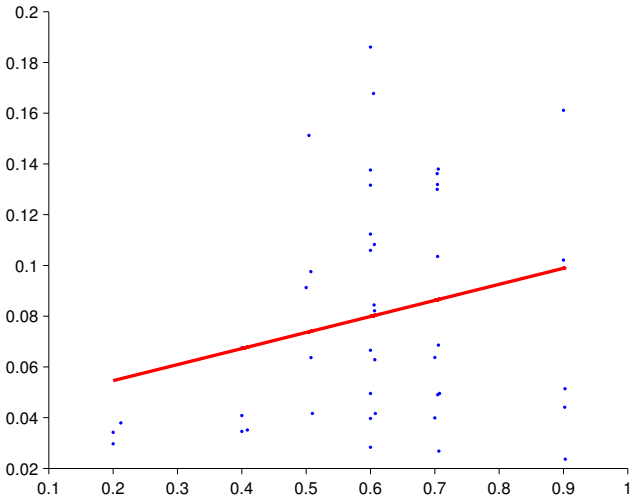
heavily dependent on the control skill of the subject for value of $c_p > 0$.



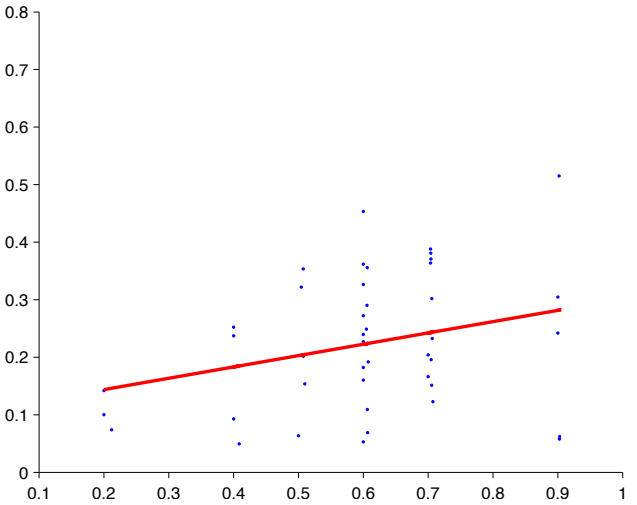
(a) Linear Path Deviation (b) Rotational Path Deviation

Figure 4.15: Example error profiles in linear and rotational control dimensions for different levels of c_p entries over single successful reaching trials. The X-axis shows the elapsed time of the trial, while the Y-axis shows the deviation from the direct path to the target in meters (a) and radians (b). The shading of the lines increase with increasing FF attenuation level (decreasing c_p), ranging evenly over the span $0.8 \ldots 0$.

FF had an overall effect on reducing the average trajectory error in experimental trials, even though the subjects had more experience in controlling the robot as the c_p

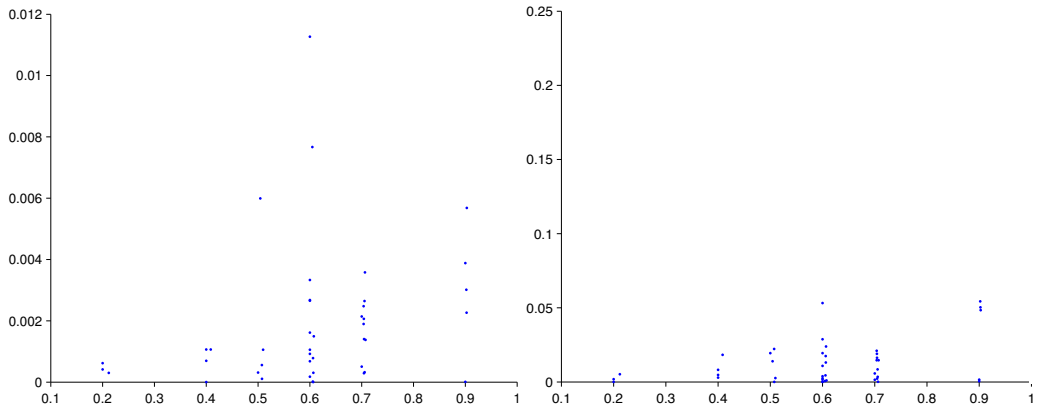


(a) Linear deviation at different c_p levels. Regression slope = 0.09, $p < 0.001$



(b) Rotational deviation at different c_p levels. Regression slope = 0.22, $p < 0.001$

Figure 4.16: Trial average deviation in meters (a) or radians (b) at different c_p levels (X-axis). Each data point is the average deviation for one trial.



(a) Linear DoF (b) Rotational DoF

Figure 4.17: Residuals from Figure 4.16, showing generally increasing range at higher c_p levels

level was lowered as trials progressed. Linear and rotational deviations were averaged over individual trials for all sessions and compared to the level of all c_p in each trial. Figure 4.16 shows a plot of this relationship with best-fit lines for trials where all $c_p \neq 1$. $c_p = 1$ trials were not included in the regressions because many more trials were conducted at the end of sessions in this condition than trials at other individual levels of $\boldsymbol{C}_p$, which would skew the fit of the model. The relationship between c_p level and error variance is shown in Figure 4.17, which suggests decreased variability between trials at decreased levels of c_p.

Fixturing reduced overall error in experimental sessions. Tabular results of comparing mean trial deviations for individual sessions and overall are in Table 4.1 and Table 4.2. Application of FF significantly reduced error overall, at higher levels and at greater significance in each individual session at higher levels of fixturing assistance.

	Mean Err Dev				p Value No Help vs.			
Subject	No Help	High $\boldsymbol{C}_p$	Med $\boldsymbol{C}_p$	Low $\boldsymbol{C}_p$	High $\boldsymbol{C}_p$	Mod $\boldsymbol{C}_p$	Low $\boldsymbol{C}_p$	any $\boldsymbol{C}_p$
1	0.126	0.089	0.086	0.079	0.095	0.075	**0.031**	**0.006**
2	0.149	0.156	0.074	0.053	0.686	**0.000**	**0.000**	**0.001**
3	0.112	0.085	0.081	0.040	0.125	0.105	**0.000**	**0.001**
4	0.093	0.091	0.081	0.057	0.865	0.312	**0.001**	0.088
5	0.142	0.116	0.101	0.047	0.144	**0.012**	**0.000**	**0.000**
6	0.114	0.105	0.107	0.050	0.537	0.659	**0.000**	**0.027**
7	0.113	0.100	0.061	0.069	0.408	**0.002**	**0.007**	**0.002**
8	0.103	0.100	0.070	0.063	0.781	**0.001**	**0.000**	**0.004**
9	0.147	0.126	0.097	0.061	0.398	**0.032**	**0.000**	**0.003**
All	0.122	0.108	0.084	0.058	**0.017**	**0.000**	**0.000**	**0.000**

Table 4.1: Left, average linear deviation from straight trajectory for c_p entries at different levels. High, medium, and low c_p levels are in the range $1 > c_p \geq 0.733$, $0.733 > c_p \geq 0.467$, and $0.467 > c_p \geq 0.2$, respectively. Right, p-values for t-test of differences in mean deviation, p values < 0.05 are in bold.

	Mean Err Dev				p Value No Help vs.			
Subject	No Help	High $\boldsymbol{C}_p$	Med $\boldsymbol{C}_p$	Low $\boldsymbol{C}_p$	High $\boldsymbol{C}_p$	Mod $\boldsymbol{C}_p$	Low $\boldsymbol{C}_p$	any $\boldsymbol{C}_p$
1	0.299	0.198	0.209	0.158	0.079	0.147	**0.009**	**0.004**
2	0.380	0.359	0.125	0.114	0.699	**0.000**	**0.000**	**0.000**
3	0.382	0.277	0.257	0.106	**0.048**	**0.010**	**0.000**	**0.000**
4	0.196	0.211	0.212	0.216	0.779	0.711	0.647	0.628
5	0.394	0.388	0.213	0.164	0.918	**0.000**	**0.000**	**0.001**
6	0.198	0.172	0.143	0.065	0.485	0.139	**0.000**	**0.005**
7	0.198	0.179	0.134	0.156	0.598	0.067	0.237	0.099
8	0.243	0.221	0.189	0.161	0.478	0.053	**0.007**	**0.016**
9	0.396	0.371	0.170	0.125	0.727	**0.001**	**0.000**	**0.001**
All	0.299	0.264	0.183	0.141	0.067	**0.000**	**0.000**	**0.000**

Table 4.2: Average rotational deviation from direct trajectory, identical conditions as Table 4.1. p values < 0.05 are in bold.

Chapter 5

Shared Guidance For Adaptive Neural-Prosthetic Control

5.1 Introduction

Prosthetic systems controlled by brain-computer interfaces (BCI) have become a reality within the last decade. Using single-unit recordings from microelectrodes implanted in the motor cortex, cursors [106, 113] and then robotic arms [13] have been controlled via direct brain interface. So far, these experiments have been focused on the control of computer cursor or robotic hand translation, with an experiment in our laboratory adding the control of a robotic gripper in 2008 [121]. During the same period, advanced anthropomorphic robotic arms and hands have been under development [2, 119, 120]. While these devices are intended to be used as prosthetics, they have more controlled degrees-of-freedom (DoF) than have been shown in BCI control. In order to use the brain to operate advanced prosthetic devices that approach the complexity of the human limb, systematic methods to extend BCI beyond control of hand translation and closure to control over more complex robotic behaviors must be found.

We have recently created a BCI arm and hand control system to afford control of hand rotation in addition to the translational and hand aperture DoF that have already been demonstrated. In a series of experiments, two monkeys have learned to control the 7-DoF robotic manipulator in an oriented grasping task using an implanted cortical microelectrode interface. The emergence of control skill in the two monkeys

was the result of two interdependent processes:

(1) A cortical decoding system for robot control was constructed based on brain activity recorded as subjects observed the moving robot. This established a first approximation to the link between control intent and robot movement.

(2) The subject then learned to operate the device through practice of a control task. This was a process of the cortical output changing to realize subject intent within the decoding system established in (1). To drive this process, the difficulty of the control task was constrained to be low at first and increased in time as the subject developed skill.

Both of these processes relied on an adaptation of computer-operator shared control, in which movement of the robot was driven by a mixture of control information from the brain of the subject and synthetic control commands from a computer. Shared control systems, in which human control of a robot is modified to keep robot movements within safe boundaries or enhance task performance, originated in telerobotic and robotic surgery applications [53, 61, 95]. For neuroprosthetic systems, shared control has been used in an offline study for stabilizing BCI output in [54], and in our laboratory for online reduction of error in cursor and robot hand transport control [121]. Somewhat different interpretations of shared control have been envisioned for neuroprosthetic interfaces in which a subject and agent learn to cooperatively complete a motor task using reinforcement learning [98], for actuating coordinated hand postures in [19], and in operation of a wheelchair with a limited-bandwidth EEG interface [82].

In this chapter we describe a novel "bimodal" shared control (BSC) algorithm that was able to drive the processes of observation-based model calibration and control assistance for training in the 7-DoF robot experiment. The first "active" BSC mode was similar to one previously used offline for stabilization of recorded BCI control signals [54]. In our experiment, this type of shared control was used to provide a movement substrate for observation-based cortical decoder calibration, to provide a template for patterning task-related behavior in the subject, and to restrict the degrees of freedom over which the subject was required to control the device during early learning. The second "passive" BSC shared-control mode was applied to selectively reduce control error after the subject had acquired some skill in device operation and required training in its operation under a consistent task-reward framework. Integration of the two shared control modes in the BSC system was integral to successful

achievement of full brain control over the 7-DoF manipulator.

5.2 Recruitment of Neural Activity for Brain-Computer Interfaces

The connection between the activity of individual neurons in the motor cortex and actuation of the upper limb at individual muscles or joints is complex. Areas in the motor cortex that have been used for BCIs contain neurons that innervate distributed sets of muscles [65], and single neurons often simultaneously encode multiple representations of motor movement across joints [86]. Many of these encodings correspond to coordinated movements of multiple joints or muscle groups, rather than simple muscle contraction or even actions at a single joint [46]. At the same time, representations of individual hand movements are distributed among populations of neurons [5, 91]. A motor control model that has captured the neural encoding of a type of coordinated motion is the preferred-direction (PD) model for linear Cartesian hand movements [32]. This model describes how hand movement direction information is encoded in the activity of populations of neurons, which has led to recruitment of these populations to drive the movement of cursors and robots using BCIs [13, 106, 113].

Specific population activity occurring during natural movements and BCI control is a spontaneous and unconscious process resulting from conscious movement intention. While it has been shown that individual neurons in isolation can be modulated using processes such as operant conditioning [26, 27], whether or not these units can be intentionally modulated in isolation without coordinated activation of other neurons is unknown. Either way, conditioning the differential activation of individual neurons is not a natural way of driving physiologic or robotic movements.

Prosthetic control interfaces such as peripheral nerve transfer [60] and manual interfaces (e.g. foot control) use the deliberate control of individual body movements or muscles to effect desired movements of an external device. These types of interfaces differ from BCI in that movement at individual joints and activation of some individual muscles can be consciously individually controlled. The process of learning to control these interfaces is simplified by the ability of the prosthetic user to systematically explore the effect of individual inputs on the output of the system.

The clinical population to whom BCIs may be most valuable, however, are those

who cannot produce movement or peripheral neural activity on their own, such as those with spinal cord injury or neurological diseases such as ALS. These are also the patients for whom the highest number of controlled prosthetic DoF will be the most useful. For these people to use sophisticated neuroprosthetic devices under BCI, there is no equivalent to a prosthetic control interface based on selective voluntary activation of individual inputs to the system. Despite popular belief, BCI systems do not simply read the mind of an operator to decode intended motor movements from neural activity. Instead, a useful BCI control system must be adapted to the way in which the subject is able to activate coordinated groups of neurons.

While patterns of neural activation occurring during natural movements can be adapted for control [13, 113], this will not be available for clinical BCIs. Imagining the movement of the prosthetic limb may be a useful foundation for building decoding models in human subjects [40], but may be incomplete if the mechanics of the effector robot are different from the human limb or motor control command signals are diminished in the brain due to disuse.

Another method that has used to identify movement-related patterns of cortical activity is to record neural activity during the observation of movement in a device. Using a monkey that is trained to manually move a robot, it has been shown that task-related neural activity is produced when the monkey observes the robot moving autonomously [115, 116]. Observation of a moving robot has been used for building the cortical decoder used in our laboratory's 4-DoF BCI robot control task [121]. While these experiments involved a preliminary hand movement task that formed a model connecting voluntary movements of the monkey to corresponding movement in the device, we can now show that observation of a moving robot alone can be used to elicit task-modulated activity. Neural activity from observation may reflect the kind of visual identification with the movements of other actors as hypothesized in the study of the "mirror system" of the brain [30, 94]. As task-modulated activity was recorded in the majority of motor cortical neurons sampled during observation periods, these results indicate a more fundamental relationship between execution and observation of movements possibly based on mental simulation [23, 90]. Models built on observation-based activity allowed the monkeys to successfully operate the 7-DoF robot.

After a cortical decoder is built, the subject must learn to use it to control the robot. When people approach a set of unlabeled controls for a machine, they deliberately explore the controls, observing the effect of manipulating each input on the

output of the machine. Because of the abstract and unconscious nature of the activity of individual cortical neurons, there is no correlate to direct exploration of the inputs in BCI systems of significant complexity. Control based on recording neural activity is also subject to both the stochastic nature of neural systems as well as apparent drift and noise resulting from unrelated processes taking place simultaneously. It is becoming clear that a very important part of establishing quality BCI control is using closed-loop systems in which the user is given a chance to practice using the device. While practicing, the user learns the mapping from the activity of a large number of spiking neurons to some set of controlled robotic DoF. This can be described as the neurons learning the inverse of the decoder plant [38].

5.3 Bimodal Shared Control Algorithm

The remainder of this chapter describes the "Bimodal" Shared Control (BSC) algorithm that was used to drive the process of building a cortical decoder based on observation-based activity, then allowing the subject to practice use of the BCI device while controlling task difficulty. Each part of the BSC algorithm will be discussed, followed by how they were integrated into a single control law.

5.3.1 Active Shared Control

Active shared control directly mixes subject control commands with control originating from an artificial source. Weights assigned to each control DoF control the mixing ratio between the two sources. Artificial control information can come from any source; in the offline control study of Kim et al [54], artificial control originated with a set of sensors mounted on a robotic arm to stabilize and enhance task-related control properties of the robot. In our experiment, artificial control originated with an autonomous software agent using a model of the running BCI task to continuously compute commands that would help to complete the task. This input was used in the shared-control scheme to provide the movements of the robot required for observation-based decoder calibration, as a template to pattern task-related behavior in the subject, and to isolate the robotic DoF over which brain control information was required for task completion as the monkey learned.

The active shared control algorithm simply mixes a column vector of subject

control commands $\boldsymbol{u}$ with a vector of automatic control commands $\boldsymbol{d}_1$ to produce the mixed control command vector $\boldsymbol{v}$:

$$\boldsymbol{v} = \boldsymbol{C}_a\boldsymbol{u} + (\boldsymbol{I} - \boldsymbol{C}_a)\,\boldsymbol{d}_1 \tag{5.1}$$

An $n \times n$ diagonal matrix of active shared control parameters $\boldsymbol{C}_a$, where n are the number of controlled robotic DoF, is used to set the portion of control coming from the user in each DoF. Each entry c_a in $\boldsymbol{C}_a$ is in the range $0 \leq c_a \leq 1$. By adjusting the individual c_a, some control dimensions can function more autonomously, while others can be dominated by control generated by the subject.

5.3.2 Passive Shared Control

The "passive" shared control mechanism is the Flexible Fixturing algorithm described in Chapter 4. Control command directions embedded into the desired command matrix $\boldsymbol{D}$ represent directions in the control space of the robot that would move it towards task completion or towards more desired robot configurations. The $\boldsymbol{D}$ matrix was generated in our experiment by the autonomous controller, where the multiple columns in $\boldsymbol{D}$ reflected movement toward different ways of approaching a cylindrical target object for grasping. Columns in the $\boldsymbol{D}$ matrix could also be formed by sensors attached to the prosthetic robot. While not shown in our experiments, the passive shared control system is potentially very useful for sensor-based shared control applications because it passively attenuates control error rather than driving the robot directly.

5.3.3 Integrated Active and Passive Shared Control

The complete Bimodal Shared Control algorithm combines both passive and active shared control methods into a single control law. While the use of both algorithms at once may at first appear to add a large amount of complexity to the system, the two types of shared control have different purposes in BCI experimentation and are not generally active over the same sets of DoF at the same time. While the active shared control system is more useful when the subject has little or no control skill in control of a robotic DoF, the passive method is employed to allow the subject to gain skill in particular DoF by attenuating task error as they practice using them. Integration

of both types of shared control into the same control law reflects how operation of the different DoF can be learned in incremental stages. Training sequences that take advantage of both shared control modes are discussed in Section 5.1

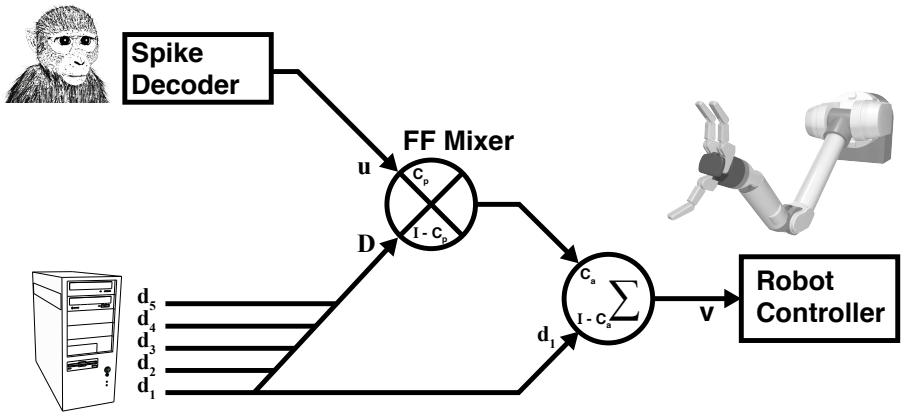


Figure 5.1: Schematic for the bimodal shared control system showing two layers of integration between BCI and automatically generated robot commands.

The Bimodal Shared Control Algorithm

The BSC algorithm takes as input the command vector $\boldsymbol{u}$, the mixing parameters of both the active and passive shared control mechanisms $\boldsymbol{C}_a$ and $\boldsymbol{C}_p$ respectively, and the autonomous control input $\boldsymbol{D}$ and $\boldsymbol{d}_1$. $\boldsymbol{d}_1$ can be one of the columns in $\boldsymbol{D}$, in our experiment taken to be a vector pointing towards the center of a region of object grasp robot poses. While the scale of the columns in $\boldsymbol{D}$ do not matter, the vector $\boldsymbol{d}_1$ must be scaled appropriately to act as a direct command to the robotic system. Integration of the active and passive algorithms is sequential such that the user input is masked or attenuated by the passive algorithm, then mixed directly using the active control stream, as illustrated in Figure 5.1.

The BSC system computes the output command vector $\boldsymbol{v}$ from the subject control input $\boldsymbol{u}$, attenuation target matrix $\boldsymbol{D}$, and optimal control vector $\boldsymbol{d}_1$ as

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{C}_a \left(\mathrm{S}(\boldsymbol{D}, \boldsymbol{u}) + \boldsymbol{C}_p \mathrm{K}\left(\boldsymbol{D}, \boldsymbol{u}\right)\right) \boldsymbol{u} + (\boldsymbol{I} - \boldsymbol{C}_a)\boldsymbol{d}_1 \\ \mathrm{S}(\boldsymbol{D}, \boldsymbol{u}) &= \widehat{\boldsymbol{D}}(\widehat{\boldsymbol{D}}^T \widehat{\boldsymbol{D}})^{+} \widehat{\boldsymbol{D}}^T \\ \widehat{\boldsymbol{D}} &= \mathrm{MaxPosSpanBasis}\left(\boldsymbol{D}, \boldsymbol{u}\right) \\ \mathrm{K}(\boldsymbol{D}, \boldsymbol{u}) &= \boldsymbol{I} - \mathrm{S}\left(\boldsymbol{D}, \boldsymbol{u}\right) \end{aligned} \tag{5.2}$$

where

$\boldsymbol{u}$ is an $n \times 1$ vector in the input DoF.

$\boldsymbol{C}_p$ is an $n \times n$ diagonal matrix of passive shared control parameters in the range $0 \ldots 1$

$\boldsymbol{C}_a$ is an $n \times n$ diagonal matrix of active shared control parameters in the range $0 \ldots 1$

$\boldsymbol{v}$ is an $n \times 1$ vector of output commands.

$\boldsymbol{D}$ is an $n \times p$ matrix of p desired command vectors.

$\boldsymbol{d}_1$ is an $n \times 1$ desired command vector. This can be any vector from $\boldsymbol{D}$, but it may be preferable for this to be chosen for optimality.

S is the positive span projection matrix of $\boldsymbol{u}$ on $\boldsymbol{D}$.

K returns the kernel matrix complementary to S.

MaxPosSpanBasis is a function returning a set of columns in $\boldsymbol{D}$ that maximally positively span the input $\boldsymbol{u}$ (see algorithm 2 in Chapter 4).

5.3.4 7-DoF Brain-Computer Interface Experiment

The 7-DoF BCI robot control experiment mentioned previously relied heavily on the BSC algorithm in all phases of cortical decoder building and subject training. We discuss how the BSC algorithm was used to develop the ability to control the 7-DoF device here. The overall experiment is discussed in greater detail in Chapters 7 and 8.

Two monkeys had chronic microelectrode arrays implanted in their motor cortices. Recordings of neural activity at these electrodes were processed to form control commands for a prosthetic robot. This robot, a WAM robot with attached Barrett hand (Barrett Technologies, Cambridge, Mass., USA) was mounted next to monkey subjects and across from a rigid target presentation robot. The WAM robot was controlled by the monkey in the 6-DoF space of linear and rotational velocities of the robot hand and 1-DoF grip aperture control of the fingers. A cylindrical target object was mounted at the end-effector of the presentation robot. Targets in each trial were set to be approximately 30 cm away from and 40 degrees of rotation from the starting location and orientation of the WAM hand. The experimental task setup is shown in figure 5.2 with the robot successfully grasping the oriented target.

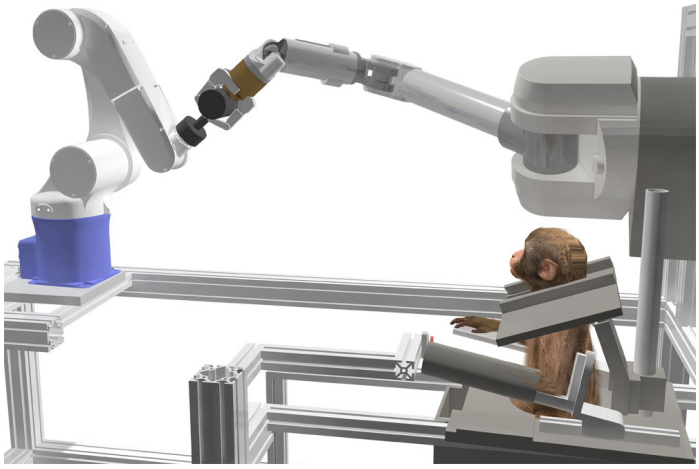


Figure 5.2: The experimental setup used for 7-DoF BCI control. The prosthetic robot is shown grasping the target object, completing an experimental trial.

Daily experimental sessions consisted of upwards of 100 experimental trial reaching tasks. Control commands to the robot were supplied by a software module running the BSC system continuously in 30 ms intervals throughout the experiment. The output command vector $\boldsymbol{v}$ from the BSC module directly commanded the movement of the robot.

When the task began, the autonomous control module supplied a matrix $\boldsymbol{D}$ whose columns pointed towards a sphere approximating each of the extents of the linear and rotational space in which the robot could grasp the target cylinder. If the cylinder was successfully grasped within 12 to 14 seconds, the trial was a success and the monkey received a liquid reward. If this time elapsed without a successful grasp, or if the experimenter detected that the monkey had stopped attending to the task, the trial was manually aborted. Attendance to the task was signified by the monkey remaining grossly stationary and watching the moving robot arm. Each daily experimental session consisted of an adaptation period in which a cortical decoding model was trained using observation-based activity elicited in the robot, followed by a training session in which the monkey learned to use the trained decoder to complete robot tasks. The corresponding settings of $\boldsymbol{C}_a$ and $\boldsymbol{C}_p$ used during the session are discussed in the next two sections.

5.4 Shared Control Cortical Decoder Calibration

The active shared control mechanism provided a simple method to perform observation-based calibration. During this process, all $c_a = 0$ so that the subject simply watched the robot performing the task autonomously. An example autonomous execution of the robot task is shown in Figure 5.3. Sampled neural activity and robot velocities were collected from 18-40 trials and used to calibrate the cortical decoder as discussed in Chapter 3.

After observation-based calibration, a "coadaptive" period could be used in which the decoder was repeatedly calibrated while mixing increasing amounts of subject control signals with autonomously generated ones (a similar process was performed in [121] and [113]). This enabled the robot to continue to successfully perform the task as in the observation-only segment, but for the trajectories to be influenced to some degree by commands originating from the BCI. The motivation for this scheme was that subsequent calibrations were thought to better reflect the control intent of the monkey, and it could also help to promote understanding of the influence of intent on the robot trajectories. At the same time, this produced a sort of moving target for the monkey to learn the decoding system. Coadaptation was used with one of the two monkeys, but both were able to successfully control the device at the end of training; the utility of coadaptation remains unclear.

5.5 Shared Control BCI Training

BCI training using the BSC algorithm allowed a consistent motor task to be performed by a prosthetic robot while the underlying difficulty of the brain-control task was manipulated. When the monkeys first began to control the robot, active shared control was applied such that the task was completed almost autonomously. The function of the shared control mechanism at this stage in training was to establish the operant conditioning paradigm of associating task completion with reward. A small amount of monkey control was added to promote understanding that the robot was under brain control, but once the task pattern had been taught to the monkeys the passive shared control mechanism was used preferentially. The passive shared control coefficient matrix entries c_p were at first set to low levels, then increased gradually as the subject gained control skill. The subjects trained sequentially by first operating

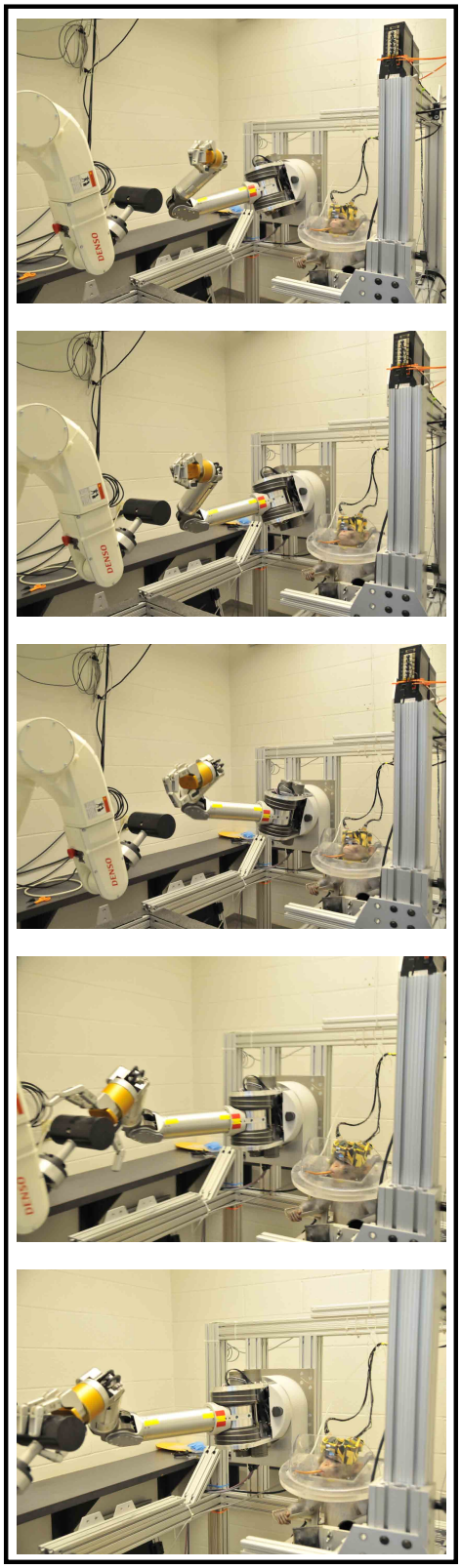


Figure 5.3: A monkey observing a model reach sequence during decoder calibration.

the linear DoF with gripper actuation, then the rotational DoF and grip, and finally all of the controlled DoF together. The active shared control mechanism allowed degrees of freedom not being trained to be automatically controlled. During training in the complete set of controlled DoF, the linear and rotational shared control parameters (c_p) were often set at different levels for different degrees of freedom, reflecting differing skill levels in the control of each movement type. The goal was to match the difficulty of the task to the skill level of the monkey; the skill needed to complete the task was to be just high enough for the monkey that learning would be promoted, but not so high that the monkey would become discouraged by task failure.

5.5.1 Task state-dependent shared control application

Each trial in the experiment was organized as series of "task states" that organized the temporal sequence of events that comprised a complete reaching and grasping trial. The shared control parameter elements c_a and c_p were set depending on the task state of the system. During reward delivery and intertrial periods when the monkey was not expected to control the robot, $c_a = 0$ so that the robot was controlled automatically. This provided a convenient mechanism to reset and standardize the robot position at the beginning of experimental trials, without imposition of a separate control system to handle procedural rather than task-related control.

The oriented grasping task was divided into five task states (Figure 5.4): Reaching, HandOpening, Orienting, PreGrasp, and Grasp. The action of the robot in each state was as follows:

HandOpening The hand opened without otherwise moving.

Reaching The closed hand moved from the starting position to the center of the workspace in a forward orientation.

Orienting The hand oriented toward the target in a fixed position.

PreGrasp In the target orientation, the hand approximated the target cylinder for grasping.

Grasp The hand closed on the target.

During the reaching and grasping portion of each trial, c_a and c_p were set to different levels depending on the monkey's skill in each task state movement. Task states in the sequence could also be skipped completely, so that only the specific

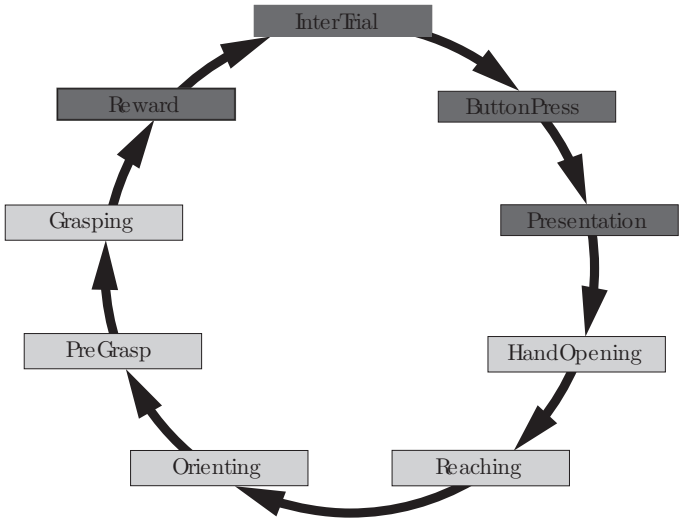


Figure 5.4: The sequence of task states that comprise a trial. Procedural portions of the trial in which the robot did not move are more darkly shaded.

movements on which training was desired would need to be performed to complete the task. By manipulating which task states are skipped and the c_p level for the individual DoF during each task state, a progression of training regimes could be imposed on the subject. The specific sequence that was conducted in our experiment is as follows:

Table 5.1: Shared-Control Training Progression

	HandOpening	**Reaching**	**Rotation**	**PreGrasp**	**Grasp**
Phase 1 - Linear/Grip (simplified task)					
Linear	$c_a = 0$	c_a, c_p max	X	X	$c_a = 0$
Rotation	$c_a = 0$	$c_a = 0$	X	X	$c_a = 0$
Grip	c_p max	$c_a = 0$	X	X	c_p max
Phase 2 - Rotation/Grip					
Linear	$c_a = 0$	X	$c_a = 0$	X	$c_a = 0$
Rotation	$c_a = 0$	X	c_p max	X	$c_a = 0$
Grip	c_p max	X	$c_a = 0$	X	c_p max
Phase 3 - Sequential Linear/Rotation/Grip					
Linear	$c_a = 0$	c_p max	$c_a = 0$	X	$c_a = 0$
Rotation	$c_a = 0$	$c_a = 0$	c_p max	X	$c_a = 0$
Grip	c_p max	$c_a = 0$	$c_a = 0$	X	c_p max
Phase 4 - Crossover Linear/Rotation/Grip					
Linear	c_p max	$c_p = 1$	c_p max	X	c_p max
Rotation	c_p max	c_p max	$c_p = 1$	X	c_p max
Grip	$c_p = 1$	c_p max	c_p max	X	$c_p = 1$
Phase 5 - Simultaneous Linear/Rotation/Grip					
Linear	X	X	X	c_p max	c_p max
Rotation	X	X	X	c_p max	c_p max
Grip	X	X	X	c_p max	c_p max

where

c_p **max** was the maximization of corresponding parameters to the tolerance of the monkeys (see below).

$c_a = 0$ was the automatic control of corresponding parameters.

X was skipping the state completely.

The progression through task phases was essentially a method for behavioral shaping [83] that gradually increased the demands of a BCI motor task until full control of the robot was performed. The monkeys first learned a simplified reaching task **Phase 1** that was equivalent to previous BCI robot control experiments. The simplified task was a center-out task that involved no hand rotation but included hand closure. Entries $c_a < 1$ were used at the same time as c_p at first during **Phase 1** to drive association of task completion with reward and set up the operant conditioning paradigm. When that was completed, the monkeys completed a rotation task **Phase 2** that was like linear task but in the space of rotations. By skipping the reaching

state, the hand went directly to a central position at the beginning of trials from which it could perform pure rotation to align with each target. When rotation was complete, the hand automatically approximated the target and the monkey controlled closure of the gripper. Next, **Phase 3** was begun, which was the complete task that required linear, rotational, and grip motion to complete. To introduce this more difficult task to the monkey, we first had it perform a sequential version of the two linear and rotational tasks that it had already learned. In **Phase 4**, opposing movement types slowly became subject controlled during the linear and rotational sequential task phases, so that the monkey learned simultaneous control of the complete set of 7 control DoF simultaneously. When control was headed towards complete admittance of monkey commands, **Phase 5** was begun, in which the goal from the start of the trial was to complete the overall task.

5.5.2 Models for Programmed Learning

Within each training phase, a theoretical model for updating c_p max is indicated in Figure 5.5. At first, the c_p entries are set at a low level. The inexperienced monkey would start at a low success rate even with a large amount of assistance. As it practiced, the success rate would increase. When it reached a high level, entries c_p would be increased to make the task more difficult, and as a result the success rate would return to a lower level. This process would repeat until the monkey fully controlled the device. The BSC algorithm functioned in this way like an adjustable set of training wheels to keep accumulating control error from causing the robot to enter poses from which successful task completion is very difficult. The use of BSC to keep difficulty within a manageable range is in accordance with theories of learning such as the "Challenge Point Hypothesis" [33] and the process of skill acquisition being promoted during "Flow" states [21]. This also has correlates in calibrating the difficulty level to skill level in robot-assisted rehabilitation [58, 59, 62], in studies involving auditory discrimination training [92], and previous brain-computer interface experiments performed in our laboratory using "deviation gain" [121].

In the actual experiment, the need to sustain monkey motivation made the process of c_p value calibration more complicated. A set of heuristics were developed that generally followed the theoretical model, but in which special attention to motivation was given at certain critical points by the experimenters. These points were usually when high subjective challenge levels were sustained for longer periods

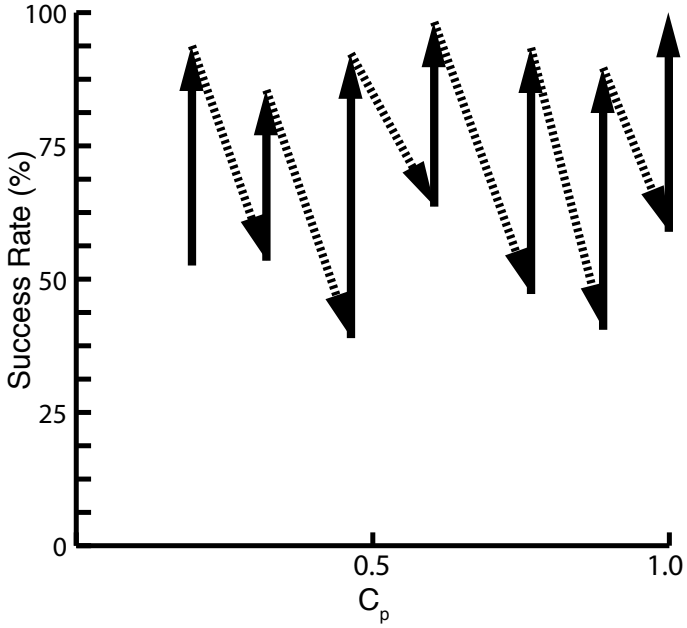


Figure 5.5: Theoretical model of progress to full control of a device by adjusting the assistance level.

of time; the difficulty was then dialed back, or supplemental rewards were given to the monkeys to encourage continued attempts to control the device. Often, monkey attention was cyclical and the programmed difficulty level was manipulated back and forth to attempt to match the motivation level of the monkey at a particular moment in time.

From the beginning of the many weeks of training that one monkey received using this system, we occasionally attempted to give it full control of the prosthetic device immediately after decoder calibration. Uniformly, the monkey had difficulty at the beginning of the training period in performing the task; even in simplified (3-DoF linear movement) trials, the robot would be controlled poorly at first. This would lead to task failure and witholding of reward. Quickly, within one or two failed trials, the monkey would lose focus and stop participating in the task. This was evidenced by the monkey squirming in the chair, turning away from the robot, and generally acting in an unsettled way that was incompatible with brain control. To maintain attention and participation, we had to consistently make it possible for reward delivery to be achieved by increasing the assistance level. During the latter portion of the experiment when the monkey was highly trained, it may have been possible in some sessions to avoid the use of the BSC algorithm for error reduction, as we were able to decrease the assistance level very quickly. However, it was impossible

to know in advance on which days this would be possible. Causing the monkey to enter the non-participatory state made it more difficult to control its behavior over the entire session, so it was not done. We did not attempt to put the second monkey into immediate full control of the device. That monkey displayed very high sensitivity to task failure and it had become clear with the first monkey that the programmed use of shared control was integral to promotion of the process of learning.

5.6 Results

Observation trials occurred at the beginning of each experimental session in which the monkeys viewed the robot executing trajectories like those in Figure 5.3. After 18-40 trials were observed, spike rates and robot velocities recorded during observation were used for decoding model calibration as described in chapter 3.

During the subject training phase of the experiment, the monkeys now used brain control to acquire the target in the linear, rotational, or all six movement DoF in addition to the grip DoF if the hand was available. As discussed, shared control was used to attenuate control error such that the control task was possible for the monkey to complete. Examples of the passive mechanism of shared-control assistance in human subject joystick control are in Chapter 4. The task dynamics are the same as those in the joystick experiments, with the target set to a region of radius 8-12 cm and 12.5-25 degrees from perfect alignment.

Results from a simulation of how linear assistance altered a brain-controlled trajectory are in Figure 5.6. In this figure, a dynamic model of the WAM robot was used to simulate robot movement resulting from recorded BCI and autonomous control commands under different levels of shared control. The actual trajectory recorded during the experiment, resulting from linear $c_p \approx 0.4$ is shown with simulated trajectories at linear $c_p = 0$, $c_p = 0.5$, and $c_p = 1.0$. The linear target region is the indicated sphere, approximating a region of valid grasps of the target cylinder. While the trajectory corresponding to full control moved well outside the target region, the simulated $c_p = 0.5$ moved the robot to the border of the target and the fully constrained trajectory went near the target center. Note that even with $c_p = 0$, lateral movement still directed between the target boundaries was not attenuated.

During training sessions the process for adjusting c_p parameters that was used

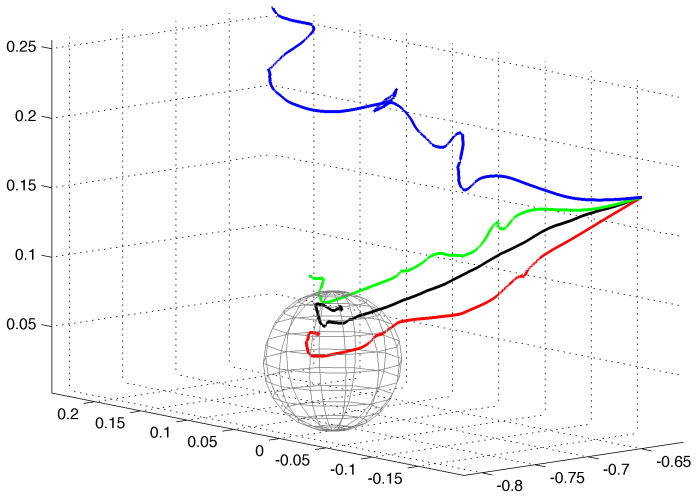


Figure 5.6: Results from simulation of a single control trial at different linear assistance c_p levels. Recorded brain-control commands were input into the BSC algorithm that output control commands at different levels of assistance. This output controlled the trajectory of a dynamic simulation of the prosthetic robot. Trials at $c_p = 1$ (blue), $c_p = 0.5$ (green) and $c_p = 0$ (red) are shown with the original recorded trajectory ($c_p \approx 0.4$, black). The linear control target region is denoted by the sphere.

with the monkeys followed the programmed pattern indicated in Section 5.5.2 reasonably well. As described earlier, however, heuristic methods for maintaining monkey motivation were also part of the process for developing skilled control.

The effects of using shared-control assistance for training during three different BCI sessions are shown in Figures 5.7-5.12. Each successive pair of figures indicates monkey performance over a single BCI session. Each of these sessions incorporated some successful 7-DoF control trials, but the pathway towards full control was somewhat different in each. The first panel of each pair shows the relationship between shared control assistance and success rate throughout the session. Shared control parameters c_a and c_p are shown as $1 - c_a$ and $1 - c_p$ to indicate the amount of assistance given to the monkey. The beginning of each session started with high active shared control levels, corresponding to the observation-only segment. After decoder calibration near trial 40-60, the monkey was given control of the device by quickly reducing active assistance. Passive parameters were decreased more slowly, in response to the performance and behavior of the monkey. In the first figure from the first session (Figure 5.7), the assistance level was decreased monotonically yet the success

rate was maintained around 50% until the end of the session. While the success rate gradually decreased, success rates upwards of 50% indicate that general proficiency in using the device has been achieved, and the success rate can be primarily attributed to the level of motivation of the monkey. As the session continued, the monkey became more fatigued and more satiated with water. Near the end of the trial, the success rate went down rapidly as the monkey's behavior indicated that it no longer wanted to participate in the session.

The second panel for each session shows the same success and shared-control assistance rates, but now a rudimentary measure of skill is added to show how the monkey's control changes through the course of training. This measure is simply the trial average dot product between the monkey's command vector and the normalized autonomous controller command vector $\boldsymbol{d}_1$ throughout control periods of trials. The same measure applied to the composite control command after shared control application is also shown in each secondary figure. In the second panel for the first session (Figure 5.8), these lines merge as expected as $(1 - c_p)$ parameters near 0. What is interesting in this graph is that (1) the apparent skill level is high at the beginning of training when the shared control assistance level is high, (2) this skill level decreases as the challenge level is increased, and (3) the skill level increases again at low levels of assistance before full control is given. The skill level then corresponds to some extent with the success rate for the remainder of the session. The drop in skill rate corresponding to the monkey becoming unwilling to continue performing the BCI task can also be seen near the end of the session.

Moving to the first panel of the second session (Figure 5.9), the trial began much like Figure 5.7, but the rate of decrease of $(1 - c_p)$ parameters was slowed between trials 50 and 70 to maintain the success rate while the monkey practiced using the decoder. While the success rate before the slowdown in increasing difficulty did not drop dramatically, the monkey's behavior during this period indicated that it was finding these trials difficult. Certain behaviors such as agitation can be observed when the monkey is in this state, and challenge levels were not decreased until the monkey started to find the trials easier. Over the course of the session, the success rate gradually decreased as the monkey became satiated and fatigued, but once again was maintained at a level near 50% until the end of the session. The skill measurement data shown in Figure 5.10 have a similar relationship to the progression through the session as that in the first session (Figure 5.8).

The same set of figures for a third session are shown in Figures 5.11-5.12. The skill

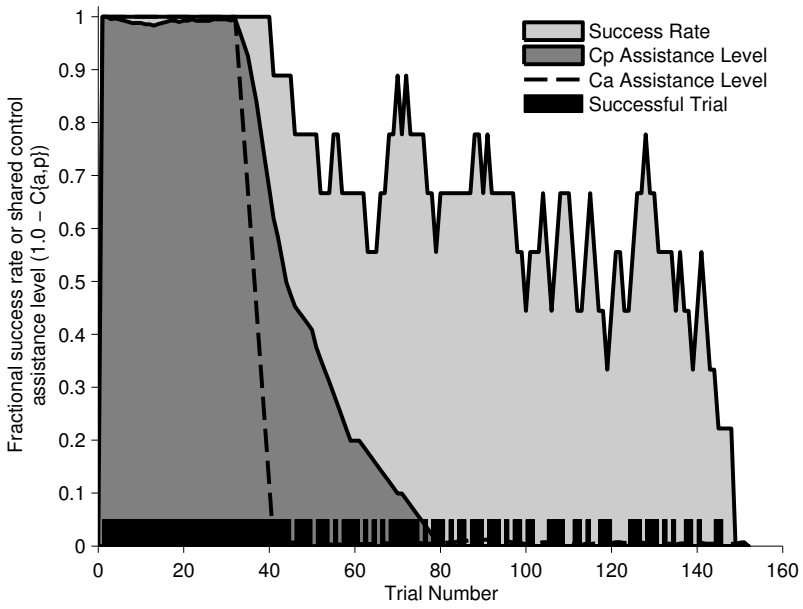


Figure 5.7: Shared-control parameters vs. success rate, first session.

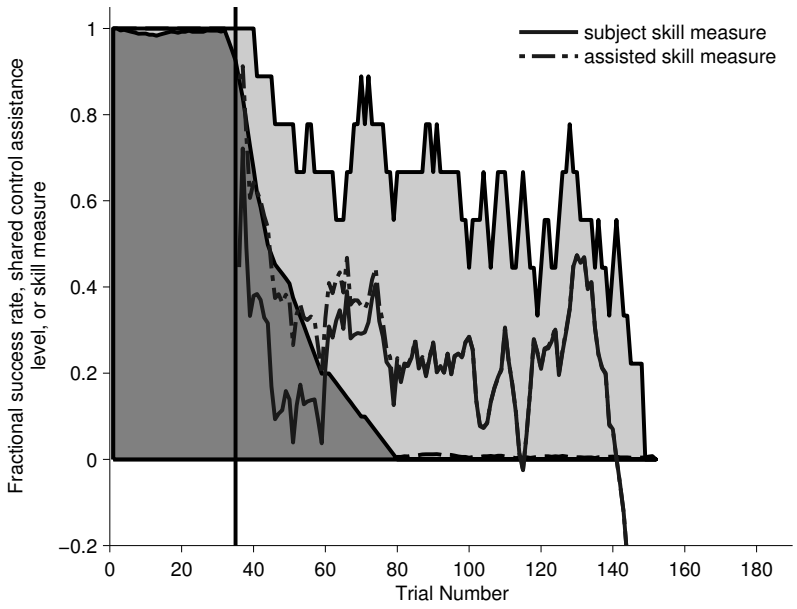


Figure 5.8: Shared-control parameters vs. success rate and skill measure, first session.

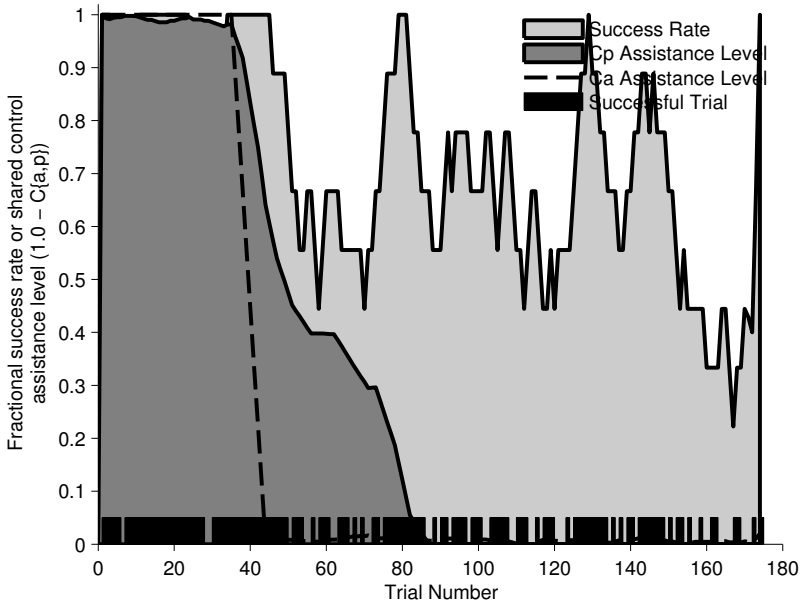


Figure 5.9: Shared-control parameters vs. success rate, second session.

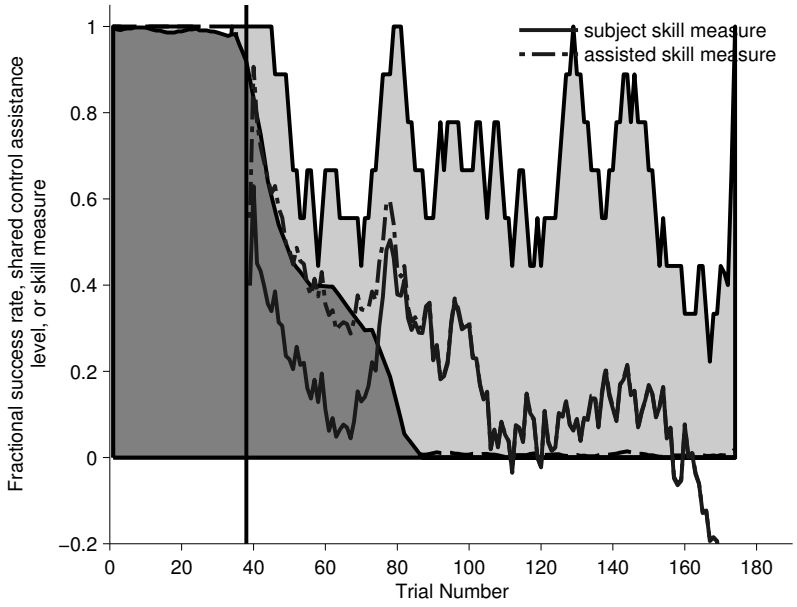


Figure 5.10: Shared-control parameters vs. success rate and skill measure, second session.

measures are slightly different for subject and composite commands after $1 - c_p = 0$ due to a synchronization problem in the recording; the actual difference between the two streams was negligible. In this session, assistance decrease was attempted too rapidly; note the two instances of decreasing success level and corresponding return to higher assistance. After additional trials, the $(1-c_p)$ levels could be decreased such that good performance under full brain control could be achieved. Note the decrease in performance in this session near trial 150, then a return to higher performance within 10 trials. This is an instance of wavering attention and motivation of the monkey. Instead of decreasing the challenge level, other methods such as extra reward delivery were used to coax the monkey into executing the task.

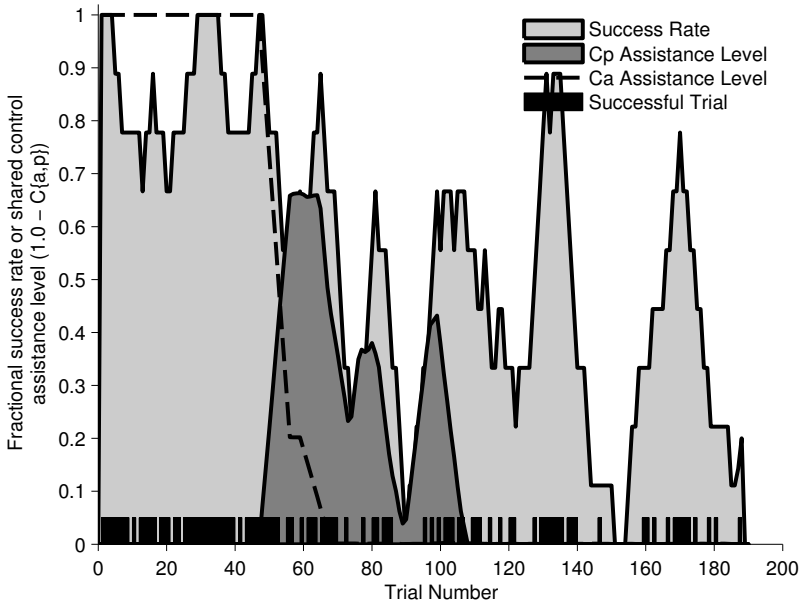


Figure 5.11: Shared-control parameters vs. success rate, third session.

The variability of the skill measure seen at high success rates during the middle and latter part of each session is interesting; this corresponds to the monkey completing the task in less direct ways. The control vector may not point directly at the target for much of the trial, but the summed effect over time is toward the target in all trials where the average skill measure is above 0. The converse may be the reason why the apparent skill is high early in the training period with a large amount of help; at this level, the robot tends to stay on a path directly from the starting position to the target. These short and direct trials may make it easier for the monkey to compose a consistent direct command signal towards the target.

The relationship between shared control assistance level, success rate, and deviation from direct movement to targets was next examined over a large number of 6-D and 7-D control sessions. In Figure 5.13, every trial from 127 sessions is plotted as

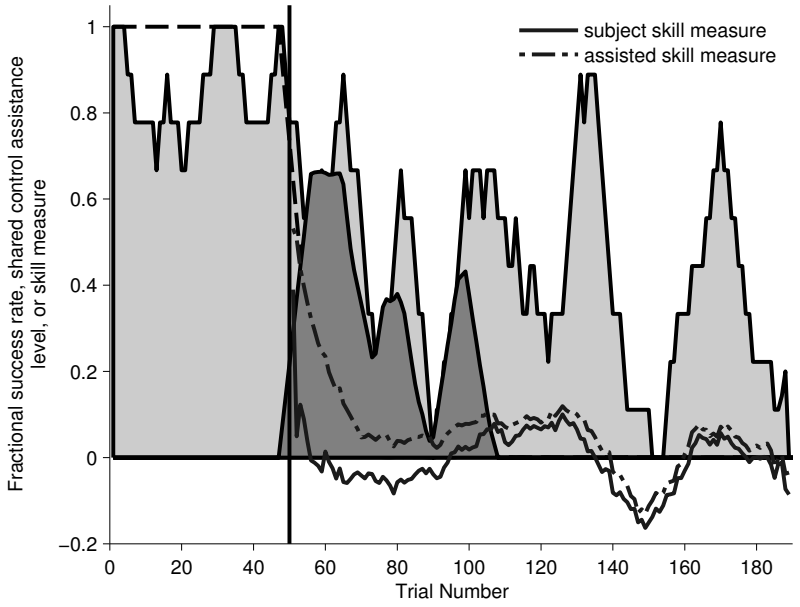


Figure 5.12: Shared-control parameters vs. success rate and skill measure, third session.

a mean trajectory deviation at its mean shared-control assistance level. Trajectory deviation was measured by sampling the vector between the robot starting position and target to 100 evenly-spaced points, then taking the distance to the closest point on the path from the robot at each time sample during the movement period of a trial. The shared-control assistance level on the x-axis is $c = (1.0 - \bar{c}_p) + (1.0 - \bar{c}_a)$, an indicator of the mean overall help given in the trial. Average success rates are also shown at the different assistance levels. Both success rates and trajectory deviation are averaged in bins centered on 0.1 intervals of c parameters and plotted, with ± 1 standard deviation of the trajectory deviation also shown. Data is shown for a total of 19343 trials. In this figure, the success rate lowered looking from left to right as the level of assistance decreased. Mean deviation from straight trajectories also generally increased at lower levels of assistance, but this effect leveled off as assistance reached the lower levels. This means that the robot inhabited regions of the workspace generally at the same range from the target with no assistance and low assistance, but the slight assistance level was associated with a greater amount of trials fully completed.

In Figure 5.14, a subset of 29 sessions (3590 trials) in which notably high-quality 6 or 7-D control was performed are extracted from Figure 5.13 and plotted alone. In this figure, success rates were higher and average deviations were correspondingly lower compared to the full set of 127 sessions. A different pattern was present, however, in which success rates and deviations did not follow the trend of improving with increased assistance near the 0.2 c level. Increased performance at lower shared-control assistance levels indicates increased skill development in training that occurred

over the course of these sessions. Poorer performance overall may have occurred with c parameters $= 0$ because after the session proceeded to full control the c_p parameter level was rarely increased in response to decreased performance, and performance tended to decrease over the course of long sessions as the monkey became fatigued. When $c = 0.2$, however, the monkey is always engaged in the control task on the way to establishing full brain control.

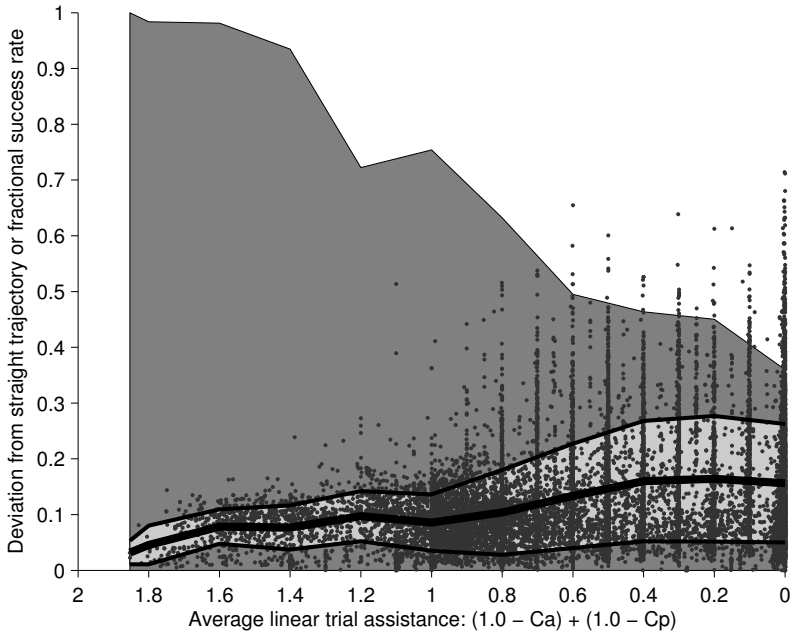


Figure 5.13: Relationship between shared control assistance level, trial success rate, and linear deviation from a direct path to target, from over 19000 trials in 127 sessions. Success rate is graphed in the darkly shaded region. Average deviations for each trial are plotted as points, with the mean ± 1 std of these points form the lightly shaded region.

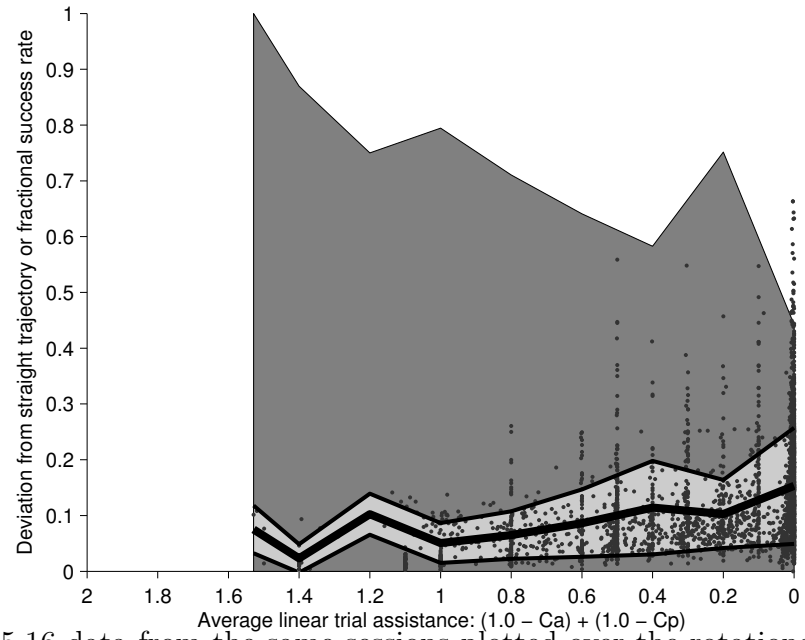


Figure 5.14: Relationship between shared control assistance level, trial success rate, and linear deviation from a direct path to target, from 3590 trials in 29 sessions in which the monkey achieved full brain control of the device. The quantities plotted are the same as those in Figure 5.13.

In figures 5.15 and 5.16 data from the same sessions plotted over the rotational DoF. Rotational deviation was measured by finding the shortest rotational path from

starting orientation to target using quaternion interpolation (Slerp algorithm) sampled 100 times from starting position to target. The distance to the closest of those points to the robot orientation by angular distance at each time sample was used as the rotational deviation. A similar pattern is seen in the rotational DoF as the linear ones. The significant dip in success rate and increase in deviation around $\boldsymbol{C} = 1$ in the record of all sessions is possibly due to early training in composite linear and rotational commands that included very small amounts of rotational control. While rotation was very well represented in the neural adaptation data (Chapter 3), it was at first difficult for the monkeys to perform rotational control in addition to linear control; a number of sessions with a large amount of rotational assistance were required to promote learning. As with the linear DoF, success rate and trajectory deviation were not linked directly to the amount of assistance given, indicating a successful process of BCI training using shared control. While the success rate overall decreased as the amount of assistance decreased to 0, this level is still near around 50%, especially in the sessions that were noted to contain especially high-quality control. These overall success levels were maintained sometimes for up to about 100 trials at this level of difficulty. Given the inherent difficulty of controlling a 7-DoF manipulator in real time, maintenance of even a 50% success rate in the task represents a demonstration of a large amount of skilled high-dimensional brain control.

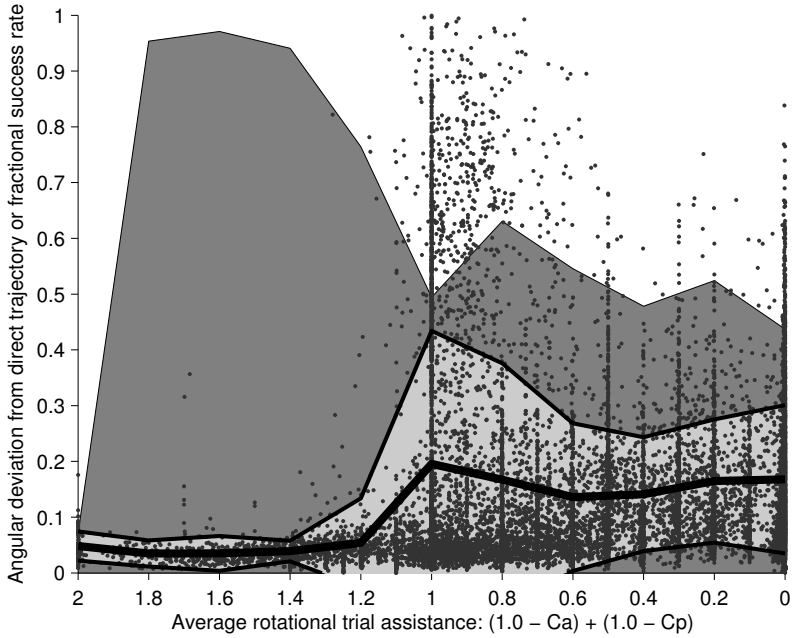


Figure 5.15: Relationship between shared control assistance level, trial success rate, and rotational deviation from a direct path to target, from over 19000 trials in 127 sessions. Rotational equivalents to the quantities plotted in Figure 5.13 are shown.

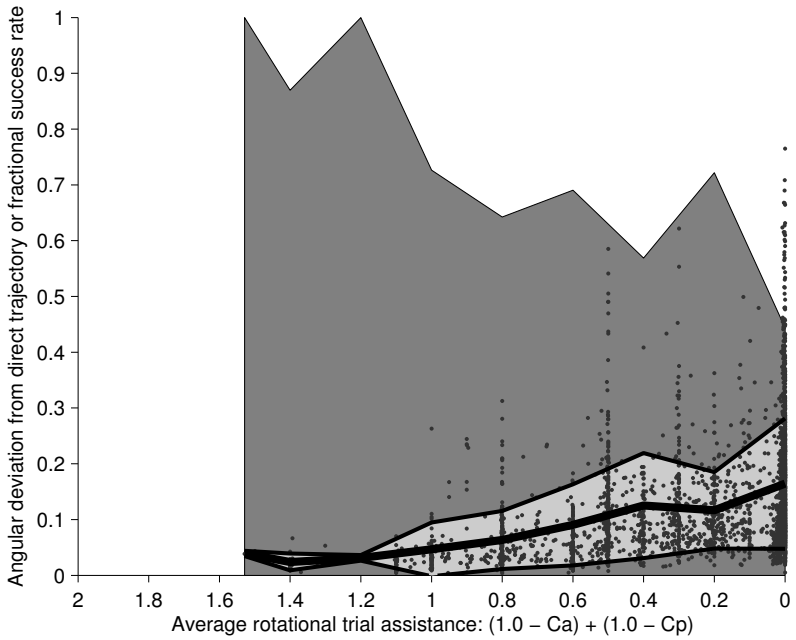


Figure 5.16: Relationship between shared control assistance level, trial success rate, and rotational deviation from a direct path to target, from 3590 trials in 29 sessions in which the monkey achieved full brain control of the device. The quantities plotted are the same as those in Figure 5.15.

Chapter 6

Velocity and Impedance Control of a Prosthetic Robot

6.1 Introduction

Compliant motion control is an important topic in the design of robots that interact with an unpredictable world. Industrial robots that are designed to move rigidly and in very precise ways can be unsuitable or dangerous outside of controlled environments. Beyond safety concerns alone, the design of robots with which both movement and force can be controlled simultaneously can enhance their utility for many applications, including interaction with humans. Haptic or human-interactive robots must operate at least with the ability to safely limit interaction forces, but direct control of the relationship between robot kinematics and forces will often be desired. In an ideal robot, interactions between the robot and a human or the environment (intentional or not) will elicit a controlled disturbance response while it maintains effort towards following a commanded motion [4].

The application that drove the development of the control system described here is a brain-computer interface controlled prosthetic arm robot. Our work is an extension of the work of Velliste et al [121], in which monkeys were able to directly interact with a brain-controlled arm. Interactions among robot, user, and environment designed to simulate the operation of the primate limb call for a robot controller that allows for similar compliant and flexible interaction. In this work, the decoded command signal is typically in the domain of end-effector linear and rotational velocity,

but the target for reaching and grasping is a solid object that the robot often collides and interacts with, especially when the monkey subjects are unskilled in operating the robotic prosthetic device. In this chapter, we present an algorithm to allow force and kinematic control to be integrated smoothly into a control system so that monkeys could learn to operate the brain-controlled robot in the presence of obstacles.

In one of his classic papers, Hogan [41] describes the interaction between a compliant robot and its environment as that of an impedance interacting with an admittance or vice versa. A robot control method for impedance is described in which force interactions between a robot and its environment are directly specified as a second-order system with six degrees of freedom. This controller causes a robot to behave like a physical system with a given mass, damping and stiffness. An ideal impedance controller of this type would completely mask the dynamics of the robot and replace them with this desired set of force responses to environmental perturbations. This means that at the point of interaction with the robot, its response to perturbation is indistinguishable from the physical system specified in the controller.

Impedance control has been implemented using an acceleration-resolved approach for motion control [108], in which linearized robot dynamics are computed to obtain the Cartesian acceleration behavior of the robot given a certain set of impedance control parameters in an absence of interaction. These are then cast into applied torques at the robot actuators given the relationship between static forces and torques given by the Jacobian [109]. A position-based impedance control scheme that involved explicit dynamic masking was given by Kang et al [48]. Another approach to Cartesian impedance control has been given in a large amount of highly successful work with the DLR robot[4, 77].

Here we present a new way to approach the application of impedance control to joint torque-controlled kinematic chain manipulators. Using this method, the dynamics of a robot are first explicitly masked or compensated for so that in theory it provides no resistance to environmental admittances. This means that the controller attempts to regulate the exact joint torques needed to maintain the kinematic state in the presence of gravity and torques due to the dynamics of the robot. This is possible using an actual robot to the degree that a complete kinematic and inertial model of it can be ascertained, including accurate measurement of its joint velocities and accelerations. Next, a link to the desired impedance behavior of the robot is given using the concept of "Virtual Links" in a Newton-Euler dynamic model of the robot. Virtual Links allow desired Cartesian forces and torques to be directly set at

any point on the robot. Taking advantage of Hogan's principle of superposition of impedances [41], an arbitrary number of simultaneous impedance-type controls can be implemented at the same time. Joint-level impedances or desired torque behaviors can be similarly superimposed.

We implemented a type of impedance control of at a robot endpoint in which the kinematic behavior of the robot could also be directly controlled. This was done using a 7-DoF Barrett whole-arm manipulator (WAM) robot (Barrett Technology, Cambridge, MA). The WAM robot is a completely backdriveable, cable-driven robot designed for haptic interaction. This controller was used in a brain-computer interface prosthetic control experiment in hundreds of hours of continuous operation in which monkeys often controlled the robot very poorly. Brain-derived kinematic commands oscillated rapidly and were completely unpredictable, so that no trajectory planning could be used and collisions with objects in the environment were frequent. This controller proved to be very robust and useful in this experiment, responding fluidly to chaotic input and allowing the control of force production between the robot and external objects.

6.2 Masking Robot Dynamics

In this section, we describe how the robot dynamics are masked such that they can be replaced with the desired dynamic behavior. The equations of motion that relate the dynamics (in the case of the WAM, a vector of joint torques, $\boldsymbol{\tau}$) of a robot to its kinematics (the joint angles, $\boldsymbol{q}$, and their first and second derivatives), adapted from Spong [109] is:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}}) \tag{6.1}$$

where $\boldsymbol{M}$ is the joint-space inertia tensor, $\boldsymbol{C}$ are inertial coupling terms, including centrifugal and Coriolis torques, $\boldsymbol{F}$ are friction terms, and $\boldsymbol{G}$ are the pose-dependent torques on the robot links due to gravity. This is a general form of the inverse dynamics problem that can be solved in multiple ways using Lagrangian [110], Kane's equations [47], or Newton-Euler based approaches such as one presented in the next section.

6.2.1 Recursive Newton-Euler Dynamics

The recursive Netwon-Euler (RNE) equations efficiently compute the inverse manipulator dynamics (Equation 6.1) of a kinematic chain. Orin [76] originally used the Newton-Euler equations of rigid body motion applied to links of a robot, while Luh et al [68] formulated the recursive RNE approach used here. The clear description of RNE and figure here is based on that of of Corke [20]. The RNE method is a two-step process:

1. In a first outward recursive step, the forces and moments at the center of mass (COM) of each link based on propagating robot kinematics (movements of links based on joint angles, velocities, and accelerations) are calculated.
2. In a second inward recursive step, the forces and torques at each link COM are propagated backwards through the kinematic chain from the endpoint of the manipulator to find the total forces and torques acting at the location of each joint. For rotational joints, the portion of the torque lying in the joint axis is the actuator torque for that kinematic state of the chain.

This presentation of the algorithm requires a kinematic and inertial model (mass, COM, moment of inertia) for the robot in reference frames following the standard Denavit-Hartenberg (D-H) notation [34]. The RNE equations here are for a robotic manipulator consisting of $n+1$ links $[0, 1, \ldots, n]$, where link 0 is the base of the manipulator, and n revolute joints indexed by $[1, 2, \ldots, n]$. Links and corresponding reference frames are indexed by the variable i. The relationship between links and joints in the standard D-H notation, as well as many of the variables in the RNE equation, is shown in Figure 6.1. The left superscript of variables in the equations below denote the reference frame for the variable. For instance, ${}^{i+1}\boldsymbol{R}_i$ is the rotation of link/frame i with respect to link/frame $i+1$. The equations shown here are for kinematic chains with rotational links only, see [20] for prismatic equivalents.

Recursive Newton-Euler Dynamic Equations

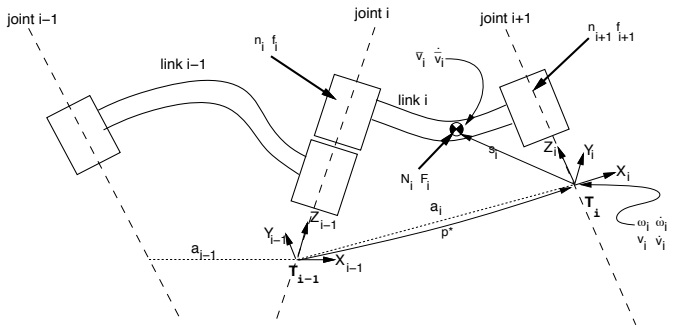


Figure 6.1: Diagram of link notations used in the RNE equations. From Corke [20].

1. Outward Recursion, $0 \leq i \leq n$.

$$
\begin{aligned}
{}^{i+1}\boldsymbol{\omega}_{i+1} &= {}^{i+1}\boldsymbol{R}_i({}^{i}\boldsymbol{\omega}_i + \boldsymbol{z}_0\dot{q}_{i+1}) \\
{}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} &= {}^{i+1}\boldsymbol{R}_i\{{}^{i}\dot{\boldsymbol{\omega}}_i + \boldsymbol{z}_0\ddot{q}_{i+1} + {}^{i}\boldsymbol{\omega}_i \times (\boldsymbol{z}_0\dot{q}_{i+1})\} \\
{}^{i+1}\boldsymbol{v}_{i+1} &= {}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{i+1}\boldsymbol{p}^*_{i+1} + {}^{i+1}\boldsymbol{R}_i\,{}^{i}\boldsymbol{v}_i \\
{}^{i+1}\dot{\boldsymbol{v}}_{i+1} &= {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} \times {}^{i+1}\boldsymbol{p}^*_{i+1} + {}^{i+1}\boldsymbol{\omega}_{i+1} \times \{{}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{i+1}\boldsymbol{p}^*_{t+1}\} + {}^{i+1}\boldsymbol{R}_i\,{}^{i}\dot{\boldsymbol{v}}_i
\end{aligned}
$$

$$
\begin{aligned}
{}^{i}\boldsymbol{F}_i &= m_i \cdot ({}^{i}\dot{\boldsymbol{\omega}}_i \times \boldsymbol{s}_i + {}^{i}\boldsymbol{\omega}_i \times \{{}^{i}\omega_i \times \boldsymbol{s}_i\} + {}^{i}\dot{\boldsymbol{v}}_i) \\
{}^{i}\boldsymbol{N}_i &= \boldsymbol{J}_i\,{}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i}\boldsymbol{\omega}_i \times (\boldsymbol{J}_i\,{}^{i}\boldsymbol{\omega}_i)
\end{aligned}
$$

2. Inward Recursion, $n \geq i \geq 1$.

$$
{}^{i}\boldsymbol{f}_i = {}^{i}\boldsymbol{R}_{i+1}\,{}^{i+1}\boldsymbol{f}_{i+1} + {}^{i}\boldsymbol{F}_i \tag{6.2}
$$

$${}^{i}\boldsymbol{n}_i = {}^{i}\boldsymbol{R}_{i+1}\{{}^{i+1}\boldsymbol{n}_{i+1} + ({}^{i+1}\boldsymbol{R}_i\,{}^{i}\boldsymbol{p}^*_i) \times {}^{i+1}\boldsymbol{f}_{i+1}\} + ({}^{i}\boldsymbol{p}^*_i + \boldsymbol{s}_i) \times {}^{i}\boldsymbol{F}_i + {}^{i}\boldsymbol{N}_i \tag{6.3}$$

$$
\boldsymbol{\tau}_i = ({}^{i}\boldsymbol{n}_i)^T({}^{i}\boldsymbol{R}_{i+1}\boldsymbol{z}_0) \tag{6.4}
$$

where

$\boldsymbol{z}_0$	is the vector [0 0 1]
$\boldsymbol{J}_i$	is the moment of inertia of link i around its center of mass (COM)
$\boldsymbol{s}_i$	is the position of the COM of link i wrt frame i
$\boldsymbol{\omega}_i$	is the angular velocity of link i
$\dot{\boldsymbol{\omega}}_i$	is the angular acceleration of link i
$\boldsymbol{v}_i$	is the linear velocity of link i
$\dot{\boldsymbol{v}}_i$	is the linear acceleration of link i

$\boldsymbol{N}_i$	is the moment at the COM of link i due to motion of the link
${}^{i-1}\mathbf{R}_i$	is the rotation matrix defining the orientation of frame i wrt the frame attached to link $i-1$
${}^{i}\boldsymbol{p}_i^*$	is the displacement of frame i wrt frame $i-1$
$\boldsymbol{F}_i$	is the force at the COM of link i due to motion of the link
$\boldsymbol{n}_i$	is the summed moment exerted at joint i by links $i \dots n$ and outer links acting on it
${}^{i}\boldsymbol{f}_i$	is the summed force exerted at joint i by links $i \dots n$
$\boldsymbol{\tau}$	is the torque at joint i

If ${}^{i}\dot{\mathbf{v}}_i$ is set to the acceleration due to gravity in the base frame, the commanded torques are modeled to be exactly the same as the actual torques at each joint for a motorless robot in free fall [68] . If the torques are reversed, the free fall is nullified so that gravity compensation is performed. The dominant torques in the dynamic equation (6.1) come from gravity for relatively low-friction and slow-moving robots.

In computed-torque control schemes, the kinematic parameters $q, \dot{q}, \ddot{q}$ are set to the desired behavior of the robot. The RNE equation then solves for the joint torques necessary to produce these desired kinematics [25]. However, the purpose of this step in our formulation is to attempt to compensate for the dynamics of the physical robot, so that instead we solve for the joint torques necessary to maintain the current kinematic state. Using only the *measured* kinematics of the robotic manipulator to compute the joint torques to actuate the robot, the actual dynamics of the robot are masked by active compensation. Thus, if the endpoint (or any point) of the robot is moved by an external force, torques will theoretically be generated such that no resistance is felt by the external actor. Without the compensation running, pushing on the robot would be resisted both by gravity as well as the inertia, friction, and coupling forces of the links of the robot. With the compensation algorithm running, a current is commanded in the motors to counteract the force which would be felt externally, so that no resistance is felt. This is of course limited by the bandwidth of the running model; a resistance to external force will be felt until the running model can read the new kinematic state of the robot and compensate for it. Masking of the dynamics is also of course highly dependent on the accuracy of the model used; to completely mask the dynamics of the robot all of the terms of the dynamic model would need to be known exactly and the kinematic state accurately measured.

In practice with the WAM, compensating the gravity term alone is sufficient for

many purposes. The second largest term in terms of torque magnitude in the dynamic equations are the frictional forces, while the coupling and mass terms produce even smaller torques in the WAM and similarly lightweight robots. Compensation for these terms is more difficult to implement for real-time compensation because the velocity and acceleration parameters are difficult to estimate over different timescales. While differences between discrete joint angle measurements are noisy, filtering these measurements makes the estimated velocity inaccurate for rapidly changing movements. A full dynamic masking model has been implemented in simulation, and masking of the frictional forces has been performed in real-time control. Because of the velocity measurement problem, it is difficult to account for friction in both the case of large movements over long time scales as well as when, for instance, the robot endpoint is grasped and shaken rapidly. Thus, the gravity compensation alone has generally been used in validation experiments.

6.3 Virtual Links

The RNE equations have a boundary condition in the inward recursion step. This boundary condition exists at the final link, $i = n$, when ${}^{n}\mathbf{f}_{n}$ and ${}^{n}\mathbf{n}_{n}$ are calculated. These parameters for the final link describe the "load" or external force (force in the remainder of the chapter refers to forces and torques) on the robot imparted by the environment [25]. While this force has been set to some real imparted load for computed-torque controllers using RNE, we consider the effect of entering the force due to a fictional load here. During the backwards recursion step, the RNE algorithm will compute the joint torques needed to compensate for this fictional force. The result is that the robot will produce an equal and opposite force at the end effector. Similarly, fictional forces at any link can be entered into the backwards RNE equation. Each of these fictional forces can be represented as a virtual link acting on some real link of the robot. Their effect is calculated in the inwards step as if they were the forces imparted by a physical link, so that their effect is summed over the center of mass of each inward link and the torque calculated at each joint to compensate for it. The end effect is a superposition of torques that compensate for the dynamics of the real links and torques that produce desired forces at the virtual link interaction points. While the link calculations for each frame are performed with forces in the frame of each link, the transform between the base and link frames have been calculated with the forward kinematics of the robot such that imparting forces in the base frame is

easily done. Thus, Equations 6.2-6.3 above become:

$$
\begin{aligned}
{}^{i}\boldsymbol{f}_i &= {}^{i}\boldsymbol{R}_{i+1}{}^{i+1}\boldsymbol{f}_{i+1} + {}^{i}\boldsymbol{F}_i + {}^{i}\widehat{\boldsymbol{F}}_i \\
{}^{i}\boldsymbol{n}_i &= {}^{i}\boldsymbol{R}_{i+1}\{{}^{i+1}\boldsymbol{n}_{i+1} + ({}^{i+1}\boldsymbol{R}_i{}^{i}\boldsymbol{p}_i^*) \times {}^{i+1}\boldsymbol{f}_{i+1}\} + ({}^{i}\boldsymbol{p}_i^* + \boldsymbol{s}_i) \times ({}^{i}\boldsymbol{F}_i + {}^{i}\widehat{\boldsymbol{F}}_i) + {}^{i}\boldsymbol{N}_i + {}^{i}\widehat{\boldsymbol{N}}_i
\end{aligned}
$$

where $\widehat{\boldsymbol{F}}$ and $\widehat{\boldsymbol{N}}$ are the forces and torques due to the summed action of any virtual links attached to link i. These are with reference to frame i, such that base-frame forces and torques must be converted to frame i coordinates using the forward-kinematic transform.

Desired joint-level behavior can also be superposed into the model as well by directly adding them to the output $\boldsymbol{\tau}_i$:

$$\boldsymbol{\tau}_i = ({}^{i}\boldsymbol{n}_i)^T({}^{i}\boldsymbol{R}_{i+1}\boldsymbol{z}_0)$$

becomes

$$\boldsymbol{\tau}_i = ({}^{i}\boldsymbol{n}_i)^T({}^{i}\boldsymbol{R}_{i+1}\boldsymbol{z}_0) + \widehat{\boldsymbol{\tau}}$$

where $\widehat{\boldsymbol{\tau}}$ is a goal torque at joint i. This works because it is equivalent to the action of a virtual link that produces only a torque directly at the joint at its axis. As long as these desired torques are added to the sum total of torques at each joint, rather than replacing them, the desired Cartesian and joint behavior will take place simultaneously. This is simply an elaboration of the examples of superposition of impedances given in [42]. The recursive form of RNE is convenient in that it provides direct access to potential interactions with each link within a single consistent framework.

While the kinematics of the robot are not directly controlled in this step, the control of forces and torques at any point or joint on the robot provides the link to control needed to implement desired force, impedance, or kinematic behaviors, with gravity and the dynamics of the robot masked. Any control algorithm that would be used with the joint-torque controlled robot directly can now be applied to the RNE inward recursion, which will have the same effect except that gravity and

physical robot dynamics do not have to be accounted for. For instance, in a computed-torque kinematic control scheme, or in PID control of joint angular positions (two common algorithms), these can be superimposed on the joint torques in the RNE; coefficients controlling the tightness of the control loop can be loosened significantly as the controller does not have to overcome gravity in one direction. A particular choice of replacement dynamics that affords a combination of impedance and kinematic control for a robot is given in the next section.

6.4 Impedance-Velocity Controller

We implemented a method of performing integrated Cartesian velocity and impedance control. The behavior of the robot using this control method was like a PID velocity controller under changing kinematic inputs, but its steady command state behavior matched that of a classic impedance controller. Hogan [42] introduced the impedance equation describing the interface force $\boldsymbol{F_{int}}$ exerted by an end-effector in dynamic interaction with the environment:

$$\boldsymbol{F_{int}} = \boldsymbol{K}[\boldsymbol{X_d} - \boldsymbol{X_c}] + \boldsymbol{B}[\boldsymbol{V_d} - \boldsymbol{V_c}] - \boldsymbol{M}\frac{d\boldsymbol{V_c}}{dt} \tag{6.5}$$

where $\boldsymbol{K}$, $\boldsymbol{B}$, and $\boldsymbol{M}$ are matrices describing the force/displacement (stiffness) relationship, the force/velocity (damping) relationship, and the inertia tensor, all in end-effector coordinates. $\boldsymbol{X}$ and $\boldsymbol{V}$ are the end-effector positions and velocities, where 3-D orientation and rotational velocity vectors are concatenated together with the linear position and velocity values. The subscripts d and c denote the desired and current positions or velocities. With the off-diagonal terms in $\boldsymbol{K}$, $\boldsymbol{B}$, and $\boldsymbol{M}$ ignored, this system can be thought of as a three sets of linear and rotational spring-mass-damper systems operating in each endpoint (Cartesian) DoF independently.

The equations of motion given in equation 6.5 can just be implemented directly as described in Hogan[41], where the torque is computed with reference to a disturbance from a kinematic setpoint $\boldsymbol{M}$, $\boldsymbol{X}$, $\boldsymbol{V}$. As Cartesian forces are directly accessible within the Virtual Link framework, this has a straightforward implementation at any point in the kinematic chain, by setting $\hat{\boldsymbol{F}} = \boldsymbol{F}_{int}$ computed for the desired point of interaction on the robot. It can also be used to control joint torques directly so that joint-based control can be superimposed.

In our prosthetic experiment, robot control commands provided by the monkey subject were in the space of Cartesian linear and rotational velocities at the endpoint. A useful integration of velocity and impedance control was created so that the robot could move under velocity-based brain control while interacting with the environment like a mass-spring-damper system. Motion control applications that are integrated with this type of impedance controller often function by dictating a position setpoint $\boldsymbol{X}_d$, which is like moving one end of the virtual spring around, with the robot following. We took a related approach to integrating kinematic and impedance control. Consider an impedance control algorithm operating in a frame with position $\boldsymbol{X}_d$ and velocity $\boldsymbol{V}_d$, where the desired movement relative to this frame is always 0 displacement and velocity. The impedance control equation becomes:

$$\boldsymbol{F_{int}} = \boldsymbol{K}[{}^d\boldsymbol{X}_c] + \boldsymbol{B}[{}^d\boldsymbol{V}_c] - \boldsymbol{M}\frac{d[{}^d\boldsymbol{V_c}]}{dt} \tag{6.6}$$

where ${}^d\boldsymbol{X}_c$ and ${}^d\boldsymbol{V}_c$ are the position and velocity of the robot endpoint with respect to the desired movement frame. Forces are produced by equation 6.6 to track the desired movement frame, at the same time producing impedance behavior equivalent to equation 6.5 if the desired movement frame does not change.

Considered as a mass-spring-damper system, this is like attaching a manipulator with mass parameters $\boldsymbol{B}$ to the moving desired movement frame with a spring of parameters $\boldsymbol{K}$ and a damper governed by $\boldsymbol{M}$. From the perspective of the damper, if the robot is stationary but $\boldsymbol{V}_d$ is set to a velocity $\boldsymbol{\nu}$, the robot is considered to be suddenly moving with velocity $-\boldsymbol{\nu}$ with respect to that frame. The damper issues a force which reduces this apparent velocity, moving the actual robot manipulator in the direction of $\boldsymbol{\nu}$ relative to the robot's base (global) frame. This sudden application of a velocity-dependent force is countered by $\boldsymbol{M}$ acting on the acceleration of the manipulator (really an apparent acceleration, due solely to the change in command velocity).

In the next time step, the desired movement frame has been translated by the desired velocity so that the base of the virtual spring drags the endpoint towards this new position. If the commands to the robot are issued in velocity and the position and acceleration terms are the integral and derivative of that velocity, this is just PID velocity control. For steady-state commanded motions, it acts like the desired impedance controller. With respect to the base frame of the robot, an equivalent

control equation is

$$\boldsymbol{F_{int}} = \boldsymbol{K}[\boldsymbol{X}_d - \boldsymbol{X}_c] + \boldsymbol{B}[\boldsymbol{V}_d - \boldsymbol{V}_c] + \boldsymbol{M}\frac{d[\boldsymbol{V}_d - \boldsymbol{V_c}]}{dt}$$

6.4.1 Implementation in the WAM Robot

An impedance-velocity controller of the type outlined in the previous section was implemented in the WAM robot. In this robot, the actual torque at each joint is directly specified so that an equivalent current is regulated at each motor by a fast PID control loop. Since the robot is backdriveable and these torques are the overall torques at each joint, joint torques do not need to be separately measured. Joint angles were also determined from readings from optical encoders at each motor. The robot controller ran in a 500Hz loop in which the joint angles were read, the impedance-velocity commands calculated, and torque commands sent to the motor controllers. The control loop period executed using a real-time system implemented in RTAI (real-time Linux) on older PC hardware. The basic RNE impedance control algorithm has been run at upwards of 1kHz loop timing with no modification.

Cartesian velocity commands were received over a network interface asynchronously at ≈ 30 ms intervals. These commands updated the desired movement commands $\boldsymbol{V}_d$ whenever they were received (between control loop executions). These commands and joint angles from the encoder readings were the changing inputs to the real-time loop. The endpoint position and orientation of the robot were calculated using the forward kinematics of the robot. Robot endpoint linear and rotational velocities were calculated as differences in endpoint position between subsequent iterations of the control loop, which was then processed using an exponential decay filter 15 ms wide with a 2% per-sample falloff rate. The filtered velocities $\boldsymbol{V}_c$ entered the following discrete-time representation of the impedance control loop evaluated at time T, with control loop period τ:

$$\begin{aligned}\boldsymbol{F_{int}} = \boldsymbol{K}\sum_{t=0}^{T}[\boldsymbol{V}_{d,t} \;&-\; \boldsymbol{V}_{c,t}]\,\tau + \boldsymbol{B}[\boldsymbol{V}_{d,T} - \boldsymbol{V}_{c,T}] + \\ \boldsymbol{M}([\boldsymbol{V}_{d,T} - \boldsymbol{V}_{c,T}] \;&-\; [\boldsymbol{V}_{d,(T-\tau)} - \boldsymbol{V}_{c,(T-\tau)}])/\tau\end{aligned}$$

In the WAM implementation, this was a straightforward PID feedback control

loop with the proportional gain in velocity and the loop output in terms of Cartesian forces and torques, which entered the RNE equations as a virtual link at the robot endpoint. $\boldsymbol{K}$ was of a limited magnitude so that the control was primarily over the velocity of the robot directly. During object interaction, this caused the command velocity to act as a proxy for forces produced into the object. The quantity of force due to the position term was capped so that forces could not ramp up very highly over time if the arm was commanded consistently to move into an object. The performance of this controller in terms of kinematic and impedance control is discussed in Section 6.5.

Additionally, a variety of other controllers were implemented simultaneously to keep the WAM within a desired workspace and to limit its output forces and torques. Additional controllers kept the joints within certain convenient limits, using virtual torsional springs near these limits superposed with the impedance control output forces and torques. Another controller provided a joint-torque type virtual link at the first joint of the robot, controlling the redundant robot DoF (elbow swing) at a reasonable angle for reaching and grasping. All of these controllers were trivial individually, but superimposed on the joint and Cartesian forces and torques providing input to the RNE inward recursion step, which was calculated to mask the robot dynamics with the composite desired dynamics at each time step. Output torques from the RNE inward loop (equation 6.4) were applied directly as current commands at the robot motors.

6.5 Validation

In this section, we report the results from experiments conducted to validate the performance of Virtual Links and the impedance-velocity controller. The following experiments were performed using the 7-DoF WAM manipulator running the gravity compensation algorithm alone for dynamic masking. WAM kinematic and dynamic behaviors are compared to theoretical models as well as a simulated WAM dynamics model in which friction was masked in addition to gravity.

The simulated WAM model was built using the Robotics Toolbox for Matlab [20]. The inverse dynamics model used to define the joint torque output from the RNE control equations was identical to the one implemented on the WAM robot, except that the other dynamic parameters (friction, Coriolis torques, etc) in Equation 6.1

could also be modeled. The forward dynamics were calculated for simulation using the method of Walker and Orin [124], which relies on inverting the inverse dynamic model to compute the resulting kinematics of torque application at the joints. While all of the dynamic parameters were used in the forward model, only the gravity and friction compensations were used in the control (inverse) model because friction is the next largest torque component and compensation for it in the real robot is the most likely to be implemented.

6.5.1 Virtual Link Experiment

In this experiment we masked the dynamics of the WAM, applied a Virtual Link at the robot endpoint that effected a static force in the "up" (+z) direction in the robot base frame, then hung weights from the end of the WAM. This setup is shown in Figure 6.2. The weights were at first held in place by hand below the attachment point of the chain to the WAM with any slack taken out. The Virtual Link force was then applied, and after a pause the weights were released in a single motion. We wished to primarily show that commanded upward forces at the endpoint would balance weights that produce an identical downward force. The set of weights and forces that were used are shown as points in Figure 6.3. Additional trials were run at virtual link forces lower than those needed to balance the weights for analysis of the dynamic behavior of the robot under force control (the two dynamic trials that are shown here are below the gravity line in Figure 6.3 at a mass of ≈ 1.25 kg). Resulting trajectories from the static force trials are shown in Figure 6.4. The balance trajectories are all plotted, but overlap at 0 deflection. Figure 6.5 shows a trial with a balanced load.

To provide insight into the effects of friction on the model, we look more closely at the non-balanced trials in Figures 6.6 and 6.7. Recorded z displacements from trials are plotted against a theoretical model of a falling point mass starting at rest and moving according to equation 6.7, along with two simulations of the WAM that did and did not perform friction compensation. While the gravity-only simulation matches the actual WAM performance fairly well, the addition of friction compensation causes it to behave like the point model. This suggests that the control model functions as expected and performance could be increased by applying friction com-

pensation. The equation governing the behavior of the theoretical mass is as follows:

$$M_e d\boldsymbol{V}/dt = \boldsymbol{F_{ext}} + \boldsymbol{F_{int}} \tag{6.7}$$

where M_e is the mass of the weight, $\boldsymbol{F}_{ext}$ is the force of gravity on the weight $(-M_e \cdot g)$, and $\boldsymbol{F}_{int}$ is the applied force by the robot.

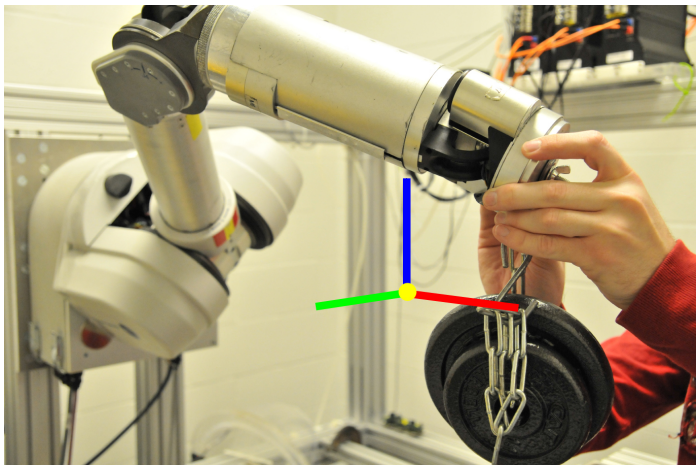


Figure 6.2: Virtual Link static force validation setup. X, y, and z axes in the robot base frame are indicated by red, green, and blue coordinate axes superimposed on photo.

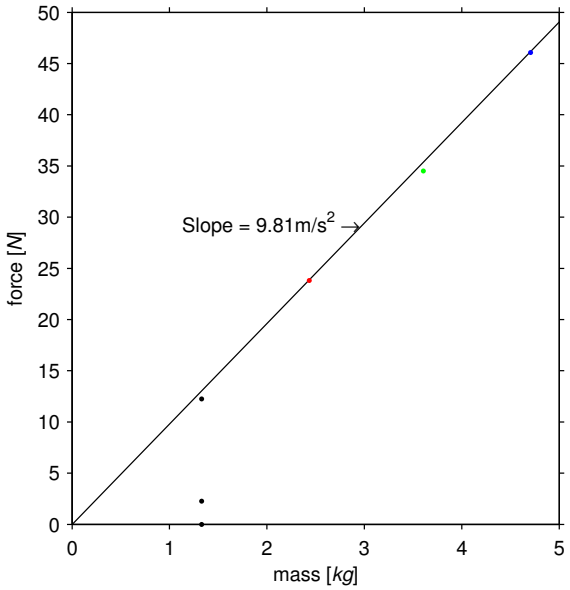


Figure 6.3: Masses hung from the WAM endpoint vs. virtual link compensation forces that were used in experimental trials.

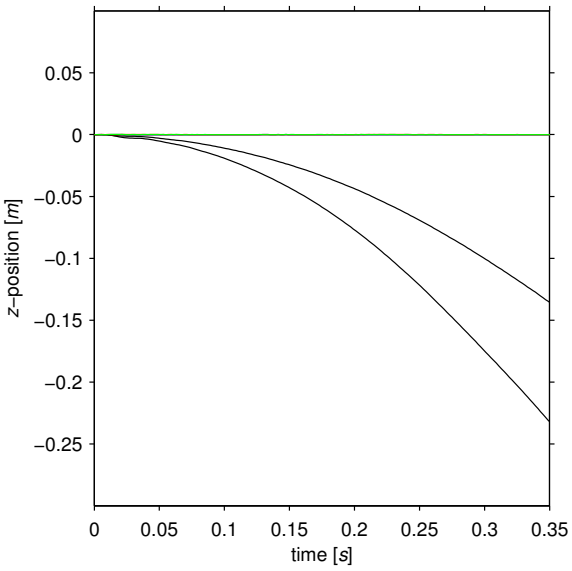


Figure 6.4: Trajectories from the virtual link experiment. The most deflected trajectory is from the trial with a $\approx$ 1.25 kg load with no applied force, and the other deflected trajectory is with the small compensatory force.

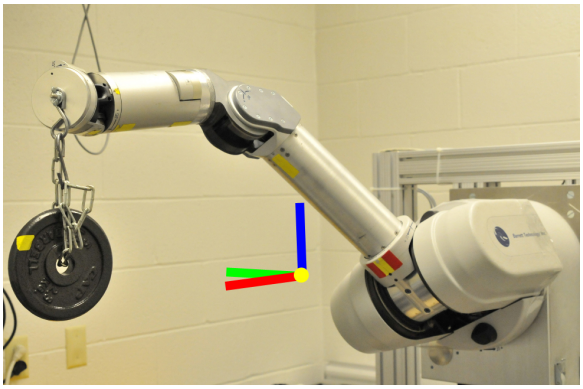


Figure 6.5: A balanced load in the virtual link force compensation experiment.

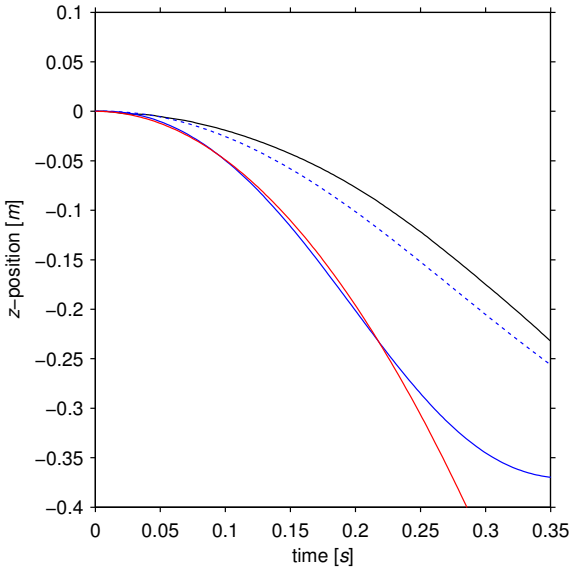


Figure 6.6: Real and simulated trajectories, dropping a $\approx$ 1.25 kg weight vs. no applied force. The solid black line is the recorded trajectory. The blue dashed line is the simulated performance of the WAM with no friction compensation. The red line is the theoretical trajectory of a falling point mass. The solid blue line is a simulation of the WAM with friction compensation. Drop times of the simulations are aligned with the point of greatest absolute acceleration in the recorded trajectory z-dimension because the exact drop time of the weight was unknown. The curve in the simulated trajectory after 0.22s is from the WAM reaching a joint limit.

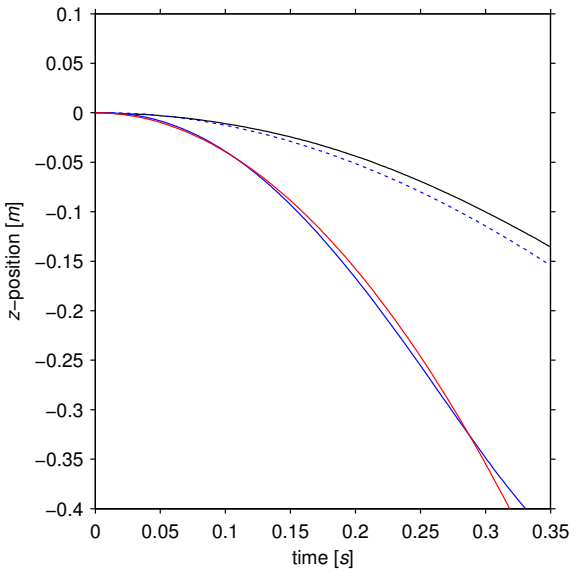


Figure 6.7: Real and simulated trajectories, dropping a $\approx$ 1.25 kg weight vs. 2.25 N compensatory force. The simulated WAM fell with a different trajectory than that of Figure 6.6.

6.5.2 Impedance Control Experiment

In this experiment, we tested the impedance response of the controller under force loads similar to the experiment in the previous section. The impedance control parameters used in this experiment are given in Table 6.2. Two sets of $\boldsymbol{K}$ values were used. While the arm was in an initial resting state with zero endpoint velocity commanded, varying masses were added to the endpoint of the arm on a chain and released, as in the virtual link experiment.

Table 6.2: Control Parameters for the Impedance Experiment

Coord.	x	y	z	ϕ_x	ϕ_y	ϕ_z
K	50, 200 $\frac{N}{m}$	50, 200 $\frac{N}{m}$	50, 200 $\frac{N}{m}$	2.5 $\frac{N \cdot m}{rad}$	2.5 $\frac{N \cdot m}{rad}$	2.5 $\frac{N \cdot m}{rad}$
B	60 $\frac{N \cdot s}{m}$	60 $\frac{N \cdot s}{m}$	60 $\frac{N \cdot s}{m}$	6.5 $\frac{N \cdot m \cdot s}{rad}$	6.5 $\frac{N \cdot m \cdot s}{rad}$	6.5 $\frac{N \cdot m \cdot s}{rad}$
M	0.14 kg	0.14 kg	0.14 kg	0.008 $kg \cdot m^2$	0.008 $kg \cdot m^2$	0.008 $kg \cdot m^2$

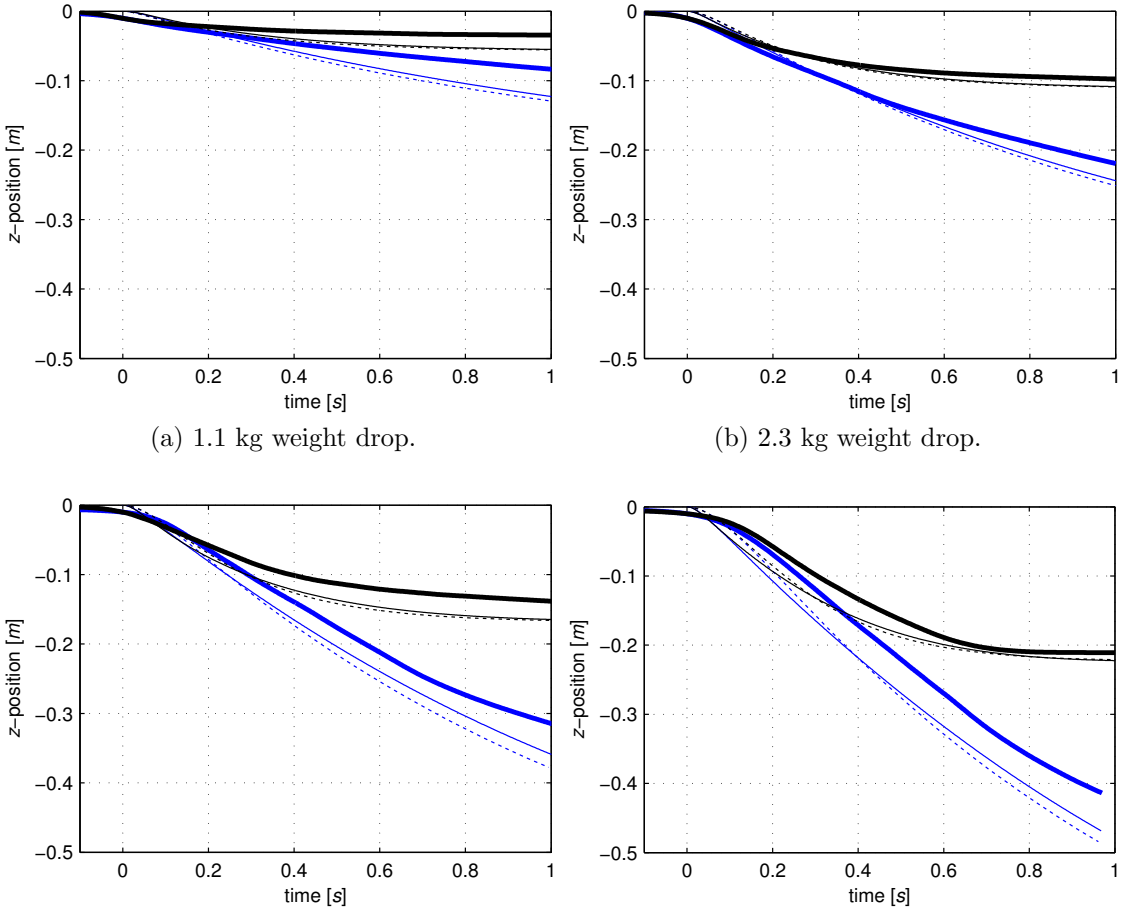
We compared the performance of the WAM robot to an idealized model of the desired impedance behavior. With an external force $\boldsymbol{F_{ext}}$ acting upon a point implementing the desired impedance behavior, we combine Equations (6.7) and (6.5) to find an overall equation of motion for simulation:

$$(M_e + M)d\boldsymbol{V}/dt = K[\boldsymbol{X_0} - \boldsymbol{X}] + B[\boldsymbol{V_0} - \boldsymbol{V}] + \boldsymbol{F_{ext}}.$$

The performance of the robot at the two $\boldsymbol{K}$ and uniform $\boldsymbol{B}$ and $\boldsymbol{M}$ settings from Table 6.2 are plotted in Figure 6.8 for comparison to each other as well as to their theoretical performance in four force perturbation trials. The performance of the WAM with only gravity compensation behaved like the theoretical controller in these trials, but the addition of friction compensation to the simulation even more closely reflected ideal performance.

6.5.3 Kinematic Performance Experiment

In this experiment, the impedance-velocity controller described in Section 6.4 was implemented at the endpoint of the WAM and its performance in responding to



(a) 1.1 kg weight drop.

(b) 2.3 kg weight drop.

(c) 3.4 kg weight drop.

(d) 4.5 kg weight drop.

Figure 6.8: $\boldsymbol{B} = 60.0, \boldsymbol{M} = 0.14$ in each trace, $\boldsymbol{K} = 200$ (black), $\boldsymbol{K} = 50$ (blue). The measured performance is the thick solid line. The theoretical performance is the dashed line. Simulated WAM performance with friction compensation is the thin solid line.

kinematic commands was evaluated. A Barrett Hand was attached to the WAM at the endpoint in this experiment and incorporated into the kinematic and inertial model of the robot. Input velocity commands to the robot were driven by a handheld controller (Sony Playstation 3 Sixaxis controller) and delivered to the RNE robot controller in 30 ms intervals. The impedance parameters were set as listed in Table 6.3. These values correspond to the diagonal terms in the matrices K, B, and M. In the first free movement trial, controller commands followed no intentional pattern, but kept the robot endpoint inside of the workspace in front of the WAM base. The resulting trajectory is plotted in Figure 6.9, with the commanded and measured velocities of the WAM endpoint in the x, y, and z dimensions following in Figure 6.10.

These results show good trajectory following performance in the 7-DoF robot.

Table 6.3: Control Parameters for the Kinematic Performance and Collision Experiments

Coord.	x	y	z	ϕ_x	ϕ_y	ϕ_z
K	550 $\frac{N}{m}$	550 $\frac{N}{m}$	550 $\frac{N}{m}$	2.5 $\frac{N \cdot m}{rad}$	2.5 $\frac{N \cdot m}{rad}$	2.5 $\frac{N \cdot m}{rad}$
B	60 $\frac{N \cdot s}{m}$	60 $\frac{N \cdot s}{m}$	60 $\frac{N \cdot s}{m}$	6.5 $\frac{N \cdot m \cdot s}{rad}$	6.5 $\frac{N \cdot m \cdot s}{rad}$	6.5 $\frac{N \cdot m \cdot s}{rad}$
M	0.14 kg	0.14 kg	0.14 kg	0.008 $kg \cdot m^2$	0.008 $kg \cdot m^2$	0.008 $kg \cdot m^2$

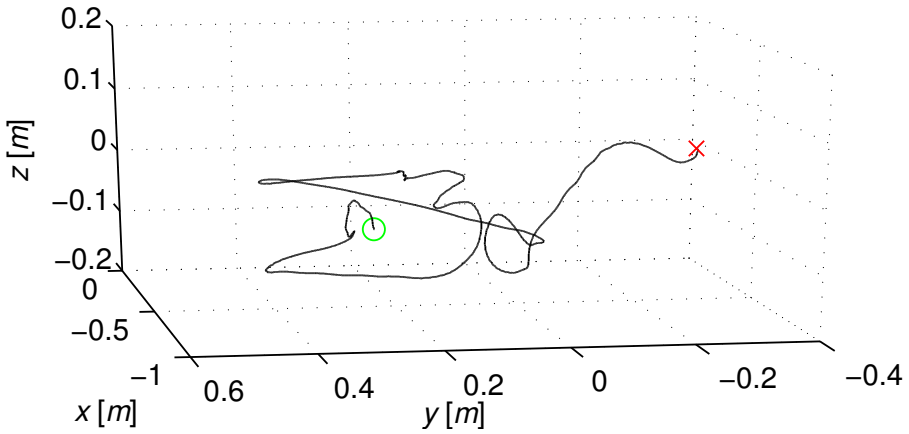


Figure 6.9: Trajectory of free movement guided by controller input, starting at the circle and ending at the X.

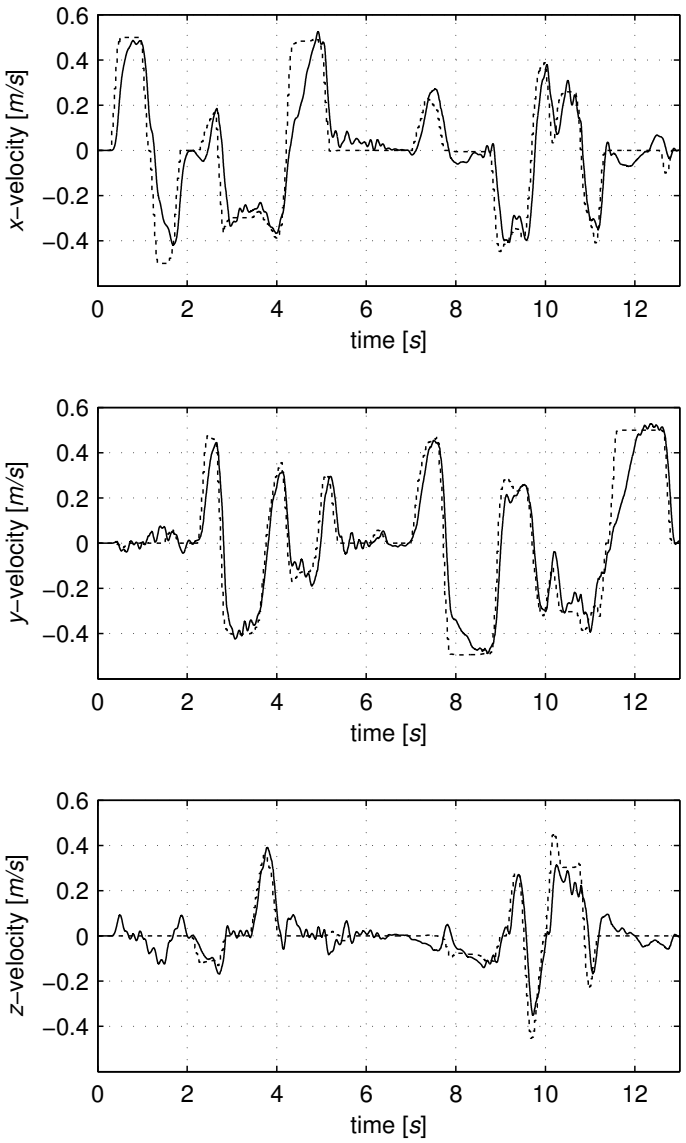


Figure 6.10: Kinematics during simultaneous control of x, y, and z velocities. The commanded and actual velocities are shown as dotted and solid lines, respectively.

6.5.4 Collision Experiment

In this experiment, a movement command from the controller was used to drive the Barrett Hand first in a short free movement in one direction, then in a similar trajectory except that the hand collided with a rigid obstacle. The parameters from the kinematic performance experiment were used in this experiment as well. Fig. 6.11 shows the hand in the starting position and in contact with the rigid object. During both trials, the commanded velocity, $\boldsymbol{V_0}$, was zero in all dimensions except for the the $+y$-direction. The velocity profile of the free movement trial is indicated in Figure 6.12. Starting from rest, the command velocity was rapidly increased to $0.3\frac{m}{s}$. This command velocity was maintained for $\sim$ 1.7s prior to dropping back to rest. The force after release does not decrease to 0 because of the value of $\boldsymbol{K}$ value used in this trial; the resulting position-dependent force was below the amount needed to move the arm due to friction.

In the second trial the same commanded input was given to the robot, but the hand hit the obstacle after $\sim 1.7s$ of movement. The kinematics and commanded force from this trial are displayed in Fig. 6.13. The force response of the robot is stable, but ramps up quickly due to the relatively rigid $\boldsymbol{B}$ and $\boldsymbol{K}$ parameters in the model. It levels off after reaching a maximum force level applied to the robot controller output.

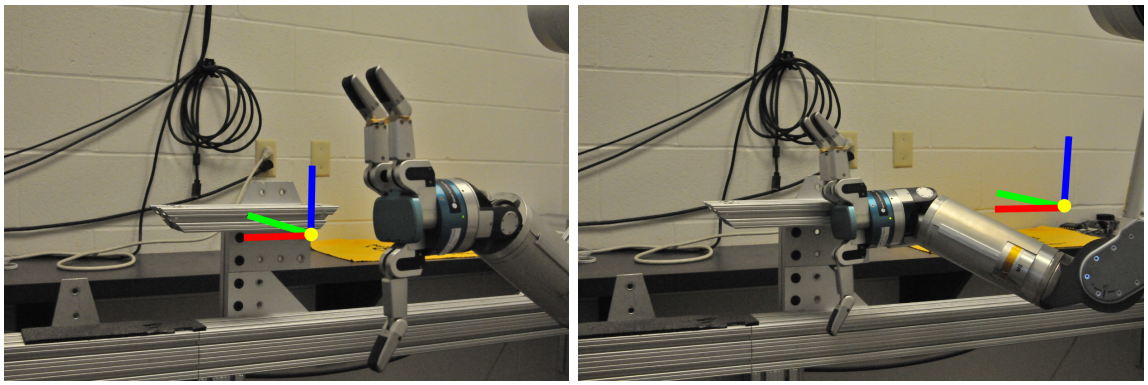


Figure 6.11: Experimental setup for response to step velocity input with a collision.

6.6 Discussion

In this chapter we have presented a novel application of the recursive Newton-Euler dynamics algorithm in which the dynamics of a kinematic chain robot can be masked.

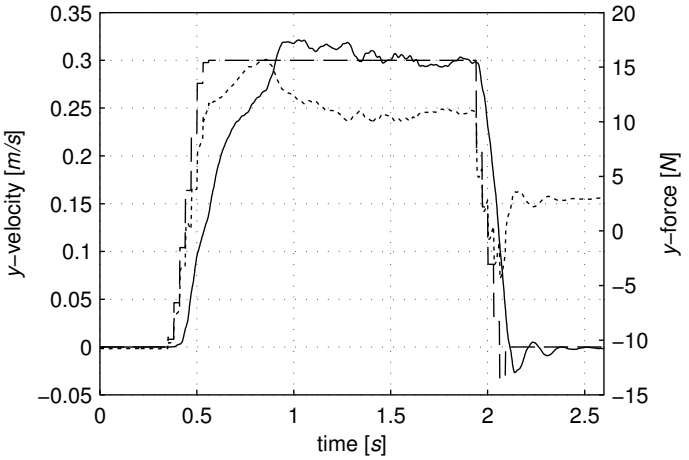


Figure 6.12: Kinematics and commanded force during uninterrupted movement. The solid and dashed lines show the actual and commanded end-effector velocities. The dotted line shows the commanded end-effector force.

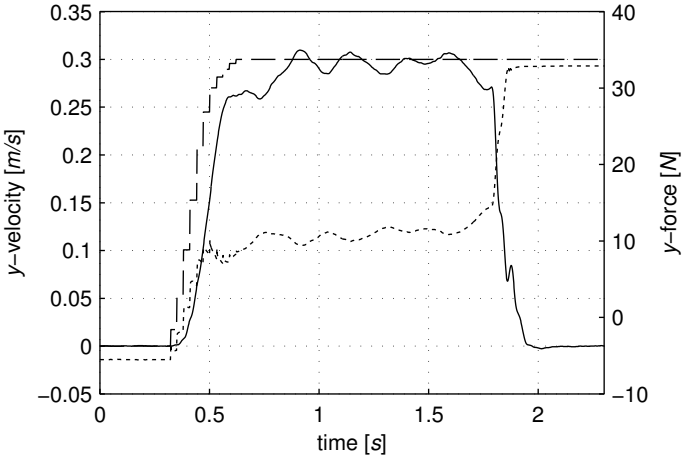


Figure 6.13: Kinematics and commanded force during collision. See Fig. The solid and dashed lines show the actual and commanded end-effector velocities. The dotted line shows the commanded end-effector force.

We then showed how the RNE equations allow access to the dynamics of each link in the kinematic chain so that desired behaviors could be inserted using the concept of virtual links. The principle of superposition of impedances allowed multiple simultaneous dynamic behaviors to take place at different points in the chain. We then showed a specific type of behavior in an impedance controller that also allowed for velocity control of the robot endpoint. Different behaviors were tested using this controller with a 7-link robot.

While compensation for the friction and the other terms of the model could improve its performance, this can significantly increase the complexity of implementing this control scheme in real situations. (Approximate) kinematic and inertial parameters of the robot can be taken directly from the design model of the robot, and joint positions can be read directly from joint encoders. Accurate high-speed joint velocity and acceleration, however, can be difficult to measure robustly and may require specific methods to measure with specific robots. Likewise, friction parameters are difficult to estimate and can be tricky to measure. We based the control model here on the parameters that are easy to obtain, and the result has favorable characteristics for performing control tasks. We showed in simulation that if more accurately modeled performance characteristics are required, friction and other components of the dynamics can be incorporated into the model.

Robert Rasmussen was extremely helpful in performing these experiments and preparing some of the figures in this chapter.

Chapter 7

Overall Results

7.1 Introduction

Since the discovery of models relating arm movement to neuronal population activity in the motor cortex [32], there have been efforts to recruit this activity to control external devices directly with the brain. Brain-computer interface (BCI) prosthetic devices have the potential to aid the over 250,000 people in the US alone who suffer from debilitating motor deficits such as spinal cord injury and ALS [131]. BCI systems can do this by bypassing motor lesions located outside of the CNS; cortical activity reflecting subject intent can instead be expressed directly by action in a machine.

Progressively more sophisticated BCI systems have been demonstrated over the last decade, moving from 2 and 3-dimensional control of a cursor on a computer screen [106, 113] to indirect [13, 112] and finally direct control of a 4 degree-of-freedom (DoF) robot arm [121]. In the 4-DoF experiment, the use of an anthropomorphic physical arm facilitated the monkey incorporating arm behaviors related to its physical structure into its control. As we progress towards the control of increasingly sophisticated prosthetic arms, the related concept of embodiment gains importance; BCI prosthetic devices that incorporate features of natural movement may be more easily mapped into familiar patterns of neural control. Natural arm movements integrate hand rotation with translation and are characterized by fluid transitions between arm and hand motions when reaching to and interacting with objects. While these types of movements are desired in prosthetic control models, no prosthetic arm will be able to directly reproduce the exact movements and dynamics of a specific subject's natural

arm. Therefore, effective models for BCI control will incorporate general principles of natural movement without reliance on exact replication of subject physiology. We have recently developed a brain-computer interface robot control system that directly addresses these issues. Recording from single units in the motor cortex, we can now demonstrate brain control of a prosthetic arm that exhibits the following features:

(1) simultaneous 7 DoF brain control over robot hand translation, rotation, and finger aperture, (2) integrated kinematic (movement) and dynamic (force) BCI control of a prosthetic robot, (3) simplified methods for constructing cortical extraction models based on only observation of the moving robot, and (4) a generalized method for training subjects to use complex prosthetic robot devices using a novel form of operator-machine shared control.

7.2 7-DoF BCI Experiment

Two monkey subjects (F and G, both naive to brain control) were implanted with 96-channel chronic intracortical microelectrode arrays. Monkey F had a single array in the right hemisphere while G had three arrays implanted in both hemispheres. Array locations for both monkeys are shown in Figure 7.3. Cortical activity captured with these arrays was used to drive the movement of a 7-DoF robot arm with 4-DoF attached robot hand (Barrett WAM arm and Hand, Barrett Technologies, Cambridge, MA) that was mounted to the right of the subject (Figure 7.1) in the experiment. The arrangement of the links of the prosthetic robot were anthropomorphic except at the robot wrist, which replaced the abduction/adduction joint at the hand with an additional axial rotation joint. The reachable space of the WAM endpoint was similar to that of the human arm with different joint space configurations in each endpoint pose. The brain-controlled movement variables in the experiment were the Cartesian linear and rotational velocities of the whole hand, along with grasp aperture. Finger movements of the hand were not always available, so that many control sessions were reduced to 6-DoF execution (references to the 7-DoF task include some sessions in which grip closure was not performed).

Spiking activity in the brain was recorded using RZ2 (TDT Inc., Alachua, FL) signal processing systems. For monkey F, a threshold at each electrode channel was fixed at 5 to 7.5 times the standard deviation of measured voltage deflections and all threshold crosses were counted as a neural spike of a single cortical unit. Monkey

G units were at first manually sorted from within waveforms crossing the threshold, but later automatic threshold crossing was used with a small number of additional units (10-15) manually sorted. Spike events were transformed to firing rates using inter-spike interval times in 30 ms bins. Firing rates were filtered with an exponential filter (width 15 bins, decay constant 0.95) throughout the experiment.

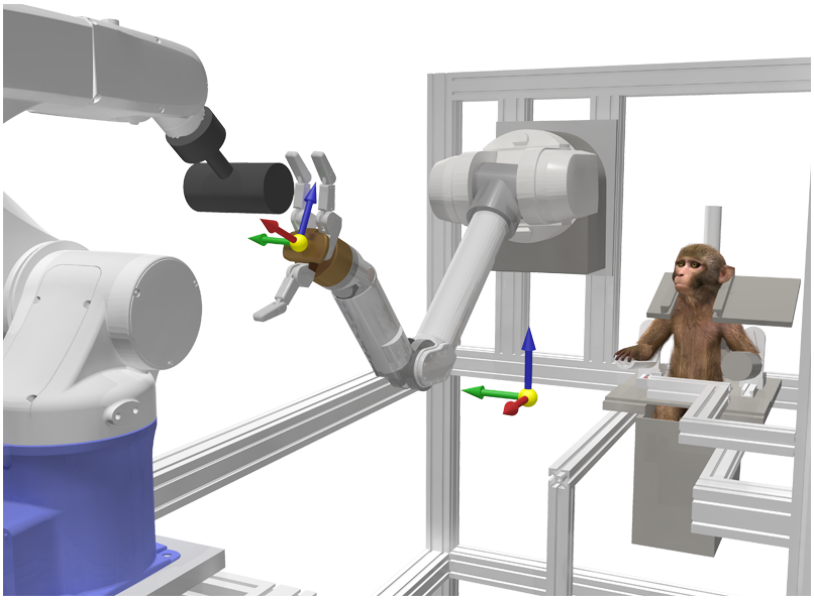


Figure 7.1: The BCI control experimental setup with monkey, prosthetic, and targeting robot. Axes representing robot base and hand coordinate frames are superimposed. Red-Green-Blue arrows correspond to the X-Y-Z axes of experiment.

7.3 Cortical Decoder Calibration

During the first part of each daily control session, subjects observed the prosthetic robot as it autonomously executed oriented grasping tasks. Each movement execution was prompted by the monkey reaching out and pressing a button with its own unrestrained right hand. The presentation robot then moved the target to one of six orientations while the prosthetic robot moved to one of six positions on the edges of the robot workspace (Figure 7.2 illustrates the set of starting positions and target orientations). Next, an audible cue was emitted to begin the trial. The robot hand autonomously translated, rotated towards, then grasped the target as measured cortical spike rates and robot movement velocities were recorded in 30 ms intervals. When

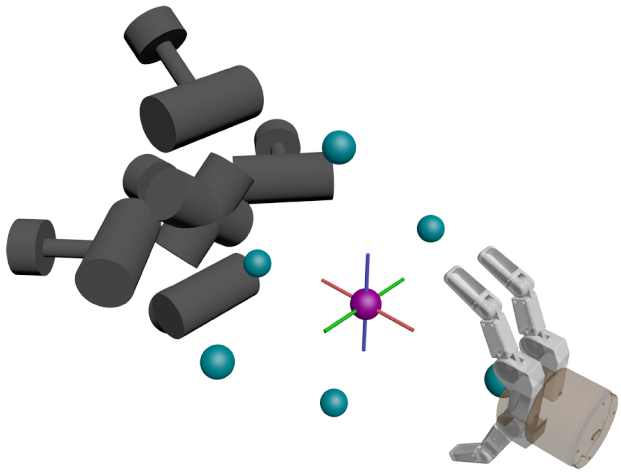


Figure 7.2: Starting positions of the hand (spheres) and orientations of the target object. 2 axial rotation targets are superimposed in the center of the target set. The robot hand moved backwards from the sphere closest to the center of the targets before orienting.

the robot grasped the target, a liquid reward was delivered to the monkey, the robot positions were reset, and the system waited for the next button press. Trials timed out and were failed after 12-14 seconds if the hand had not grasped the cylinder. Trials were also aborted if the monkey stopped attending to the task or was moving in the chair.

At the end of 18-40 observation trials, a cortical decoder was calibrated to produce a model for generating movement commands from sampled spike rates. Sampled robot velocities were filtered with a 15 bin boxcar filter. Samples of firing rates and velocities in which a significant amount of movement was taking place (> 0.02 m/s or rad/s total velocity) were used for calibration. 1000-3000 samples were normally used for model calibration in each session.

The calibration process first fit a preferred-direction (PD) model of the relationship between individual unit firing rates and robot movements that included the 3-D linear and rotational velocities of the hand (Equation 7.1). A separate PD model working in parallel was used to relate cortical firing rates to 1-D robotic hand aperture velocity (Equation 7.2). Control over this DoF was separated from the transport control equation because grip actuation during calibration was identical in each trial and occurred after transport and just before reward delivery; some cortical activation

during gripping appeared to be related to rising reward expectation which may not have fit a linear model incorporating all 7 degrees of freedom.

$$\lambda_i = a_i v_x + b_i v_y + c_i v_z + d_i \omega_x + e_i \omega_y + f_i \omega_z + b_{i0} + \epsilon_i \tag{7.1}$$

$$\lambda_i = g_i \gamma + g_{i0} + \epsilon_{gi} \tag{7.2}$$

Sampled linear (v_x, v_y, v_z) and rotational (ω_x, ω_y, ω_z) velocities were related to firing rates by linear (a_i, b_i, c_i) and rotational (d_i, e_i, f_i) PD coefficients that comprised 6-D preferred directions for each unit. For the grip model, γ is the sampled hand opening and closing speed, while g_i is the 1-D preferred direction of neuron i for grip actuation. By using linear and rotational velocities as the controlled motor quantities in Equation 7.1, the 6-D PD model can be described as operating over a 6-D vector space in which linear and rotational quantities are controlled. In this model, the well-known 3-D linear velocity vector preferred-direction model can be extended to rotational DoF with no loss of descriptive power. This is not the case for orientation (heading)-based models, or for other representations of rotation such as Euler angles (yaw-pitch-roll) or quaternions.

PD model calibration was followed by application of the minimal variance Optimal Linear Estimation (OLE) method [17, 96] to produce a decoding matrix that transformed spikes to movement commands for the remainder of the experimental session. OLE is similar to the well known Population Vector Algorithm (PVA) but avoids bias introduced by nonuniform distributions of preferred directions.

7.3.1 Observation-Based Calibration Results

Two examples of modulated neural activity recorded from a subject during observation of automated robot movements are shown in Figure 7.5. These examples were taken from a session in which the subject later performed full 7-DoF BCI robot control. No natural movement task was performed by the monkey previous to observation sessions, so that the observed modulated activity shown here was a product of spontaneous activity elicited by observation of the robot, or from an internal model of robot control originating from observation.

Goodness-of-fit (R^2) statistics for the PD model fit from observation across multiple sessions indicated that the representation of linear and rotational motion was well

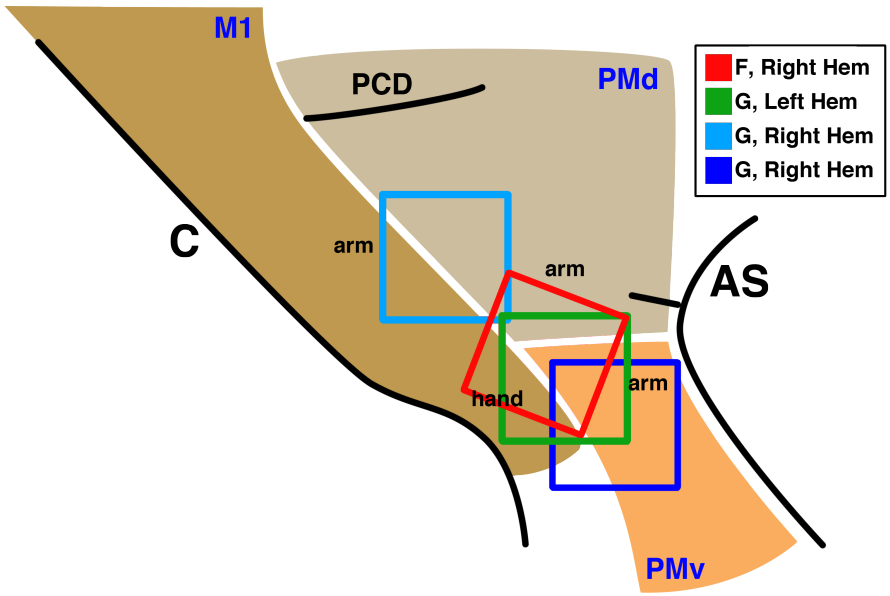


Figure 7.3: Locations of cortical implants in both monkeys. Square outlines are silhouettes of Utah electrode arrays. Blue = monkey G right hemisphere arrays. Green = monkey G left hemisphere array. Red = monkey F right hemisphere array. M1 = primary motor cortex. PM(d/v) = dorsal/ventral premotor area. C = central sulcus. PCD = precentral dimple. AS = arcuate sulcus. Topography adapted from He et al [37].

distributed among all units, rather than clustered into groups of neurons representing distinct linear or rotational movement types. Typical goodness-of-fit parameters for the model of neural activity to robot movement during observation periods of one control session in each monkey are represented in Figure 7.4, showing a widespread distribution of observation-modulated activity among primary and premotor brain areas and hemispheres. A comparison of linear and rotational partial R^2 (coefficient of partial determination) statistics for individual units is shown in Figure 7.8. Linear and rotational tuning parameters (linear and rotational preferred directions) from this session are in Figures 7.6 and 7.7, showing the distribution of PDs for both types of movement.

7.4 Brain-Computer Interface Operator Training

After the model calibration period of each session, the experiment entered a training phase in which the monkey learned to use the calibrated model to perform the

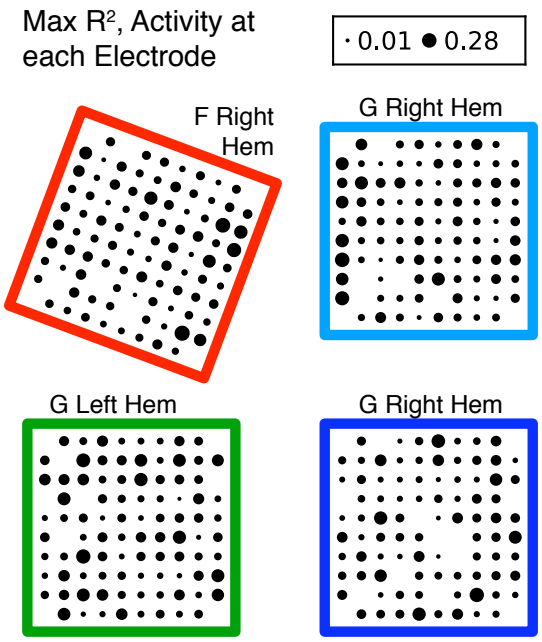


Figure 7.4: Maximal R^2 statistic for units at each channel for fit to 6-D movement model. Color outlines correspond to arrays in Figure 7.3.

oriented grasping task. When naive monkeys were first introduced to the task, they started by first performing 3-D linear control while the 4 remaining DoF of the oriented grasping task were controlled automatically. After proficiency was gained, the monkeys performed the 3-D rotational portion of the otherwise autonomously controlled task. Finally, the 6-DoF control task was performed under brain control. Grip control was also performed during all phases when the robot hand was available.

For each DoF under brain control, an operator-machine shared control system adaptively modified subject control commands by comparison to sets of model commands provided by an autonomous software program that modeled the motor task in real time. The autonomous controller generated multiple movement commands simultaneously that represented motion toward and approximation of the range of robot poses that would complete the grasping task. This reflected the fact that there are many ways that hands grasp objects, so that we allowed that flexibility in using the brain-controlled arm and hand.

The shared controller decomposed the subject commands into (1) the portion projected onto the submanifold of (pointing towards) autonomous controller com-

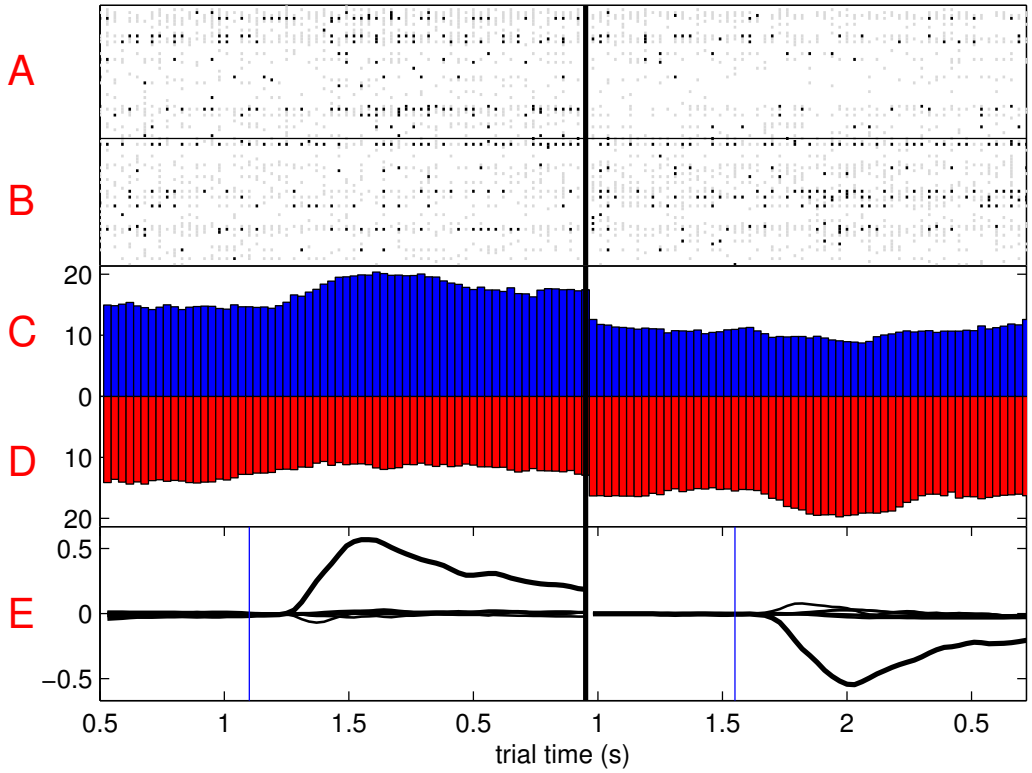


Figure 7.5: Spiking activity during observation of robot movement during two individual trials recorded from monkey G. In this session, the monkey progressed to full control of the robot. Row E shows traces of 6-D robot velocity in meters and radians. Highly deflected traces are the rotational velocity equivalent to hand yaw. The blue vertical line shows the target presentation time. Row A shows pseudoraster (30 ms bins) of spike activity for units modulated for positive yaw rotation (turning left). The darkness of each mark represents number of spikes recorded in the bin. Row B shows pseudoraster for units modulated for negative yaw rotation (turning right). Row C is histogram of summed spikes from row A. Row D is a histogram of summed spikes from row B.

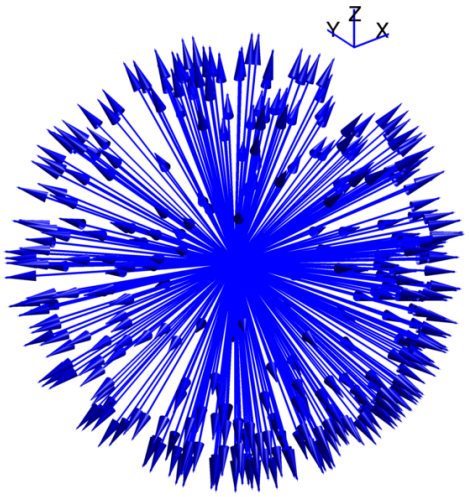


Figure 7.6: Distribution of linear preferred directions of individual units from a monkey G calibration based on observation data. An increase in spike rate for each cortical unit contributes to moving the overall robot command signal in the direction of its corresponding preferred direction. PDs are shown for units that had significant fits to the 6-D multivariate regression ($p < 0.05$, Equation (eqno))

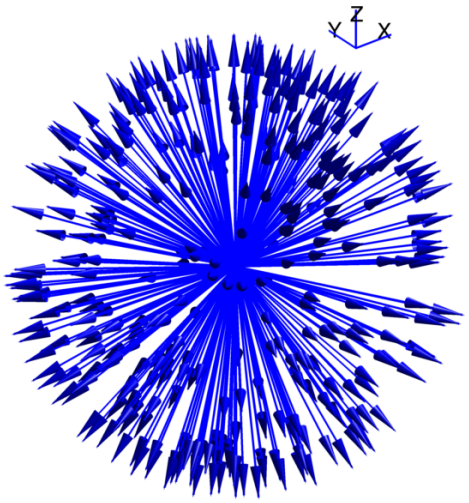


Figure 7.7: Distribution of rotational preferred directions for same units as Figure 7.6. Increases in spike rate for each unit contribute a rotational command with an axis in the direction of the unit's PD. The mean angle between linear and rotational preferred directions in this session was 96.79 degrees, compared to the 90 degree expected value for the angle between two random 3D vectors.

mands, i.e. the ideal commands and (2) the residual of the brain-control command after subtraction of 1. While (1) was always transmitted to control of the robot, (2) was multiplied by a set of shared control coefficients $c_{p,i} (0 \leq c_{p,i} \leq 1, i = 1 - 7 \text{ DoF})$ that controlled the level of admittance of error in each controlled DoF. This is an extension of a type of error limiting shared control that was used for training by Velliste et al [121]. Using this system, robot movement was not driven directly by the autonomous controller during training, but instead provided a template for reducing noise and error in the brain-control signal. The use of multiple commands in the template pointing towards the boundaries of allowable grasps allowed flexibility in how the task was completed, while still providing overall assistance.

The autonomous controller was also aware of the solid-body characteristics of the target cylinder and hand; when the path towards task completion poses was obstructed (e.g. the hand was behind the cylinder), generated commands would point

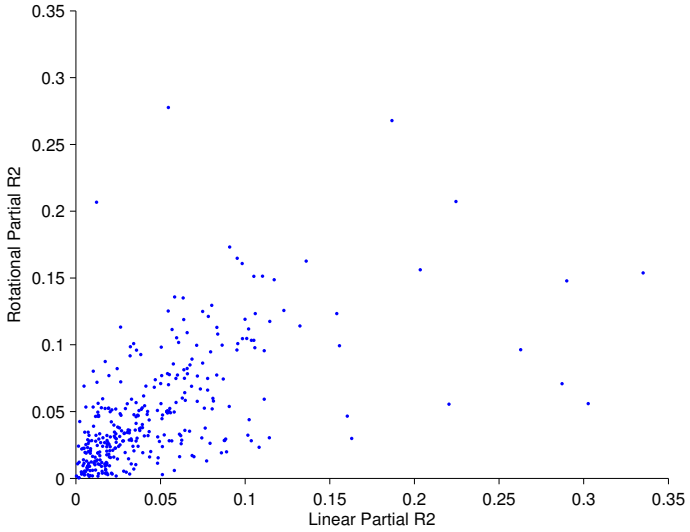


Figure 7.8: Partial R squared statistic for linear and rotational coefficients for units from Figure 4. The linear partial R squared measures the proportion of variance in the spike rate unexplained by the 3 rotational coefficients that is explained by the linear ones. The rotational partial R squared measures the variance explained by rotational coefficients not explained by linear ones. Unit activity was generally related to both linear and rotational DoF of observed robot movement.

toward intermediate poses preceding those from which the task could be completed. Automatic movements and the shared control algorithm at first provided a method for the monkey to learn the task sequence by demonstration and to connect grasping task completion with reward; this set up the operant conditioning paradigm used for training the monkeys. As the monkeys began to learn the relationship between intent and movement control, the shared control system prevented cortical activity unrelated to the task and control error from taking over robot movements. As the monkeys gained control skill, the shared control algorithm acted like a set of adjustable training wheels on a bicycle. By manipulating the underlying difficulty of the task using the c_p parameters, successful completion and reward delivery were maintained at gradually increasing levels of monkey skill, supporting monkey motivation throughout large numbers of control training trials.

The c_p coefficients were raised over time in accordance with heuristics developed by the investigators as a function of performance and perceived motivation level of the monkey. By independently changing the rotational and linear shared control coefficients, control challenges in each movement type could be effected, allowing skill to be acquired at different rates for each movement component. The goal of using

the shared control system was to gradually reduce its influence so that the monkey eventually fully controlled the device. Progress toward full control was represented by the maximum c_p parameter level at which task success could be maintained. Shared control provides a consistent method to constrain the complexity of the BCI system, which is a major factor for enabling training to take place when a large number of effector DoF are present.

7.4.1 Robot Motion Controller

PD model translational and rotational velocities were defined as the movement of the robot about a control point with a fixed relationship to the hand (Figure 7.1, coordinate axis attached to robot hand). Linear velocity DoF describe movement of this point through space, while rotation of the hand around this point was the basis for the rotational DoF. During grasping of an object, the control point was located at a location near the interface of the hand and the object. When the hand was approximated to the object, we described the coordinate system as "object centered" such that linear and rotational degrees-of-freedom approached independence. Translational and rotational DoF could be manipulated independently while maintaining a relatively consistent relationship between the hand and object in the other DoF. Use of this method allowed us to avoid problems with dependency between the linear and rotational DoF during grasping which would result from alternate motion parameterizations.

Velocity commands from the shared control system drove the movement of a torque-controlled robot arm using a novel low-level motion controller that smoothly integrated the control of arm kinematics and forces/torques at the hand. This system was based on a method for Cartesian impedance control [41] with a superimposed kinematic control model. Robot endpoint velocity was commanded directly during unconstrained movements, but these commands also indirectly controlled interaction forces when the arm was under an external load. This control model is similar to one proposed for the interaction of force and velocity in the cortical control of natural movements [117]. It allows an intuitive mechanism for transition between movement and force interactions, in addition to providing a controllably compliant and safe way to control the prosthetic arm in a realistic environment. While more traditional rigid robots and robot controllers would not be able to safely interact with objects, the monkey was able to use impedance-control characteristics of this arm to interact with

the cylinder target and other objects (including the monkey itself) even when its control skill was very low.

7.5 BCI Control Results

While control improvement was not a constant process, after monkeys started to achieve full 6-7D control, the shared control assistance parameters c_p could be brought up to full control over the course of as few as 15 trials (less than three minutes of control time). More often, however, this process took somewhere in the range of 40-60 trials to take place (about 5-8 minutes of control time). Figure 7.9 shows the relationship between success rate and shared-control assistance in a representative session. Even though the decoding model was recalibrated daily to account for changes in the specific cortical activity that was recorded at each electrode over time, a general increase in overall performance exhibited across sessions suggests that changes in cortical organization occurred to allow skill transfer from day to day.

39 sessions have been recorded in which the system was set up such that monkey F could perform 6 DoF control of the robot (finger actuation was not available throughout this period). While F had previously been able to perform high-quality control over the 3 linear DoF and 3 rotational DoF alone, simultaneous 6 DoF control has been a substantial challenge for this monkey with a single array. The overall success rate for this monkey during 6-D training with and without shared-control assistance is 60.2%. This monkey achieved full control of 6 DoF, with no assistance from the FF system, in 138 trials in 3 sessions recorded to date. Only 11 of these full-control trials so far have been successful. In 41% of full-control trials, however, the robot was within the target region with either or both the linear and rotational DoF at some point in the trial, suggesting that the monkey was nearly completing the task in a significant number of trials. The full success rate is projected to increase as the monkey gains experience in the task, but it is possible that performance will be limited by the number of recording channels available for on-line control.

Monkey G achieved full 6-DoF control of the device, in which no shared-mode assistance was supplied, in 49 separate recording sessions. Additional 1-DoF grip aperture control was possible when the robot hand was available, in roughly half of these sessions. When available, the monkey nearly always successfully performed a grip of the object if the task was otherwise completed using 6-DoF arm movement

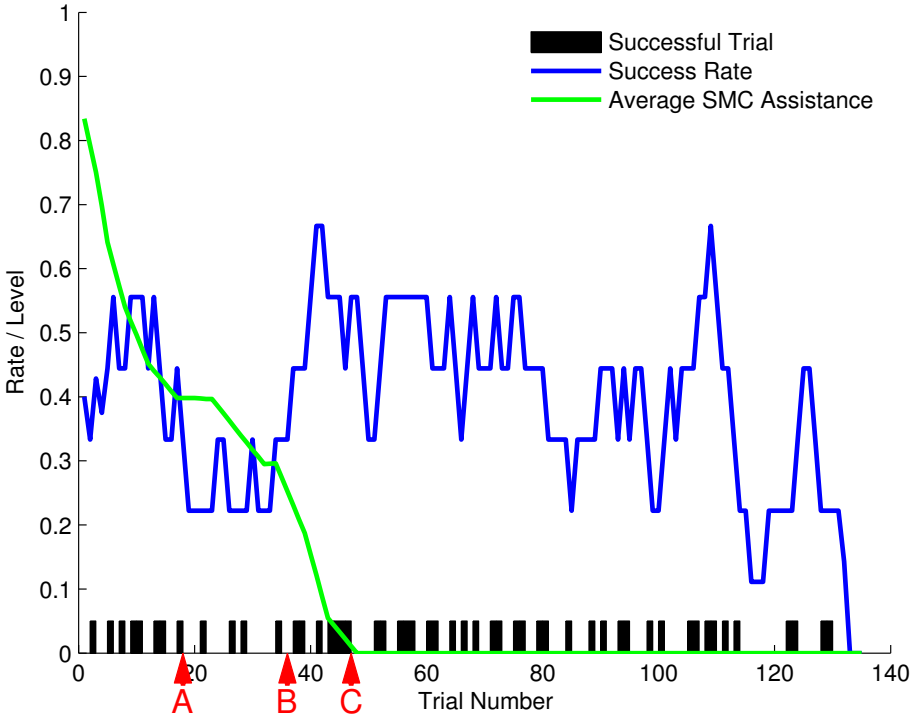
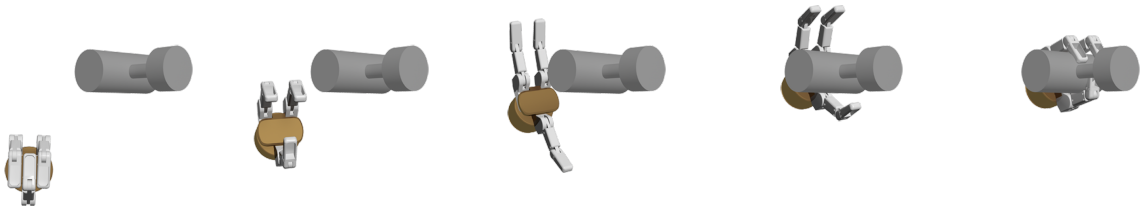
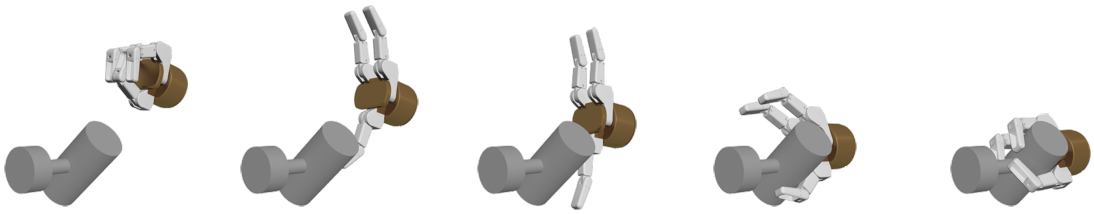


Figure 7.9: The relationship between shared control assistance and success rate during a representative control session. Black bars indicate successful trials. Binary success values were filtered with a 9-element boxcar filter to produce the success rate (blue trace). The average assistance levels $1 - c_p$ applied across all control dimensions over movement periods of each trial are shown as the green trace. From the beginning of the session, $1 - c_p$ parameters are decreased rapidly to match control task difficulty to monkey control skill (point A); they were then held at a relatively challenging level (A to B) as the monkeys learned the calibrated parameters. When the success rate of the monkey began to increase due to learning of the calibrated neural extraction model, the assistance level was quickly lowered to 0, so that the subject fully controlled the device for the remainder of the session.

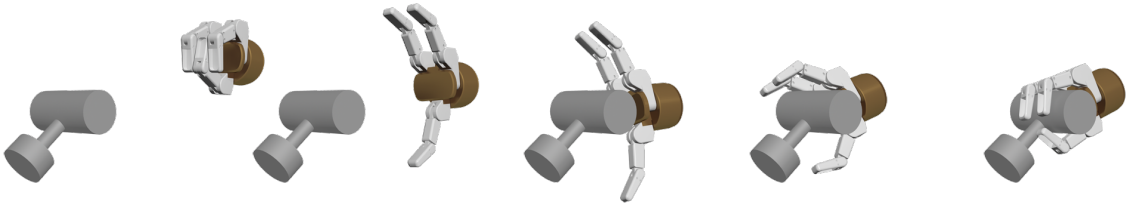
control. A set of recorded 7-DoF brain-controlled trajectories from a contiguous portion of one session are shown in Figure 7.10. The monkey would often, but not always, begin reaches by focusing primarily on operating the linear DoF, followed by rotation of the hand towards the target. Profiles of the linear and rotational distance to target from a set of control trials is shown in Figures 7.11 and 7.13. After the monkey became proficient in using the device, linear velocity profiles of movements usually had bell-shaped profiles (Figure 7.12). In each full control session, the monkey was able to achieve bursts of successful task completion, but control skill was not the only factor governing task success; monkey attention and behavior were a major factor in the success rate throughout trials.



(a) Upward linear, left yaw motion



(b) Downward linear, right twist motion



(c) Forward linear, downward pitch motion

Figure 7.10: Three successful 7-DoF brain control trials with movement in different linear and rotational dimensions.

In addition to allowing trials to continue through frequent collisions between the robot arm and objects, controlled compliant force interactions with environmental objects augmented the realism of the overall task and enhanced the performance of control trials. For instance, both monkeys were observed to use the vertical extension

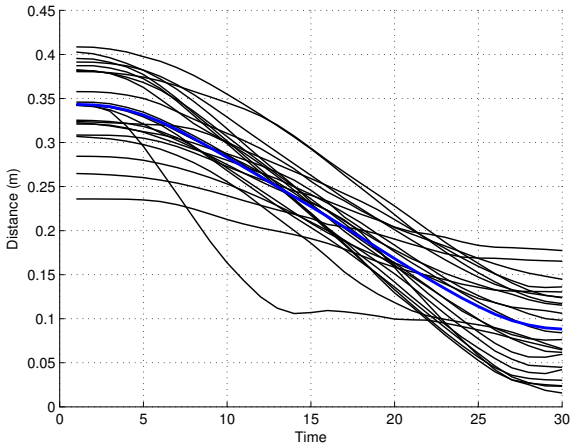


Figure 7.11: Linear distance between the robot hand and the center of target locations in 24 fully brain-controlled successful grasping trials from a contiguous portion of a single session with monkey G. The portion of each trial from when the hand was the farthest distance to the target (in distance or angle) to when it moved near to the target center was isolated and interpolated to a uniform time scale across trials. The average distance from the target (blue line) at the starting position was 34.3 cm and 9.0 cm at the end of movement.

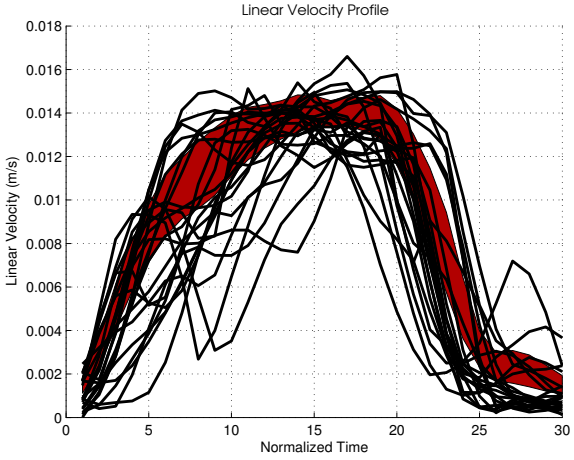


Figure 7.12: Linear velocity of the trials shown in Figure 7.11.

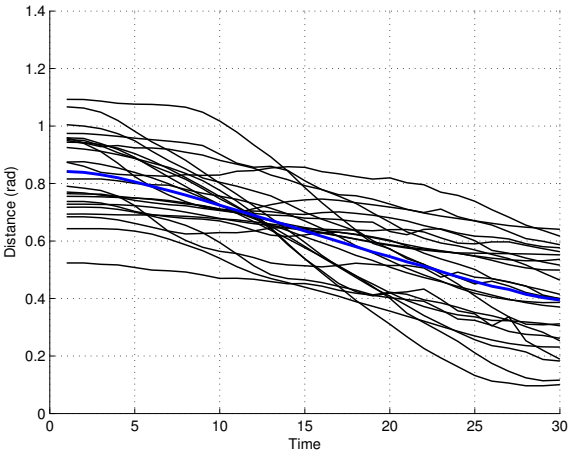


Figure 7.13: Rotational difference between the robot hand and the center of target locations in the same trials shown in Figure 7.11, using the same methods for normalizing the time scale. Rotational movement was more difficult to analyze; it is possible to successfully grasp the cylinder at some individual poses that are at a considerable angle from the central target rotation, so that while the overall distance to target went down in each trial, the relative deflection from the target was less than in the linear DoF. This does not directly reflect the actual overall angle of orientation change of the hand during each trial, which is often larger. The average starting position distance (blue line) was 0.84 radians (48.1 degrees) from the target. At the end movement in each trial, the mean distance from target was 9 cm and 0.394 radians (22.6 degrees).

of the fingers and linear translation to tilt the hand towards pitched targets. The hand often became stuck behind the target cylinder; this was allowed to happen freely. Both monkeys were observed to pull the hand backwards in cases where the robotic compliance behavior of the robot would allow it to acquire the target, or around the object when it would not. The presence of these and many other unrehearsed movements involving control of forces suggest device embodiment as found in Velliste et al, which can be extended to dynamic interactions between a brain-controlled limb and its environment. Please see Chapter 9 for further discussion of these overall results.

Chapter 8

Detailed Methods

In this chapter the experimental system and procedures that were used to implement the BCI system will be described in detail. This chapter will first present the surgical procedures performed to implant the microelectrode arrays, followed by the physical setup of the experiment and the design of the software architecture that was used to conduct brain-control sessions.

The brain-computer interface system described in this dissertation was not the product of a single design that was then executed using two monkeys. Due to the closed-loop nature of brain-computer interface experimentation, much of the development of the system described here was the result of trial and error based on long experience working with individual subjects. This evolutionary, experience-based style of development echoes that of previous successful 3D cursor and robotic BCI experiments [113, 121]. The heuristic nature of arriving at a place where a desired end result can be obtained has been a large part of BCI, which will continue as it moves into the realm of clinical therapy. Many of the approaches and steps in this chapter especially would be very difficult to prove conclusively as superior to some other approach. To do so would require a series of far more controlled experimental conditions focused on individual aspects of what was required to develop skilled prosthetic robot control. However, we attempt to present some of the reasons why certain approaches were taken and some of the history of how aspects of the system evolved. Along these same lines, there is some detail in this chapter on system components that does not relate directly to the goal of BCI control, but may help others facing some of the practical problems involved in implementing similar experiments.

8.1 Surgical Procedures

All surgical procedures were performed using sterile technique in an approved animal surgical facility and following surgical protocols approved by the University of Pittsburgh IACUC committee. Monkeys were sedated with ketamine HCl prior to receiving isoflurane for intrasurgical anaesthesia. Rhesus monkey G was implanted with two "Utah" 100-lead chronic cortical microelectrode arrays (Blackrock, Salt Lake City, UT). An incision was made roughly following the midline of the skull and the skin retracted to expose the cranial vault. A craniotomy of roughly 5 cm diameter was opened that was centered over the arm and hand area of the primary motor cortex. Removing the free bone flap from the craniotomy, the dura was incised on three sides of the craniotomy, creating a dural flap that was reflected to expose the brain. The central and arcuate sulci were visualized within the craniotomy, as well as the precentral dimple. The goal areas for implantation were the arm and hand regions of the primary motor cortex. Array 1 was placed on the precentral gyrus between the dimple and the genu of the arcuate sulcus. Array 2 was placed along the central sulcus lateral to the arcuate genu (Figure 8.1). A groove for each wire bundle was drilled into the edge of the craniotomy. Wire bundles leading from each array were placed into each groove and the attached pedestal-type connectors were fixed to the skull with bone screws near the anterior and posterior cranial midline. The dural flap was sutured back into place at each corner, then the bone flap was replaced and held with titanium straps. Dental acrylic was poured between and around the two pedestal implants and additional screws to provide additional fixation and protection to the implanted arrays and wire bundles. Pedestal connectors for each array were placed to the left of the midline anterior and posterior to the craniotomy and attached to the skull with bone screws. Additional bone screw anchors were inserted around the skull between the pedestals and the craniotomy site, surrounding the path of the wire bundle. Acrylic was poured in these areas to further fix the pedestals to the head as well as protect the wire bundles. The edge of the acrylic formed a curved border against which the retracted skin was pulled up and tightened with a purse string suture. The sutures were removed 7 days later.

After six months of recording, the M1 medial "arm" recording array underwent significant signal degradation from an unknown cause. An additional surgery was performed to implant two arrays in the right hemisphere at locations similar to the left hemisphere arrays, but with the lateral array placed further laterally and rostrally to potentially span part of the ventral premotor cortex. During the procedure, the

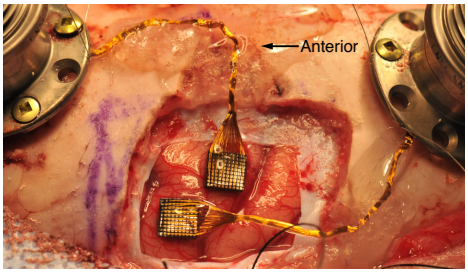


Figure 8.1: Monkey G left hemisphere implants.

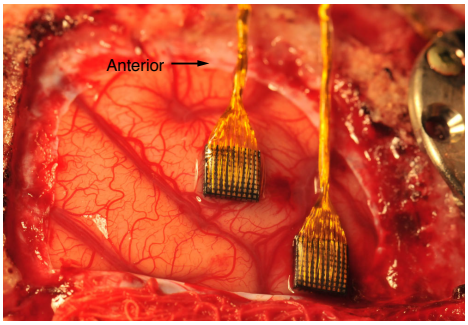


Figure 8.2: Monkey G right hemisphere implants.

wire bundle leading from the degraded array in the left hemisphere was cut and the corresponding pedestal was removed. The cause of signal loss in the medial array was not apparent as the first craniotomy was not reopened during the second surgery.

Similar procedures were used to implant monkey "F" with two arrays in its right hemisphere. The pedestals for the two implanted arrays were located again to the anterior and posterior of the craniotomy. Over time, the rear pedestal became more and more loosely attached to the skull. Eventually, it detached from the skull completely and recordings from that array were lost. The monkey then underwent an additional surgical procedure to implant an additional set of arrays in the left hemisphere, but a suitable site for mounting the additional pedestals could not be found, and this procedure was not performed.

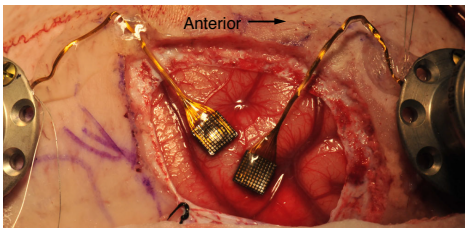


Figure 8.3: Monkey F right hemisphere implants.

8.2 BCI Experiment Physical Apparatus

8.2.1 Chronic Microelectrode Arrays

Potentials in the brain were recorded using Cereport (Blackrock Inc., Salt Lake City, Utah) chronic microelectrode arrays (Figure 8.5). These arrays had 100 1.5 mm electrodes, 96 of which were available for recording. $\sim$ 40 μm at the tip of each insulated electrode tip was exposed and recorded potentials at typically 100-750 kΩ of impedance. The primary motor cortex around which the arrays were implanted is a region about 3.0 mm in thickness that is relatively sparsely populated with cell bodies but with a high density of interconnected neural filaments. Broadmann area 4 of the cortex contains pyramidal cells in laminae II-VI that provide the substrate for regional output [87]. Most of these cells reside in layer III and V, while layer IV is absent in this area. These pyramidal cells are widely varied in size and are characterized by prominent apical dendrites. Some layer V pyramidal cells provide axonal output into the pyramidal tract that descends to the spinal cord. Non-pyramidal neurons present in this region are large "basket" cells in layers III and V, providing inhibitory inputs to the pyramidal cells [39]. The 1.5 mm electrodes extend to the layer III / layer V boundary of the 3.0 mm thick cortex [89]. Additionally, microelectrode arrays undergo a process of "sinking" into the cortex over longer periods of time (Figure 8.4) [35], which may place many of the electrodes well within layer V. This factor, combined with the large mass of apical dendrites from layer V neurons as well as the high degree of cross-correlation between proximal neurons in area 4 [63] suggest that significant layer V pyramidal cell activity is being recorded with the microelectrode arrays.

8.2.2 Neural Recording Apparatus

100-lead wire bundles led from the microelectrode arrays to titanium percutaneous connectors. These Cereport pedestals (Figure 8.6) were fixed to the skull and provided external connection to the recording apparatus. When the monkeys were brought into the experiment room, Cereport Plugs (Blackrock) were connected to the pedestal connectors. The plugs used in our experiment were fitted with custom ICS-96 headstage pin output to ZIF head stage connector outs (pedestal with adaptor is shown in Figure 8.7). Tucker-Davis Technologies (TDT, Alachua, Fl) ZCD headstages, which

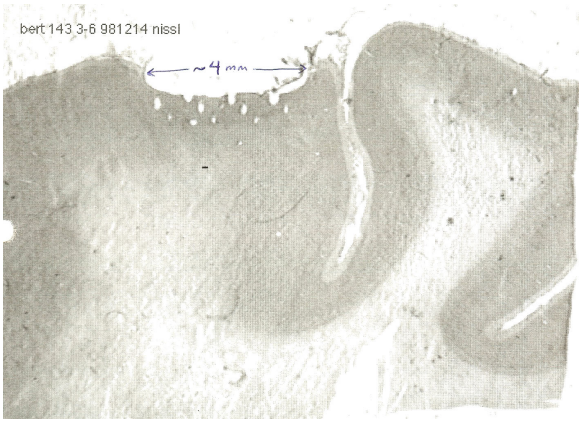


Figure 8.4: Histological slice of primate brain showing ≈ 1 mm deep indentation left by a "Utah" chronic microelectrode array.

performed the digitization of the voltages read at the electrodes (Figure 8.8), were clipped to the ZIF connector pins. Digital voltage readings from the ZCD headstages were processed by TDT PZ2-96 Preamplifiers (Figure 8.9) and sent from the recording room to a recording rack (Figure 8.10) by fiber optic connection. The fiber optic link isolated the recording system electrically from the computer processing station. In addition, the PZ2 Preamplifiers are battery operated so that measured potentials were isolated from electrical supply currents.

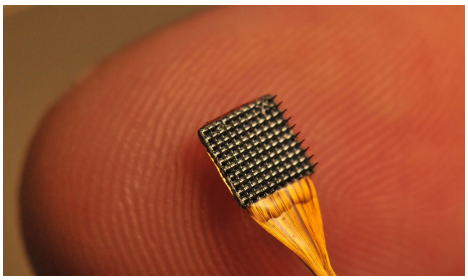


Figure 8.5: Utah 96-channel chronic microelectrode array.

At the rack, TDT RZ2 base stations (up to 3, one for each 96-channel array) received the digitized voltages at 48kHz. The RZ2 systems are programmable signal processing systems with 8 programmable DSPs apiece. The RZ2 systems interfaced with a PC which was used to program and configure their operation.

Spike count outputs from the RZ2s were sent to the spike processing system (Section 8.3.1) via UDP interface. The pathway in which spikes were translated into

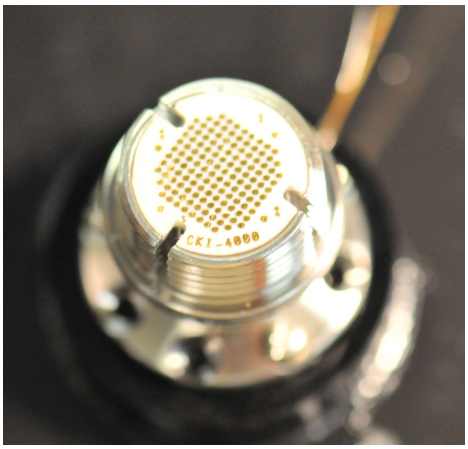


Figure 8.6: Pedestal connector for Utah microelectrode array.

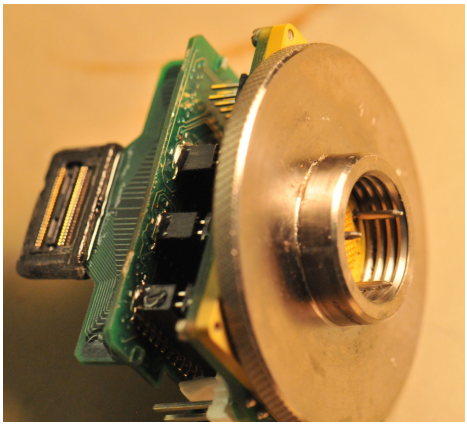


Figure 8.7: Cereport plug and ZIF adapter, showing pedestal connector and alignment pins.

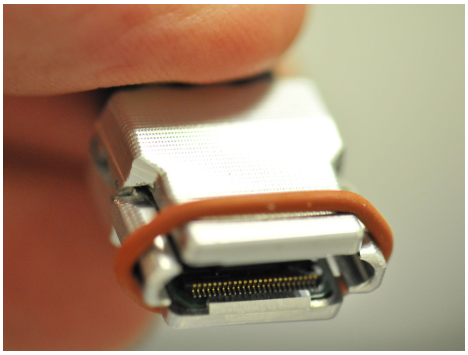


Figure 8.8: TDT 96-Channel ZIF clip pre-amplifiers.

Figure 8.9: RS3 analog to digital converters.

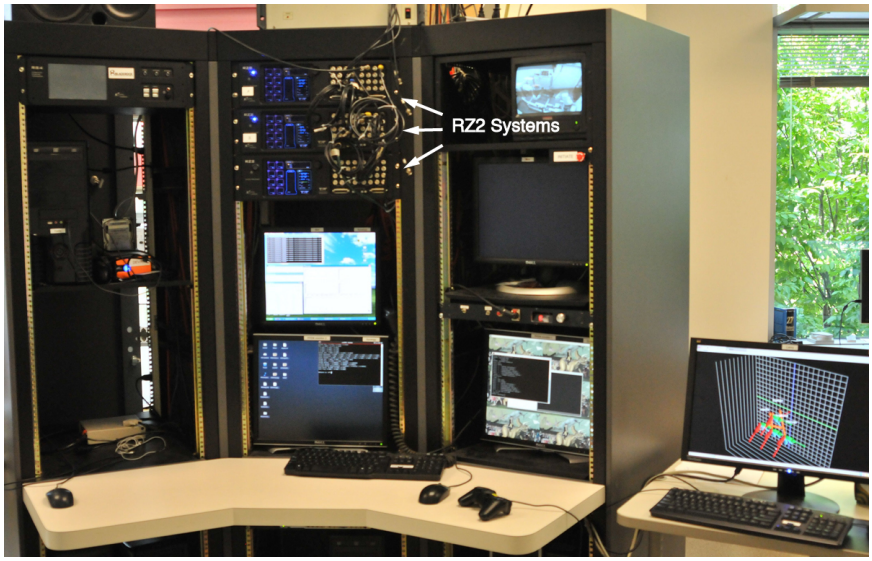


Figure 8.10: Experimental recording rack.

robot movement commands was distributed among 4 generic PCs using the RTMA communications layer (Section 8.3) over a gigabit TCP subnetwork. The bulk of the work was performed on three PCs running Ubuntu Linux, with a Windows PC dedicated to interfacing with the DENSO robot and the National Instruments NI-DAQ analog interface, for which only Windows APIs were readily accessible.

8.2.3 Experimental Room Setup

During experimental sessions, monkey subjects sat comfortably in a chair to which their rigid collar was attached. The left arm was restrained using a tube fit over the

left armrest, while the right arm was left free. The single arm was restrained only to keep the monkey from turning backwards or spinning in the chair. The chair was placed in a slot in an aluminum frame that provided a rigid mount for the robots, neural recording apparatus, and other physical systems used in the experiment (Figure 8.11).

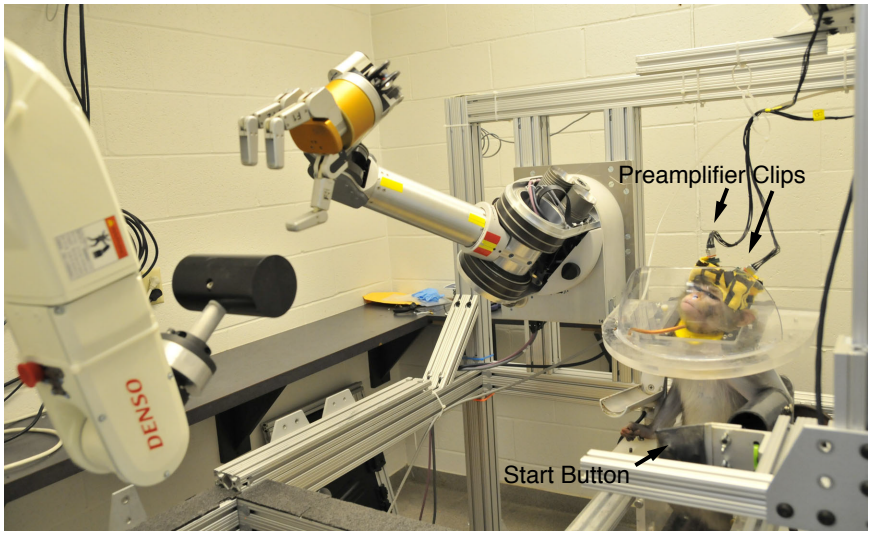


Figure 8.11: Monkey and robots in experimental frame, with smaller Start Button and recording system ZCD clips indicated.

A neck plate with integrated water tube was placed over the shoulders of the monkey. Instead of the neck plate extending laterally to block the monkey from reaching the water tube and neural recording connectors, a plastic motorcycle visor was attached to the neck plate. This minimal design reduced obstruction in the sensor range of an Optotrak movement tracking system mounted overhead, so that monkey hand movement data could be recorded during trials. The water tube was of a novel type that facilitated the reduction of compulsive licking behaviors in the monkey (Figure 8.12). The similarly minimal reward delivery device was a simple metal tube that led through the neck plate and to the rear of the monkey. The tube was connected to a small reservoir in the rear of the neck plate. The reservoir was filled by a siphon leading from a larger reservoir outside the room, with a solenoid-operated gating mechanism triggered by output from the software Analog Module (Section 8.4.2). A hole was drilled in the small reservoir so that once the reservoir was empty, additional attempts to draw reward would draw air from the hole. Very little water remained in the straw after water was retrieved, so that persistent sucking on the straw resulted in little to no water retrieval. This was a very useful design that drastically reduced mouth movements during the trials that had previously interfered

with the brain control recordings.

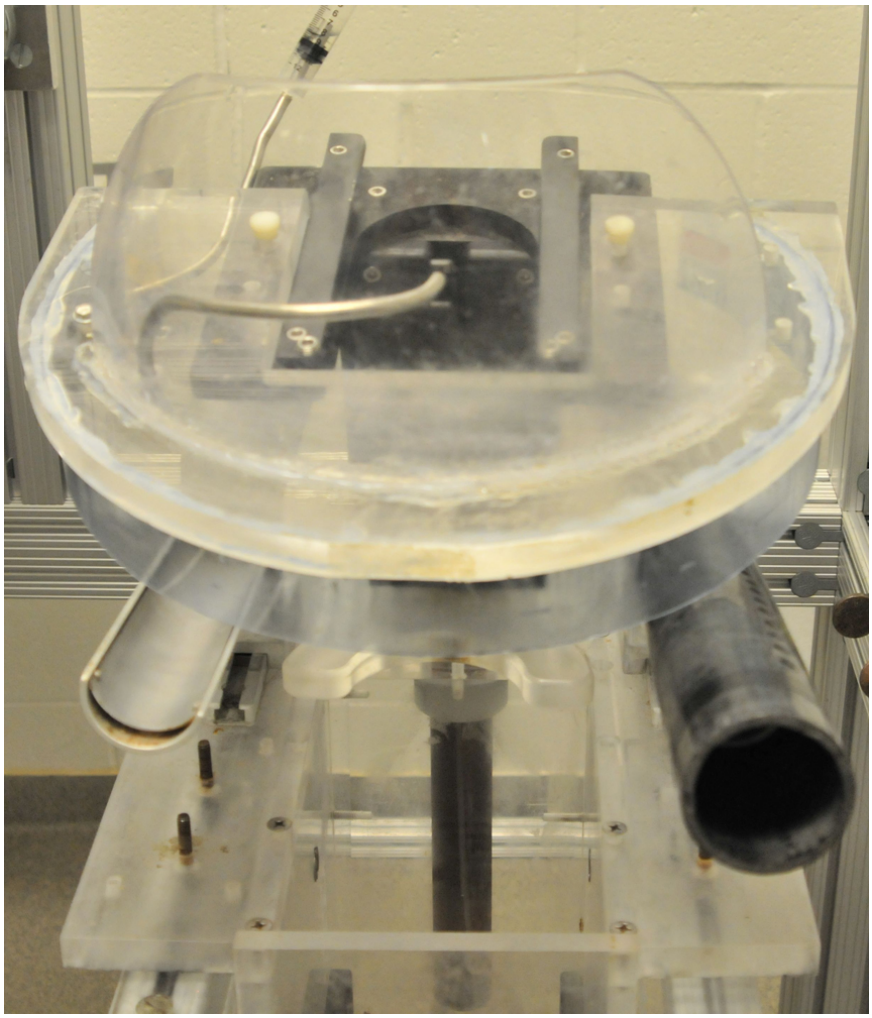


Figure 8.12: Monkey chair showing custom neck plate and water delivery apparatus.

A pushbutton mounted on a sliding arm was located in front of the monkey (see Figure 8.11). The button completed a simple circuit that was detected by the analog module (Section 8.4.2). The monkey pressed the button with its free hand to initiate brain-control trials. Because of the difficulty and length of individual BCI trials and sessions, it was important to ascertain when the monkey was willing to attempt a trial. Previously, trials had run autonomously and continuously; when the monkey was not actually attempting to control the arm, the arm moved anyway and the process did not contribute to establishing the belief in the monkey that the arm was under its control. Use of the button established some baseline willingness to participate in the task, even if attention continued to be a problem after trials had

begun (Section 8.3.6).

A Barrett Whole-Arm Manipulator (WAM, Barrett Technologies, Cambridge, MA) robot with attached BH262 hand was mounted to the right of the monkey and oriented horizontally such that it approximated the kinematic configuration of its right arm. The Barrett WAM (Figure 8.13) robot is a backdriveable 7 degree-of-freedom arm robot with roughly the same degrees of freedom of the human arm except at the wrist. In the robot, there is an additional axial rotation joint distal to the wrist flexion/extension axis, instead of the physiologic adduction/abduction joint. During operation, the torque at each joint of the backdriveable WAM is proportional to the current flow at a brushless motor, which an integrated controller regulates in fast (32kHz) PID loop. Thus, torque commands to the WAM are very good estimates of the torque present at the joint at the time resolution of the WAM control loop (500 Hz). The WAM motor controller and encoders communicated over a CAN bus with a 1.8 GHz Shuttle PC computer running Slackware linux. The WAM control loop (Chapter 6) was implemented as an RTAI (real-time linux) hard real time process preempting the linux kernel at 500Hz. The low-level throughput of the robot was limited by the CANbus communication speed, rather than the processing time of the control algorithm itself. With an improved communication system, the controller could run faster.

The Barrett BH262 hand is a 3-fingered non-backdriveable robot hand that provides direct control of the closure velocity of each finger and the spread of two of the fingers opposing the thumb. A distal joint in each finger moves at a speed proportional to the main joint, and during grasping the second joint will continue to close if the first segment of the finger is blocked in a "breakaway" mode. We created a custom control system using the Barrett real-time mode hand control protocol to simulate compliant hand control. Repeated maintenance problems with the BH262 Hand led to a significant number of recording sessions in which hand control was not performed. A more advanced replacement BH280 hand was later acquired and if shown in the figure, but was not used for control trials reported in this dissertation.

A DENSO 6-axis industrial-style robot arm was placed approximately 1 m in front of and facing the WAM robot. The target object mounted to the endpoint of the DENSO robot is a 3 cm diameter, 6 cm long cylinder, attached by an aluminum rod to the robot such that the long axis is perpendicular to the end link of the robot. The DENSO robot is an admittance robot that provides very high resolution positioning of the target, but is completely rigid. It was used to place the target and

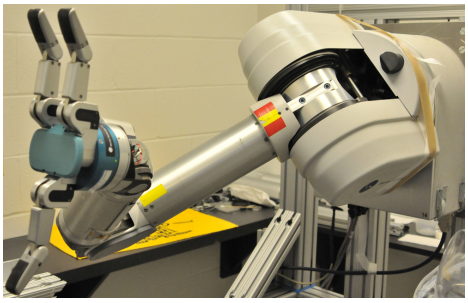


Figure 8.13: Barrett WAM robot and Barrett BH280 Hand.

functioned as a static obstacle during execution of each trial.

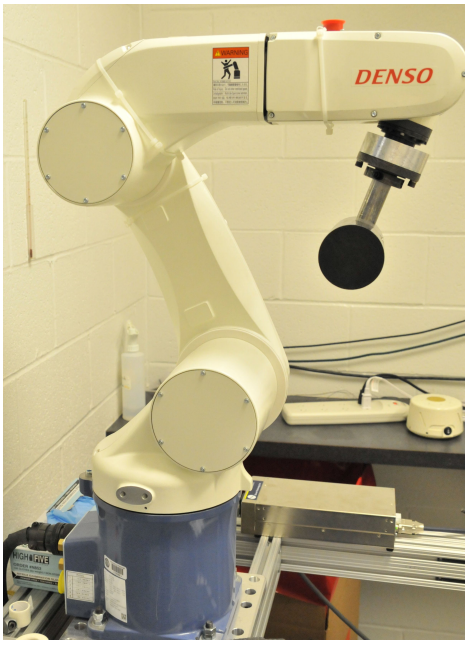


Figure 8.14: DENSO 6-axis robot.

An Optotrak 3010 kinematic recording camera set was placed along the ceiling of the experimental room (Figure 8.15). It was placed such that the free hand of the monkey and the monkey's head could be tracked without either robot arm being in the way of the line of sight of the system. We wanted to track the monkeys' movements as this is the first BCI experiment in which no motor training was used; since arm restraints were not required, we wanted to look at the relationship between the monkey's movements and movements of the prosthetic arm. We attached a 3-marker object to the monkeys' hands so that hand translation and rotation could be tracked, as well as a single marker on the head of monkey G. The Optotrak recording had mixed results; monkey G was unwilling to perform difficult brain control with

the markers attached. We reverted to collecting optotrak data at the conclusion of brain-control trials, in lieu of training the monkey specifically to tolerate the tracking markers. This data, along with Optotrak data from monkey F during BCI trials was recorded in upwards of 20 BCI sessions for later analysis.

Figure 8.15: Optotrak motion recording system.

The experimental frame was designed and constructed by Meel Velliste and Robert Rasmussen, then later modified by Sam Clanton. The reward device was designed by Zohny Zohny and Sam Clanton, with the basic water delivery system implemented by Meel Velliste, Andrew Whitford, and Chance Spalding. The button was made by Zohny Zohny and Sam Clanton. The Optotrak system was implemented by Zohny Zohny and Sam Clanton. The DENSO was installed and configured for the system by Meel Velliste.

8.3 Brain-Control Software Architecture

The overall system architecture of the BCI robot control experiment is shown in Figure 8.16. While the total system used was complex, it was designed as a distributed network of reactive agents, each agent performing a very limited set of specific tasks in reaction to messages sent over the network. The component modules communicated over a unified network message-passing architecture,"RTMA" (Real Time Messaging Architecture). Agents registered with the messaging system when they were run and signed up to receive messages of certain types that were put out on the network. They would then sleep until activated by one of these message types, execute some reactive procedure, emit further messages if necessary, and go back to sleep. This promoted rapid development and testing of experimental subsystems that could function as plug-in components for completion of some well-defined subtask. The overall system

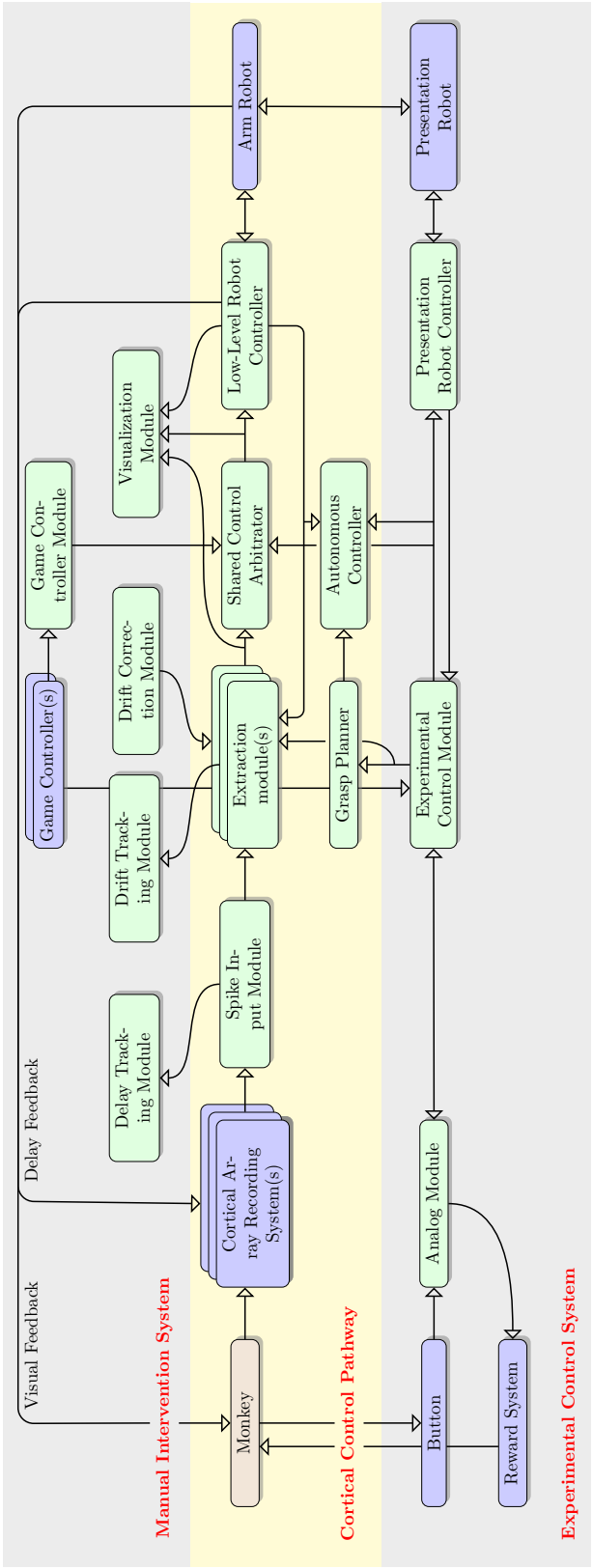


Figure 8.16: Schematic of complete Brain-Computer Interface Control System

would keep running while different candidate modules were plugged into the slots of Figure 8.16. Over the long (often 2-3 hour) sessions, observed problems could be reacted to by rapid implementation and substitution of modules providing solutions, in the best cases without breaking up the session or the monkey realizing the change. The modular communications and control architecture directly enabled the speed and flexibility necessary needed to finally achieve brain control of a 7-DoF arm. This system was intended as a prototype for systems that could facilitate the development of BCI control over complex robotic prosthetic devices in general, and is the general architecture that will be used for BCI robotic experimentation in our laboratory moving forward.

While the complete BCI system was associated with a large amount of processing and computing hardware, a streamlined post-development version of the spike count processing and control system could readily be put on a single computer. If thresholding-only approaches to spike counting are used, and the robot controller were on-board (as in more modern versions of the WAM), the BCI system could run on a single processor with little modification.

The overall system architecture is divided into the following parts:

Cortical Control Pathway This subsystem transforms brain activity to movement of a robot. This pathway includes the elements that perform this translation, as well as those that are necessary for enabling control through the shared control system described in Chapter 5.

Experimental Control System This part of the system controls the experimental procedure, including sequencing of brain-control sessions and interfacing with external devices that control the experimental paradigm.

Manual Intervention System While the entire control of the prosthetic robot arm takes place within the cortical control pathway, the system did not operate autonomously. Manual intervention by lab personnel was needed for adapting system parameters to the training requirements of the monkeys. Methods of shaping monkey behavior formed a large part of optimizing the feedback mechanisms used to drive the monkeys to develop control skill.

8.3.1 Robot Brain-Control Pathway

The core robot brain-control pathway was comprised of a set of modules that work in concert to form a kind of hierarchical control model that links brain activity to robot movement, ultimately transforming cortical spike rates to torques at the joints of a robot. At the top of the hierarchy, cortical single-unit spike rates were sampled. These spike rates were transmitted to a neural signal decoding system that once calibrated used these rates to compute high-level robot commands representing the desired linear, rotational, and finger aperture open/close velocities of a robot hand. At the same time, an autonomous controller ("Silicon Monkey") provided an idealized set of command signals to the robot in parallel with the cortical decoding system. Control signals from the Silicon Monkey and the cortical decoder provided input to the "Arbitrator" module that mixed the signals into a unified command that was sent to the robot. Output from the Arbitrator was sent to a low-level robot controller which translated the velocity commands into robot movement.

Robot Control Space

The primary messages communicated through the robot brain-control pathway were robot commands and feedback in the control space of the experiment. This control space was always a vector containing the 7-DoF linear, rotational, and grip velocity of the robot. An additional data element contained flags indicating which DoF were actively controlled by the controller from which the message originated. This facilitated a plug-in system for connecting controlling modules to the robot input and the shared-control arbitration system. Position feedback from the robot that was used by the Silicon Monkey was issued in a similar space that included a yaw-pitch-roll orientation reading.

Neural Recording System

Summary Voltages from chronic multielectrode arrays were measured and spike counts were calculated from discharge events from single neurons or groups of neurons. Spike counts were output in 30 ms bins over the experimental message-passing system.

Inputs Voltages in cortex

Outputs Spike counts

The RZ2 systems were programmed to isolate individual neural spikes from the incoming voltage streams from the arrays. A voltage threshold was set for each electrode; every time the recorded signal passed this threshold, an epoch of the waveform surrounding the threshold crossing was isolated. These thresholds were then classified as spike events or discarded, depending on the method of "spike sorting" used. Spike events were labeled as "units", corresponding to the activity of one or more neurons. Each unit formed a separate control input into the system, and up to 4 could be present at each individual electrode. Spike events for each unit were counted in 10 ms intervals and exported from the RZ2 systems via a USB interface. The process of spike sorting was done in one of three ways in this experiment:

Box Sorting Thresholds were set manually or at a multiple (5.5-7x) of the standard deviation from the mean of all activity recorded at the electrode. Epochs surrounding threshold-crossing events were isolated and displayed on a computer screen. The experimenter then used a mouse to draw boxes in two areas of the screen around overlaid epoched waveforms. These boxes corresponded to gates through which passing voltage traces were labeled; a waveform passing through both gates belonging to a unit was a spike event of that unit.

Thresholding Only This process was completely automated. Thresholds were automatically set to the mean plus some multiple of the standard deviation of voltages recorded at each channel (usually 5.5-7x). A single unit was then assigned for any theshold-crossing waveform at each channel.

Hybrid Approach This process was much like the Thresholding-Only type of sorting, but very large, clear individual units (up to 5-10 per array) were separated into separate units by the box-sorting process.

While the box sorting technique is a more traditional way of manually discriminating units, there had been a persistent belief that better control occurred when spikes were sorted liberally; that is, when many waveforms that did not directly resemble spikes were sorted into individual units and used for control. In a study from our lab, Fraser [29] had found that much of the same information is transmitted through electrodes if simple thresholding is used for spike discrimination. As the time overhead in sorting 384 channels in one monkey was having an adverse affect on monkey performance, we started to spike sort using the hybrid approach as a compromise. After success with this was achieved, we changed to a complete thresholding-only

approach for monkey F. Successful 7 DoF control occurred using all three methods of spike sorting. The thresholding-only approach is a method by which clinical brain-computer interface electrode systems could acquire signals on an ongoing basis with no manual intervention.

4 bits in each exported spike count packet were allocated to storage for each sorted unit so that up to 7 spikes per 10 ms period could be counted. Given the refractory period of individual cortical neurons, a maximum of up to about 3 spikes per sampling period was usually observed, with the majority of spike rates evaluated from units with either 0 or 1 spikes occurring per bin. The data blocks contained space for 4 channels per electrode, 384 4-bit values per array.

Exported 10 ms spike count packets were picked up by computer software module which accumulated sets of three 10 ms bins and exported the resulting 30 ms spike count packets over the RTMA network. Each receipt of 3 sequential UDP spike count packets initiated the cascade of control signals that ultimately resulted in commanded movement of the robot.

The circuits controlling TDT operation were designed and written by Andrew Whitford, Sam Clanton, Zohny Zohny, Meel Velliste, and Tim Tucker from TDT. The UDP-RTMA interface module was written by Andrew Whitford and Meel Velliste.

In the following sections, the individual modules comprising the brain control pathway will be discussed individually.

8.3.2 Decoder Module

Summary The decoder module operated in two modes. During an initial calibration phase, the module recorded real-time spike rates and velocities of the moving robot in order to train a cortical decoding model to translate spikes to movement commands. After calibration of this model, the module applied it to incoming spike counts to form real-time robot commands derived from neural activity.

Inputs Spike Counts, Robot Velocity Samples, Task Sequence Cues

Outputs Robot Control Commands

Spike count packets on the RTMA network were monitored by one or more cortical Decoding/Extraction modules (EMs) responsible for translating information into

robot control commands. As discussed, extraction modules first went through a calibration sequence, then post-calibration would begin to emit robot control commands upon receipt of spike counts in 30 ms intervals. Multiple decoders were active in the system at any given time; each decoder could be configured to handle a certain subset of the 7 controlled DoF, and multiple decoders could provide control commands for overlapping sets of robot DoF. This open system allowed for the process of incremental building from multiple 3-dimensional extraction models towards a more unitary extraction model (Chapter 5) covering 6 of the 7 total control DoF. Grip decoding remained separate because of some of the differences between whole-hand movement and grip in the experiment and reliability issues with the robot hand robot itself. Selection of which extraction modules were responsible for individual robot control DoF took place in the Arbitrator (Section 8.3.3), and was changed in real time to support experimentation with different cortical decoding models. Output not applied directly to the robot could be visualized for evaluation in the Visualization Module (Section 8.5.1).

Decoder Calibration

As outlined in Chapters 3 and 5, neural decoders were calibrated by recording spike activity that took place as the monkey observed the moving robot. During each experimental trial in this phase of the experiment, the arm robot sequentially opened its hand, performed a linear movement to one of 6 targets, a rotation movement to one of 6 targets, then gripped a cylindrical target object at that location. Trials in which the monkey was not paying attention to the robot were manually rejected using a hand-held controller (Section 8.5.2). Each of these periods of grip, linear, or rotational actuation was labeled by the experimental controller (Section 8.4.1) such that the extraction modules could interpret the type of data that they were receiving. Modules could then choose to buffer particular samples for calibration or not depending on whether or not they were configured to predict on the corresponding control DoF. This isolation of linear, rotational, and grip motion was an initial attempt to "divide and conquer" the task of 7-D control, where linear and rotational control could be established in isolation and then later integrated (see Chapter 5).

Spike rates were calculated from incoming spike counts using an algorithm that was geared towards real-time rate estimation with no delay. Packets of spike counts were transmitted relatively frequently compared to the average spike rate of many

of the units being sampled. Therefore, rather than calculating spike rate as the simple inverse of the spike counts, the Optimistic Rate Calculation Algorithm was used (algorithm 3) in which the algorithm calculated spike rates based on inter-spike intervals (ISI). If a sample arrived with a 0 spike count, the algorithm assumed that the rate was as high as the current rate, or that next spike would arrive during the next sample, whichever was lower. This allowed the spike rate to slowly decay over periods in which no activity is detected without discontinuities in spike rates that were not directly measured.

Algorithm 3 Real-time firing rate calculator based on inter-spike interval estimates. *counts* is a time series of spike counts, *rate* is the firing rate calculated for each sample. T is the sampling period (30 ms).

```
function OptFRCalculator(counts)
    τ ← ∞                          ▷ τ is the current sample inter-spike interval estimate
    τ0 ← ∞                         ▷ τ0 is a running estimate of optimistic ISI time
    for c in counts do
        if c > 1 then
            τ = T/c                ▷ ISI time for evenly-spaced spikes over multiple samples
            τ0 ← τ/2               ▷ Lagging-edge border time for evenly spaced samples
        else if c = 1 then
            τ ← τ0 + T/2
            τ0 ← T/2               ▷ Half-sample time since single spike centered in sample
        else
            τ0 ← τ0 + T
            if τ0 > τ then
                τ ← τ0
            end if
        end if
        rate ← 1/τ                 ▷ If no spikes have been seen, 1/∞ = 0
    end for
end function
```

Incoming kinematic data used for calibration was the real-time 7-DoF robot velocity reported directly by the low-level robot controller (Section 8.3.7). Buffered spike and kinematic data was held until the end of the calibration period. This was usually set to 24 or 30 trials, or $(4-5) \times 6$ repetitions of targets and starting positions requiring movement in both directions for each control DoF. The calibration procedure was performed during each daily experimental session. While the neural processes taking place in the cortex relating to control may not change between calibration sessions, recordings from the implanted arrays suffer from a process of

drift that does not guarantee consistent recordings over longer periods of time. On a day to day basis, the spikes observed during the sorting process altered in their apparent shape, new units appeared in the recordings, and a general overall change in the mixing of different unit signals occurred. Additionally, it was probable that the cortical processes involved in brain control underwent changes due to long-term plasticity as the subject learned the task. Thus, we attempted to track these changes daily through model calibration.

At the conclusion of model calibration, the decoding models began to emit movement commands in response to incoming spike rates from the neural recording system 8.3.1. The EMs emitted movement commands continuously; the arbitration system implementing the shared-mode control algorithm downstream controlled the degree to which these commands were applied to the robot (Section 8.3.3). The EM extraction models thus fixed, the monkey was then tasked with learning to use these calibrated models to perform effective control of the prosthetic robot.

Progression of Neural Decoders

The initial cortical extraction model used in the experiment with monkey G was the one used for 4-DoF arm control in the series of experiments published by our laboratory in 2008 [121]. The first 4-DoF (linear and grip) control was established using this model in a center-out task. In early 2010, the cortical extraction system was rewritten to support multiple simultaneous decoding systems based on Optimal Linear Encoding (OLE). First, 4-DoF rotational and grip control alone were established with the new decoder. Next, the separate linear and rotational phases of the experiment allowed us to allow control of both types of movement without requiring simultaneous control over all DoF; the idea was to have the monkey complete a series of 3-DoF subtasks that he had already shown competence in performing, then linear and rotational control could "bleed" into the opposing task phases over time using the shared-mode control system (Section 5.5). Copies of this algorithm were run simultaneously, each operating on a subset of the experimental control dimensions (linear, rotational, and grip decoders ran in parallel) and in the separate task phases; each experimental trial essentially became a temporal progression of the two types of motor tasks already completed, along with the constant gripping subtask. As there was no real rotation target during the linear task phase, we could quickly give the monkey control over the rotational DoF during the linear phase of the experiment.

The ability to do some rotational control during linear movements contributed to the monkey learning simultaneous control with little penalty for control error. Next, some linear control was allowed into the rotational control phase of the experiment. The first 7-DoF control trials were performed using multiple 3-D decoders working in parallel. The 6-D decoder for linear and rotational robot DoF was then developed. We were very doubtful at first that linear regression with 6 regressors was going to be effective; initial evaluation of this regression model produced a lopsided PDs with better fits for rotational rather than linear DoF. However, it was extremely difficult to evaluate candidate neural extraction modules offline due to the closed-loop nature of brain-computer interface control. At the same time, 7-DoF control was working with the split model. Decoding errors and system problems would often cause the monkey to stop working, aborting sessions that could otherwise lead to successful control. To evaluate candidate cortical extraction models online but without the potential for ruining the experiment, we starting running candidate 6-D extraction models but not applying their output to the robot. Evaluating decoder performance post-hoc was even difficult because control skill can only be evaluated when the monkey is attempting to control the device; the interaction between monkey attention and skill was difficult to separate looking at trial data after it occurred. Instead, the quickest solution was to visualize and evaluate the output candidate 6-DoF decoding algorithms in real time using the Visualization module (Section 8.5.1). Development of the attention detection system (Section 8.3.6) may make it possible to evaluate control algorithms post-hoc or offline using recorded data.

After a few weeks of using the split decoders for robot control while visualizing the output of the unified model, we noticed that the 6-D model had converged on control signals that appeared to provide effective control. We shifted control from the split model to the unified 6-D decoding model in the middle of an experiment in which the monkey was effectively controlling the robot, and no decrease in performance of the task was seen at that time. From that point forward, the unified model was used. The fully-developed 6-D algorithm was used from the beginning of experimentation with Monkey F.

On the other hand, the 1-D grip controller was never unified into a single decoding model with the other DoF, even though the equivalent was done with the linear arm control experiment described in Velliste et al [121]. While it was almost certainly possible to integrate grip control into the more recent set of experiments, it was not done for two reasons, (1) the robot hand was much less reliable than the robot arm

that was used in the experiment. Keeping the decoding model separate allowed us to easily disable the grip control aspects of the experiment in real time and limit the experiment to the 6-DoF control problem. Results related to neural activity could be compared from day to day, whether or not the robotic hand was functional. (2) Grip control occurred in a stereotyped pattern at the end of each trial. Instead of sampling over a control space in which the monkey had to provide differential control signals depending on what target was presented, he was able to grasp the target in a single pattern at the end of each trial. During the 7-DoF control trials, this action occurred right before reward delivery, so that the neural substrate for calibration appeared to be related to the task sequence and reward expectation, rather than motor control signals. Before 7-DoF control was possible, when we were first training the monkey in the 4-DoF task, we had anticipated this problem. At that time, there was a sequence of releasing the gripper and moving the arm back towards the starting position before the monkey received the reward. However, the monkey seemed to understand that gripping was of primary importance, and never fully participated in moving the gripper back to the start position. It would anticipate the reward and lick the reward delivery tube during the gripping, opening, and retraction processes during observation. The retraction part of the task was eliminated, but the anticipation of reward behavior remained during gripping. Thus, the neural signals that would operate the gripper appeared to be derived from different modality of the neurons related more to reward than movement of the robot. It seemed inappropriate to put it with the other control signals that were related directly to differentiating intended motion directions of the arm. This is in contrast with the previous food-retrieval experiment [121], in which task reward was linked to food retrieval, well after movement of the gripper occurred.

Co-adaptation

Early in training monkey G to control the arm, the strict line between observation-based calibration and subject training using the calibrated decoder was blurred. A process of "co-adaptation" occurred where increasing amounts of partial control of the arm was given to the monkey as the calibration model continued to recalibrate itself. In theory, this would allow the calibrated model coefficients become more aligned with activity relating to intentional movement commands rather than passive responses to robot movement. In practice, this must be balanced against the moving-target aspect of changing the model against which the monkey had to learn to operate the robot.

Using significant amounts of co-adaptation with monkey G seemed to have the affect of annoying the monkey rather than improving its ability to control the device. Pure observation-based calibration was used for 6- and 7-DoF control with monkey G. For monkey F, a small amount of co-adaptive trials seemed to help the monkey learn to operate the device.

The cortical extraction system was developed by Sam Clanton, with OLE equation and code from Steve Chase and George Fraser.

8.3.3 Shared Control Arbitrator

Summary The control arbitrator integrated commands from an automatic control system and cortical decoding modules to form commands sent to the low-level robot controller.

Inputs Robot control commands from decoders, Robot control commands from automatic controller, Task-sequence dependent shared-control parameters from task controller, joystick override commands, attention level

Outputs Robot Control Commands

The control arbitrator implements the bimodal shared control algorithm described in Chapter 5 and collates control information from multiple sources. Output from the Arbitrator is sent to the robot controller as a time series of 7-D control commands.

The arbitrator serves three purposes in this experiment. These are

1. To provide a substrate for calibration of neural decoding systems.
2. To provide a universal mechanism for creating suitably challenging control problems that reinforce learning.
3. To integrate information from multiple cortical decoding modules into a single brain-controlled robot command stream.

If the extraction system did not require calibration, the subject was fully proficient in the use of the device, and the robot control commands came from a single source, the arbitrator would not be needed as a part of the brain control system. An exception to this would be a future use of the shared control algorithm for augmentation of BCI control signals that are of a lower bandwidth that what is required to

fully control a device, but that function was not used in the 7-DoF experiment.

The basic function and purpose of the arbitrator and the shared control algorithm that it implemented is discussed in Chapter 5. The description here augments that conceptual description with details of its implementation and integration with the 7-D experiment. As in that chapter, the terms c and c_p refer to individual entries into the $\boldsymbol{C_a}$ and $\boldsymbol{C}_p$ matrices corresponding to the shared control parameters for the robot control DoF under discussion.

The control arbitrator buffered all robot control commands passed over the RTMA network from cortical and artificial sources. This included cortical control input coming from multiple Extraction modules (Section 8.3.2). Mappings between decoder output DoF and the input DoF that were applied to control by the arbitrator were reconfigurable in real time. Each input was kept in a state vector representing the most recent updates to the desired control commands of each of the different sources. The responsibility of the arbitrator was to mix all of these control inputs into a single output upon which the robot was to act. The mixing action of the arbitrator was triggered by a 30 ms timing signal emitted by the cortical spike processing system. Upon receipt of the timing signal, the state vector was frozen and the mixing process was performed, ending with a control command being sent to the robot. Commands sent by the arbitrator were sent synchronously with the timing of spike recording, at the cost of introducing a potential delay into transmission of individual decoded commands to the robot.

Because of the plug-in architecture of the decoding system, during development there were commonly short periods in which no decoder was online to provide movement commands to the robot. The WAM controller used the last received command to command robot movement. If this command was set to move the robot into the monkey or into an extreme position where it could get stuck, the WAM controller would faithfully continue to execute this command. To remedy this, the arbitrator began self-generating control signals after about 60 ms, attenuating commands from the decoders at a decreasing level until they contributed zero-velocity input after about 10 samples. The arbitrator switched its timing back to that of the acquisition system as soon as that input came back online.

The mixing process occurred as a single evaluation of the BSC algorithm presented in Chapter (5) over the three separate domains of linear, rotational, and grip movements. The passive fixturing system attenuated commands based on the auto-

mated controller inputs for c_p entries < 1, then mixed those with the single optimal command for c_a entries < 1. If full control was indicated by the both parameters allowing full brain-control admittance, the output of the arbitrator was a copy of the collated brain-control input.

The actual arbitrator further modified the output command in three ways not indicated in Chapter (5). First, error checking was performed here on the input to make sure that huge or NaN values are not passed as robot control commands during development. Second, the arbitrator received input from the attention detector regarding the attentive state from the monkey (Section 8.3.6). The arbitrator could gate the output to the robot based on whether or not the monkey was intending to control the robot at a given time; this reinforced the connection between intention and movement in the monkey. Third, the arbitrator received control commands from a Playstation 3 controller (Sony Inc., New York, NY) that were used to override all other inputs. Handheld controller inputs were mapped to robot control DoF (Section 8.5.2) and were manipulated by the person running the experiment to disentangle the robot from objects in the experiment room (stuck with the fingers behind the cylinder, or with the arm wrapped around under one of the structural supports of the experimental frame) or to keep the robot from being controlled into these positions. Systems designed to automatically keep the robot out of these configurations were continuously improved so that eventually controller input was never needed, but similar manual intervention might be a potentially necessary feature of real-life BCI robot control systems in complicated scenarios that are difficult to fully encode into automated controllers (similar to manual assistive control systems such as that presented in Xu et al [132]).

Flexible Fixturing Variations

The flexible fixturing algorithm presented in Chapter 4 works well to guide robot movement toward single enclosed manifolds of points representing successful behavior. As the FF algorithm is intended for use moving forward in the more sophisticated problem of dextrous hand shaping, the algorithm will work less well when point sets or manifolds are discontinuous. Using the current system, the arbitrator will attenuate command portions not positively spanned by the complete range of exemplar commands. Given a gap in the solution manifold between these points, the arbitrator will incorrectly interpret commands as correct that are pointed between them. We

intend to use a more sophisticated type of FF algorithm that can handle complex discontinuous manifolds. Two variations on the FF algorithm are briefly mentioned here as our group moves towards working on the control of dextrous grasping. The first one is just sufficient for handling discontinuities and has a simple implementation, while the second is more sophisticated. The difference between the following FF implementations lie in how the $\boldsymbol{D}$ matrix is interpreted.

Multiple-Point Passive Impedance

Instead of projecting the command vector $\boldsymbol{u}$ onto the positive span of the complete matrix $\boldsymbol{D}$, the MaxPosSpanBasis function (Chapter 4, Algorithm 3) is simply run a single time, with no recursion. This implementation then uses the closest artificial command to the command issued by the subject as the single input. This version of the control arbitrator is designed to make no assumptions about the nature of the artificial command input in terms of its shape between points represented or continuity. Instead of assuming continuity between represented points, it "snaps" the $\mathbf{D}$ matrix to the command most closely aligned with the command input from the subject.

Future Work - Discontinuous Manifolds

A complete solution to this problem will involve the snapping of the multiple-point algorithm and the projection to the positive span of the manifold algorithm. Vectors defining each individual manifold will be input into the modified MaxPosSpanBasis algorithm. This multiple-point algorithm will be run over all columns of a set of $\boldsymbol{D}$ matrices. The algorithm (below) simply selects the closest manifold for projection of the input. A parameter may need to be added to stop the system from rapidly switching between projection manifolds. It could take into account a time history of the manifold used and switch only when the closest manifold changes over a longer time period.

Algorithm 4 Modified maximum positive span basis algorithm for discontinuous manifolds.

function MAXPOSSPANBASISMULTI($\boldsymbol{u}$, $\boldsymbol{D}_1$, $\boldsymbol{D}_2$, ...)
  $\boldsymbol{D}_a \leftarrow$ columns of all $\boldsymbol{D}$ matrices concatenated.
  if $\boldsymbol{D}_a$ is empty **then**
    return $\varnothing$
  end if
  $\boldsymbol{P} \leftarrow \boldsymbol{D}_a^T \boldsymbol{u}$
  $r, index_r \leftarrow max(\boldsymbol{P})$
  if $r > 0$ **then**
    $j \leftarrow \boldsymbol{D}$ matrix index corresponding to $index_r$.
    $q, index_q \leftarrow r$, $\boldsymbol{D}_j$ column corresponding to $index_r$
    $\boldsymbol{D}_{j,q}, \boldsymbol{D}_{j,-q} \leftarrow$ column $index_q$ in $\boldsymbol{D}_j$, remaining cols in $\boldsymbol{D}_j$
    $\boldsymbol{u}_{-q} \leftarrow \boldsymbol{u} - q\,\boldsymbol{D}_{j,q}$
    return append($\boldsymbol{D}_{j,q}$, $\boldsymbol{MaxPosSpanBasisMulti}(\boldsymbol{u}_{-q}, \boldsymbol{D}_{\boldsymbol{j},\boldsymbol{-q}})$)
  else
    return $\varnothing$
  end if
end function

The command arbitration system was written by Sam Clanton.

8.3.4 Autonomous Controller

Summary The autonomous controller provided automated commands to the arbitrator that were used to modify the commands from the cortical decoding system. At every time step, it generated a set of commands that would lead to progress through the task. In this experiment, it also detected when interaction events occurred with the robot that would lead to either task progression or failure.

Inputs Robot State, Task State, Grasp Poses from Planner

Outputs Robot Control Commands

The cortical decoder and subject training mechanisms in the experiment relied upon the existence at each time step of a set of exemplar or desired commands that would drive the robot closer to completion of the reaching and grasping task. The autonomous control system continuously provided this set of exemplar commands to the system. The autonomous controller reacted to much the same information that

the monkey subject had in order to calculate desired commands; it was commonly called the "Silicon Monkey" in the experiment. The set of commands that it produced were identical to those that would originate in decoding modules measuring control intent from monkeys that were each trying to complete the grasping control problem in a different way. As the monkey saw a cylindrical object to be grasped and not a single point in space to which he was to guide the arm, the range of possible grasps of this object was similarly accounted for by the Silicon Monkey.

The autonomous control system was cued by the task phase emitted by the Task Controller (Section 8.4.1) to guide the robot towards completion of subtasks that led toward progression through the complete reaching, orienting, and grasping task. Task phases (discussed in Chapter 5) implemented the procedural order of each experimental trial - for instance, in the control task of reaching and orienting towards an object, then grasping, task phases provided a mechanism by which the control planning system knew to approximate the hand to the target object before closing the fingers.

Another input to the autonomous controller was a set of robot poses from the prototype Grasp Planner (Section 8.3.5). These poses, in the position space of the controlled DoF of the robot, represented the desired range of grasp poses that would complete the task. The autonomous controller used these poses in conjunction with the current robot state to form the vectors that were passed to the Grasp Arbitrator to form the $\boldsymbol{D}$ matrix. It also received the target location directly from the task controller; formation of a control vector pointed to the target represented the command used as the optimal command for input into the active shared control system in the arbitrator, and the only command used for the $\boldsymbol{D}$ matrix in the case that Grasp Planner input was not available; the reversion to the simpler system was a reliable fallback mode in case of failure of the planning system during its development.

Commands issued by the control planner were in the same set of control DoF in which the cortical decoding system delivered commands. Note that while this subsystem could perform full trajectory planning, there was not any expectation that the robot would obey any such trajectory; the planning system provided an appropriate command at each time step given the actual, unpredictable state of the robot. This ensured that planner commands were consistent even if the actual control of the robot was the result of large amounts of brain control information, or in the presence of a large degree of robot interaction with objects in the environment. In later versions of the Silicon Monkey, it was aware that the target region was not always

directly accessible from the current robot state. If the hand was behind the planner, the module cut off the commands based on the planning system and substituted a single column $\boldsymbol{D}$ that directed the hand to move around the back of the doorknob and retract towards a vertical plane in front of the target object.

Grasp event detection

A secondary function of the Silicon Monkey in this experiment was the detection of subtask completion or task failure based on robot position feedback. While progression through task states was governed by the Experimental Control Module (Section 8.4.1), it relied on event messages sent by the Silicon Monkey to interpret when subtasks were being completed. This was communicated by specific "Grasp Events" sent by the Autonomous Controller for specific combinations of task and robot state. For instance, if the system was in the Grasp task state and the autonomous controller found itself at the target with the fingers closed, it would emit a message indicating that the target was gripped and the task controller would use this information to advance the task state. Integration of event detection with autonomous command generation was aimed toward the autonomous controller system containing all knowledge of the specific robotic task embedded in a single module. Changing the experimental task could then be performed by substituting the appropriate autonomous controller.

Autonomous Controller State Machine

The Silicon Monkey was implemented as a state machine that composed robot commands based on the task phase of the experiment, at the same time checking for satisfaction of conditions required for producing grasp events. The pseudocode for the module was as follows:

Algorithm 5 Pseudocode for Operation of the Silicon Monkey

```
BASIC CONTROLLER()
  while EXPERIMENTRUNNING
  do
     Task_State ← RECEIVETASKSTATE
     Robot_Position ← RECEIVEROBOTSTATE
     CHECKGRASPSTATE(Task_State, Robot_State)

     switch Task_State
       case Reaching :
            if TARGETOCCLUDED
              then BACKUP
              else GOTOCENTER
       case Homing :
            if TARGETOCCLUDED
              then BACKUP
              else ROTATETOTARGET
       case PreGrasp :
            if TARGETOCCLUDED
              then BACKUP
              else GOTOTARGET
       case Grasp :
            if TARGETOCCLUDED
              then BACKUP
              else GRASPTARGET
       case Release :
            if TARGETOCCLUDED
              then BACKUP
              else GOTOTARGET
       case default :
            GOTOSTARTPOSITION

CHECKGRASPSTATE(Task_State, Robot_State)
  switch Task_State
    case Reaching :
         if NEARCENTRALPOINT
           then SENDGRASPEVENT(Near_Target)

    case Homing :
         if NEARTARGET
           then SENDGRASPEVENT(At_Target)
         if AWAYFROMTARGETORCENTER
           then SENDGRASPEVENT(No_Longer_Near_Target)

    case Pregrasp :
         if ATTARGET
           then SENDGRASPEVENT(Pregripped)
         if AWAYFROMTARGET
           then SENDGRASPEVENT(No_Longer_Near_Target)

    case Grasp :
         if GRIPPED
           then SENDGRASPEVENT(Gripped)
         if AWAYFROMTARGET
           then SENDGRASPEVENT(No_Longer_Near_Target)

    case Release :
         if AWAYFROMTARGET and HANDOPEN
           then SENDGRASPEVENT(Safehand)

    case default :
         if ATSTARTLOCATION
           then SENDGRASPEVENT(At_Start_Position)
```

Command Generation from Current and Target States

The autonomous controller module generated velocity commands based on current and target points in the control space of the robot using a simple proportional-integral-derivative (PID) controller. This controller operated in each linear and grip DoF as:

$$e(t) = \hat{x}_j(t) - x_j(t)$$
$$d_j(t) = K_j^p(e(t) + K_j^d(e(t) - e(t-1)) + K_j^i \sum_{\tau=0}^{t} e(\tau)$$

where $\hat{x}_j(t)$ is the target pose of the robot in DoF j at time t, and x_j is the corresponding current position, and K^p, K^d, and K^i are proportional, derivative, and integral gain parameters implementing the PID controller. d_j is the single-dimension command output from the automated control system. For the rotational DoF, $e(t)$ was found by finding the 3x3 matrix corresponding to rotation from current to target orientation, then applying Rodrigues' formula to find the equivalent rotational velocity axis and scale. PID parameters K_p, K_i, K_d were tuned manually so that the robot under automatic control would achieve targets quickly with minimal overshoot.

The autonomous control system was written by Sam Clanton.

8.3.5 Grasp Planner

Summary The Grasp Planner Module generated a set of robot poses that would complete the grasping task. It provided this set of poses to the autonomous controller to build the ***D*** matrix supplied to the Arbitrator system.

Inputs Target Shape, Position, Orientation

Outputs Robot Poses

The grasp planning module is a prototype of more advanced planning systems that could incorporate complex grasping and manipulation behaviors into the brain-computer interface experiment. Given the shape, location, and orientation of the target and the robotic effector communicated by the Experimental Control module, the grasp planner creates a set of robot poses that would achieve object grasp. In this trivial version of the planner, the target poses are based on random point sampling within a spherical range of positions around the central target point supplied by the Experimental Control Module (Section 8.4.1). The rotational dimensions of the poses are similarly generated in a region surrounding the target orientation in the space of yaw-pitch-roll.

The planner and autonomous controller are run as separate modules to facilitate the time-consuming nature of grasp planning. For a non-trivial planner, the idea is for it to receive a target position/orientation/shape at the beginning of a trial. The autonomous controller operates under its fallback/single point pursuit mechanism until the planner generates a cloud of points representing target poses that would each successfully grasp the object. When the autonomous controller receives that point cloud, it begins to generate sets of control commands based on the poses sent by the planner. The autonomous controller operates in the experimental timebase, while the planner runs separately, providing input to the system asynchronously. The planner that is intended to replace the trivial planning module is GraspIt!, a grasp simulation and planning system developed at Columbia University [18]. GraspIt! uses a process of simulated annealing to find grasp configurations that provide form closure of an object given a solid model of the object and robot hand. Those poses make up a target cloud suitable for use for the autonomous control portion of the BCI system. A prototype system that has incorporated poses generated by the GraspIt planner into this experiment has been developed and evaluated offline by Rob Rasmussen in our laboratory.

The trivial grasp planner was written by Sam Clanton.

8.3.6 Attention Detector

Summary The attention detector module attempted to determine if the monkey was attempting to control the device based on cortical spike rate data

Inputs Spike rates

Outputs Attention level

An auxiliary module in the control system pathway is the attention / lick detection module. The creation of this module was in response to the presence of high amounts of activity recorded by arrays present due to arm and head movements of the monkey during calibration in early brain control sessions. While this activity was omitted from decoding model calibration by manual rejection of data recorded during movement, we wanted to find a way to automate this process.

More generally, an important aspect of clinical BCI prosthetic control will be knowing when the subject is intending to control the device. Neurons that are har-

nessed for control are simultaneously involved in other normal cortical processes. As there is no physiologic gating mechanism to indicate when observed activity is to be applied to robot control or to be ignored, we have integrated this function into the brain control experiment.

The firing rates of the recorded units tended towards a certain mode or range during observation and control of the robot. During reward segments or when the monkey was moving his head or arms, however, the firing of units in the arrays generally occupied another mode or range of rates. These rates were on average much higher than those observed during brain control; periods of head and arm movement were visually identifiable looking spike rasters in real time. Periods of time in which the monkey was looking away from the experiment or was not intending to control the device could not be identified visually in the raster. A series of classifiers have been attempted to detect both of these types of non-brain control behaviors. These classifiers are trained on manually labeled samples of spike rates across all units ("attentive" vs. "non-attentive" vs. "moving") taken during observation periods. An initial version of the attention detector used a multiclass support vector machine to discriminate incoming samples, with moderate success. A more recent version of the detector, written by Meel Velliste relies on clustering within the first two principal component analysis vector projections and projection of incoming spike rates onto either one cluster or the other.

After the attention model was calibrated, the detector began to emit Attention messages that contained either a binary value indicating whether or not the monkey was paying attention, or a continuous value indicating its level of attention. The arbitrator could use this to gate outgoing robot control commands.

The original attention detector was written by Sam Clanton and Zohny Zohny, replaced by a current version authored by Meel Velliste.

8.3.7 Low-Level Robot Control

Summary The Low-Level Robot Control system translated commands in the brain-controlled DoF to torque commands at the joints of the robot. Along with the control of robot kinematics, the robot implemented an impedance control system that governed force and torque interactions between the robot and its environment.

Inputs Endpoint velocity commands

Outputs Robot endpoint position and velocity

The low-level robot control system interpreted commands from the high-level control system and translated them into joint torques of an arm robot. The filter through which this transformation took place is the impedance control algorithm discussed in Chapter 6. Velocity commands to the robot endpoint acted to update the kinematic setpoint from which the actual motion of the robot is compared during evaluation of the low-level impedance algorithm.

The low-level robot control system was implemented as a single multithreaded program. The thread subprograms were organized in a hierarchical fashion spanning the receipt of high-level kinematic commands from the Control Arbitrator at the 33 Hz timebase of the experiment to the issuance of torque commands to the robot at the 500 Hz timebase of the inner robot control thread. The 500 Hz inner control loop was implemented using the Linux Real-Time Application Interface (RTAI), which are a set of Linux extensions that allow segments of code within programs to be run in real time with a very high degree of reliability. The RTAI subsystem accomplishes this by running code segments that are declared as real-time operations first, while the entire Linux kernel and OS is run in the spare time of any real-time code. Each iteration of the inner 500 Hz control loop was initiated by a timer that guaranteed reliable execution at the time resolution of the processor.

The responsibilities and timebase of each thread are organized as follows:

Inner Control Thread Computed an impedance model taking into account target velocity and encoder readings supplied from the shared-control arbitrator. Used resulting desired Cartesian forces and torques as input to recursive Newton-Euler (RNE) inverse dynamics model to compute desired motor torques. Implemented simple joint-space control model to control the redundant DoF to keep the arm in relatively biomimetic reaching poses. Implemented safety system keeping robot within certain force, velocity, and position limits. Was implemented to run in a 500 Hz hard real-time loop using RTAI.

Network Thread Received target velocities asynchronously for input to inner control thread. Communicated with rest of BCI system regarding robot kinematic and dynamic state. Triggered synchronization pulses fed back into neural recording system (see Section 8.5.4). The thread was timed on receipt of control

packet, presumably at 33 Hz timebase of whole system.

Hand Thread Operated Barrett Hand using direct velocity commands in a soft real-time control loop operating at 50Hz.

Keyboard/Interactive Thread Operated diagnostic display, interpreted manual control keystrokes, and logged system status at a 25Hz update rate.

Inner Control Thread

At the beginning of execution of each loop iteration, the controller read the encoder values from the robot and calculated the joint angles. From these values, the forward kinematics of the WAM were computed. The endpoint velocity of the WAM was estimated by filtering the position changes using an exponential filter with a 2% decay/sample applied to a circular buffer holding 30 samples (60 ms). End-effector rotational velocity was derived by forming a difference rotation $\mathbf{R}_D$ between two successive orientation measurement rotation matrices $\mathbf{R}_A$ and $\mathbf{R}_B$, where $\mathbf{R}_D = \mathbf{R}_A^{-1} * \mathbf{R}_B$, followed by conversion of $\mathbf{R}_D$ to a vector at the axis of rotation using Rodrigues' formula. Rotational velocity measurements were found to be relatively noisy, and are filtered by a 750 ms filter with a decay rate of 0.992. Linear and rotational velocities were also determined using an endpoint Jacobian given the robot pose and measured joint velocities in an attempt to reduce the rotational velocity noise, but these were very similar to those derived from the direct differentials.

The recursive Newton-Euler formulation of the robot dynamics was calculated with a virtual link at the WAM endpoint effecting the desired impedance and kinematic state of the robot (see Chapter 6). The actual model that was used for real-time computation did not incorporate sampled velocity and acceleration terms, but the impact of velocity and non-gravity acceleration terms are relatively minor compared to the torque required to counteract gravity given the mass of the WAM robot and the speed at which it moved during control. We have been able to simulate the WAM operating with these parameters, but have had trouble with maintaining stability when a more dynamic model was used.

The desired kinematic state was based on the last received update to the target velocities from the arbitrator system, which is processed by the RNE kinematic-impedance control algorithm. The output of this calculation is a set of joint torques that are commanded to the WAM motor controllers at each joint. Each Barrett motor

controller (puck) implements a very tight current control algorithm running at 30 kHz which produces the commanded torque at the motors calculated from the model joint torques.

The impedance-based robot control algorithm was implemented for this experiment because (a) Monkey-derived robot control commands are potentially very random; there must be no expectation of smooth trajectories being commanded to the robot, and (b) interaction with solid objects was desired in this experiment so that actual movements of the robot were really a product of the movement commands that are supplied to it as well as contact forces. While it would be possible to create a model to reconcile the potentially chaotic input and interaction with objects in the environment in a completely kinematic control scheme (e.g. for admittance-based robot controllers), a full model of the environment and complete velocity command trajectories would have to be known beforehand; this model had to deal with these inputs in real time and without a full model of the robot's surroundings.

The inner control thread also superimposed a number of other control schemes on top of the impedance controller. Each of these separate controllers computed desired forces or torques into the Cartesian or joint torque goals, and were simply added to each before evaluation in the RNE equations (Cartesian force/torque) or sending to the robot (joint torque), which had the effect of the robot attempting to achieve all of the sub-task goals simultaneously.

The additional subcontrollers were:

Elbow Swing Controller This subcontroller implemented a simple and weak control loop over the angle of arm elbow swing. It attempted to keep the shoulder from abducting past 90 degrees or adducting more than 10 degrees from vertical towards the monkey. Additionally, it attempted to achieve a shoulder angle rotated to approximately 90 degrees from the angle of any commanded endpoint rotation of sufficient magnitude. The joint angle constraints helped to influence robot poses to look roughly physiologic. Moving the elbow joint in reaction to the commanded rotation angle facilitated the development of a consistent proximal arm pose over larger rotations in the wrist joints; the vertical angle of the forearm dictated the ease of effecting different Cartesian rotations because of the specific kinematics of the WAM robot in the workspace over which the robot was operating. Without any sort of calculation of inverse kinematics, this helped the robot to make specific rotations more easily in some situations.

Configuration Awkwardness Reducer Because the Cartesian controller did not calculate inverse kinematics or plan robot trajectories in the joint space of the WAM, it was possible for the WAM to get into awkward poses from which it was difficult to achieve Cartesian movement of the hand over the robot workspace. Soft joint limits were set up for each joint so that movement past these limits was resisted but not impossible to achieve. This was implemented as a simple proportional controller with some boundary force offset and hysteresis so that movement outside the joint range triggered resistance forces that were felt fairly strongly right at the limit boundary. A set of these types of controllers at each joint in the wrist were very effective in keeping the WAM robot out of awkward joint configurations, even over control sessions that lasted up to three hours.

Workspace Limit While our original goal was to let the robot completely interact with the environment in a freeform manner, there were a number of incidents with the robot getting tangled into the experimental frame, threatening to damage its own wires, or getting caught up into the monkey chair and head shield. Monkey G was observed grasping the robot hand when it came near it, and even exhibited what appeared to be grooming behaviors with the hand in two instances. Usually, the monkey was annoyed by the robot getting too close during periods that it was not attempting to control it. We implemented a workspace limit that was actually slightly beyond the boundary of the open workspace; the robot could still interact with the robot frame and monkey chair (and monkey), it was just prevented from moving very far into these regions. These limits were achieved by setting a rectangular workspace, then implementing a simple spring-damper model system that pushed the robot back into the workspace if it wandered outside of it.

8.4 Experimental Control System

The experimental control system provided the accessory systems necessary to conduct the BCI system through experimental trials and sessions in the experiment. The backbone of the experimental control system is the Experimental Control Module, which timed and sequenced the series of tasks that comprise a trial. The other parts of the subsystem were the different hardware and software pieces that implemented the behavioral parts of each trial not directly related to brain control.

8.4.1 Experimental Control Module

Summary The experimental control module placed the robot control pathway into the context of a specific experiment. It controlled the sequence of robot movement events that comprised a successful trial and paced the experiment through those events. It dictated the specific grasp target presented to the subject on each trial, controlled the shared-mode control parameters applied during trials, the reward system, and which data that was supplied to the cortical decoding system for calibration.

Inputs Grasp event notifications, Monkey button press events, Presentation robot status notifications, Controller button press events

Outputs Task states, Shared control parameters, Target locations, Decoder collection settings, Presentation commands, Reward activations, Audio cues

The experimental control module (XM) conducted the experiment by cycling through a set of task states that defined individual trials, as indicated in Figure 5.4. Task states represented conceptual phases in the procedure of a trial or motion primitives in the case of the reach and grasp movement itself. State transitions were governed by signals from other modules that indicated the occurrence of events in the experiment.

Task State	**Forward Transition**	**Failure**
Intertrial	Timeout	None
ButtonPress	ButtonPressed	Timeout
Presentation	PresentationFinished	None
HandOpening	*HandOpen*	Timeout
Reaching	*NearCenter*	Timeout
Orienting	*Oriented*	Timeout / *LeftRegion*
PreGrasp	*At Target*	Timeout / *LeftRegion*
Grasping	*TargetGrasped*	Timeout / *LeftRegion*
Reward	Timeout	None

Table 8.1: Events governing Experimental Control Module forward and failure state transitions. Grasp events are italicized.

The grasp planning subsystem, which had the domain knowledge of how to complete reaching and grasping tasks, informed the XM that in-trial subtasks were completed by sending "grasp events", indicated in italics in Table 8.1. At the beginning of each task state, the XM sent out a configuration message encapsulating the shared

control parameters, flags instructing extraction modules to collect data for calibration, target locations, and other session-dependent configuration parameters. It then executed any state-dependent actions needed such as issuance of sound cues, commands for movement of the presentation robot, or commands to the analog module (Section 8.4.2 to issue a reward to the monkey. Next, it waited for the events from Table 8.1 to trigger state transitions.

Session Execution Sequence

The experimental control began at Repetition 1, Trial 1. Repetitions represented the successful completion of the task with a complete set of presentation targets and robot start positions. These governed the progression of the session from the model calibration to subject practice phases of the experiment. By forcing decoder calibrations and experiment phase (observation, coadaptation, control) transitions to repetition boundaries, spike and movement data entering model calibration is based on evenly sampled distributions of target directions in successful trials; failed trials were rejected from counting towards repetition completion.

A set of configuration files are loaded at the beginning of each trial that contain a set of data comprising the configuration of the system. This configuration data included which task states were conducted, the shared-mode control parameters to be applied during each task state, set of targets and start positions, reward settings, and other parameters. As these are loaded each trial, the system was highly reconfigurable during the running experiment.

Next, a pair of target and robot start-point configurations were selected from those remaining in the current repetition and sent to the automatic controller. The XM then entered the ButtonPress state and emitted an audible start cue to the monkey. It then waited for receipt of a ButtonPress message from the analog module. When a ButtonPress signal was received, the experimental control system transmitted a movement command to the presentation robot control module and moved into the Presentation state. The reaction of the Automatic Controller was to move the robot to the start position. Shared-mode control parameters issued during this period were always set to full automatic control to eliminate collisions between the two moving robots. When the presentation and WAM robots were in their target and starting positions, the Presentation robot controller notified the experimental controller and the task state progressed to Reaching. During the actual reaching trial, progression

through the task states was now governed by receipt of grasp events from the autonomous controller. If and when the robot grasped the target cylinder, the XM entered a Reward state in which it issued a reward command to the Analog module and waited for the robot to retract from the target. The controller then proceeded to the next trial, incrementing the rep/trial counter as needed.

During the task execution periods of experimental trials, a timeout was configured for each task state. If the timer expired for a task state before completion, or in certain cases if the robot was controlled far away from the subtask goal corresponding to the task state, e.g. if the robot moves far away from the target when in the Grasp task state, the system enters a failure mode. During task failure, the experimental controller emitted a "buzzer" sound, put the starting and target positions of the current trial back on the remaining targets for the repetition, and started the next trial. The controller informed any decoder modules of the failure, at which time they discarded any data buffered from the trial for calibration.

Alternative success and failure modes were indicated by the experimenter using a joypad interface. If the monkey stopped paying attention to robot movement during the observation and calibration phase, or the monkey appeared to stop paying attention and control became off-track during the training phase, the experimenter could press a button to abort the trial. Alternatively, the experimenter could also advance the system through control subtasks manually; the primitive autonomous control system used in this experiment could only incompletely recognize when the subject had correctly controlled the robot to grasp an object or complete any of the grasp subtasks. The ability to manually trigger progression through the task allowed it to be completed based on commonsense notions of the performance of motion primitives. Some of the earlier training process with the monkeys often involved manual reward and punishment issued through this system. As monkeys became acclimated to the task, the manual system was relied upon to a lesser degree.

The experimental control module was written by Meel Velliste and Sam Clanton.

8.4.2 Analog Module

Summary The analog module performed analog I/O supporting external devices such as the button and the reward delivery system.

Inputs Reward Commands

Outputs Button Press Events

The analog module controlled interaction between the BCI experimental system and a few hardware systems involved in running the experiment. This module worked as an interface between a National Instruments data acquisition card and the RTMA system. It waited for events on monitored analog channels to occur which triggered signals culminating in RTMA messages, and also for certain RTMA messages that are converted to voltage outputs.

In the final experiment, these functions were limited to detection of monkey button push events and issuance of reward. The monkey button was a simple pushbutton mounted on a metal frame. Button pushes completed a simple circuit, and the analog module triggered button pushed events when triggered. This module also sent an output to a reward controller. The reward controller was a custom-made electronic component that controlled a solenoid valve on a water tank connected to the reward straw on the monkey chair.

It also could deliver a manual reward of preprogrammed length when a reward button was pressed by the experimenter. The analog module itself was preprogrammed with a number of reward patterns designed to promote motivation without delivering large amounts of water. These patterns gave out small rewards punctuated by larger ones in random intervals, or split single large rewards into many smaller ones (this was patterned after reward research in operant conditioning). Because of the requirement for monkey patience over relatively long experimental trials, it was difficult to get the monkey to work for anything but large rewards. Early in training, these alternative reward systems facilitated the monkeys working over long periods of time. After the monkeys were used to completing long trials for reward, these schemes were no longer needed.

The analog module was written by Meel Velliste, Sam Clanton, and Andrew Whitford.

8.4.3 Presentation Robot Controller

Summary The presentation robot controller moved the presentation robot into target poses.

Inputs Target sequence number for presentation

Outputs Completion of movement notifications

The presentation robot controller received commands from the XM to move the presentation robot to different positions and orientations in the WAM workspace. The presentation robot was a DENSO VS-6556G industrial-style 6-axis admittance robot that was designed for speed and accuracy, but was completely noncompliant.

Rather than sending actual pose commands to the presentation robot directly, a set of canned poses were determined for the robot to achieve during brain-control trials. First, a neutral no-target pose was determined for the robot to move to between trials. Next, robot poses were created at 40 degree orientations from a forward-facing WAM hand around a central point in the workspace. These specific locations of these targets were such that the WAM could place the hand at that central point and orient towards each target with the WAM fingers not interfering with the rotation by contacting the object.

While there was limited concern over the potential of the WAM robot to cause damage to the experimental setup or injury to the monkey, the DENSO robot would be capable of breaking many of the electrical components of the experiment and pinning the monkey in its chair. The canned poses added a layer of safety that limited the possible range of positions and transitions between them that were known to be safe. While the WAM arm was compliant, there was concern that the DENSO robot could break the Barrett Hand fingers when closed (this occurred during testing) or pin the robot against the frame and damage it. The XM relied on the WAM to self-report that it was clear of the DENSO before any movement command was issued. Then when the XM issued a presentation robot movement command, it interrupted all other actions to wait for a message from the presentation robot controller that it had completed the movement. This small amount of sequencing embedded in the movement of both robots prevented collisions caused by the noncompliant robot moving. The compliant arm robot, however, frequently collided with the stationary DENSO robot during experimental reaching trials.

The presentation robot control module was written by Meel Velliste, with modifications and pose registrations by Sam Clanton.

8.5 Manual Intervention System

Manual intervention in the BCI experiment would not ultimately be needed in a fully functional BCI. In this experiment, human interaction in tuning the decoding modules and influencing the behavior of the monkeys was critical to the success of the project. These modules fulfill those functions.

8.5.1 Control Visualization Module

Summary The Control Visualization Module presented the control-space commands that were being output from the various cortical decoders and automated control system for comparison in real time.

Inputs Movement Commands, Robot configuration

Outputs None

The control visualization module allowed the direct output of different cortical extraction and control algorithms to be visualized in real time. The module, implemented in Python using the Panda3D game engine, displayed a 3D scene with one or more objects corresponding to independent controllers specified in a configuration file (Figure 8.17). Each object indicated a position, orientation (usually linked to the robot) and linear and rotational velocities output by the corresponding decoder. Before the 6-DoF unified decoding model was developed, a number of different decoding methods were pursued. Different models for the control of rotation were being built and many different sets of parameters were being tried during evaluation of these different decoders. Until the development of the Visualization Module, decoding modules were evaluated offline for how well each motor model was able to fit or predict the cortical decoder training data, but this had only a loose correspondence to how well each model would perform during closed-loop control.

The Visualization module was a way to approach this without potentially losing a session's worth of data because of monkeys losing motivation due to poor control. The idea of the visualization system was that many decoders could be run at once, and the output displayed on a screen. Humans running the experiment could then evaluate the quality of the control given this visual output. Attempts to automate this process were confounded by the monkeys not always attending to the task; it was important to only evaluate control during periods in which the monkey was

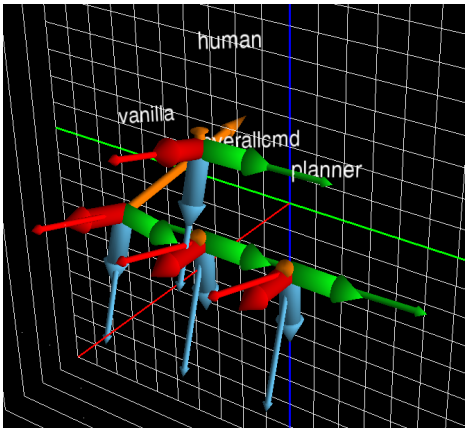


Figure 8.17: Screen capture from control visualization module.

attempting to control the device.

The Visualization Module was written by Charles Frantz, Sam Clanton, and Zohny Zohny.

8.5.2 Manual Input Module

Summary The Manual Input module provided a system in which human experimenters could manually provide positive or negative reinforcement to the monkey during running BCI experiments. In addition, game controllers were used to label data for the Attention Detector module and to directly control the robot when stuck.

Inputs None

Outputs Cancel/Advance signals destined for the XM, Data labels for the Attention Detector, Robot control override commands

The controller interface module provided an interface between one or more handheld game controllers (Sony Sixaxis Playstation 3 controller) and the brain-computer interface experiment. Controllers were used mainly during brain-control trials to manually cancel or advance subject progress. If the monkey was not paying attention to control or performing poorly for reasons not related to control skill, the experimenter would often abort the trial. The monkey was determined to not be paying attention when it looked away from the robot or started to rotate or move to a significant degree in its chair. This would give the monkey immediate negative feedback for starting a trial

that it did not intend to finish, or for giving up on a trial. Conversely, if the monkey was very close to completing a subtask, or completed the grasping task with the robot but it did not register through the Autonomous Controller module, the experimenter could register successful completion of the subtask externally.

The Game Controller module was also used to label incoming data for use in the Attention Detector module, and used in some cases to directly control the robot. Manual robot control was used occasionally in earlier experimental sessions to slightly adjust robot position if its pose approached a conformation that was difficult to proceed from to get to a goal state. Usually this occurred if the prosthetic and presentation robots became interlocked, or if the prosthetic robot became twisted into a conformation that is difficult to proceed directly from to a goal state. Messages from the Game Controller module were also used to take the place of cortical decoder commands, allowing control to be performed by a human operator directly. In this case, the robot control commands are sent to the system with the same 30 ms timing as the cortical commands. This was used in the human subject experiments discussed in Chapter 4.

The Game Controller Module was written by Sam Clanton.

8.5.3 Drift Tracking and Correction Modules

Summary The drift tracking and correction modules allowed the experimenter to compensate for apparent drift or bias in the decoded neural control commands.

Inputs Control commands from decoding modules

Outputs Corrective commands to the decoder

The drift tracking and correction modules were written to correct for bias observed in the output of the cortical decoding modules. Especially in the early BCI trials with Monkey G, after calibration the robot would usually be observed to drift in different directions in each session. During periods in which the monkey appeared to be attempting to control the device, the control signals being sent to the arbitrator seemed to oscillate around a baseline value shifted away from zero velocity, so that it was much easier to control the robot in some directions rather than others. The drift tracking module tracked the mean control commands emitted by particular cortical decoding modules over 20-30 second periods and displayed it to the experimenter.

The drift correction module allowed the addition of a reverse bias to the output of individual decoding modules. Manual drift correction was applied very gradually when needed. Much of the drift was removed from the control signals when low-movement samples started to be removed from data going into the Decoder calibration procedure, and this pair of modules were used less over time.

The drift tracking mechanism was written by Sam Clanton. The drift correction module was written by Meel Velliste.

8.5.4 Experimental Timebase

The dedicated DSPs on the RZ-2 systems provided a convenient mechanism to set the timebase of the overall experiment. UDP packets sent by the RZ2 systems at the end of each sampling cycle were sent at 10 ms intervals to an accuracy of less than 1 microsecond. The timebase of the experiment was controlled by the receipt of sets of 3 sequential packets by the spike processing module and subsequent issuance of spike count messages over the RTMA network. Issuance of spike count messages initiated a cascade of messages between control modules that culminated in a velocity command being sent to the low-level robot control system. These intermediate control modules were not self-timed, but form a part of a branched pipeline. The robot control module itself operated in an RTAI (real-time linux) framework that calculated motor torques at 500 Hz, but accepted velocity commands on an as-available basis. Thus, the system architecture was laid out as a data processing pipeline bookended by two independent real-time systems. The cortical decoding and other control modules then relied on the timebase provided by the TDT system to dictate the sampling and interval times of the data that each module operates on, rather than relying on the internal timing of any intermediate module itself. Because of the independent timing of the robot controller, the internal timing of intermediate steps is less important than would be the case if the entire system operated synchronously.

System delay, the time elapsed between a spike being recorded recorded and observing its effect on control of the robot, was monitored by synchronization pulse signals periodically generated by the TDT and attached to UDP packets streaming from the RZ-2 system. Flags indicating receipt of the sync pulse were transmitted within spike count packets in the RTMA network, and were similarly set within any packets derived from the spike packet through the brain control pathway. At the

robot control computer, the low-level controller pulsed a line on the serial port when a sample with the sync flag set was received. The serial line was connected back to inputs in the RZ-2 systems, closing the loop. By knowing the round-trip time for spike count packets that coincide with sync pulses, the overall system delay can be estimated. The total round-trip time was 10 milliseconds before Arbitrator module began to time the release of control commands on the receipt of spike count messages, which could add an additional $\approx$ 30 ms to the round-trip time. Data within spike count packets was at most 30 ms old at the time they were sent out, so that there was a total delay of slightly less than 70 ms in the spike data reaching the robot controller.

The timing system was developed by Meel Velliste.

Chapter 9

Conclusions

To conclude this dissertation, this final chapter contains a discussion of some of the important parts of what has and what has not been accomplished in the work presented here. It starts with what has not been accomplished, which is a deep understanding of the neuroscience behind how two monkeys learned to operate a 7-DoF arm with a BCI. It is followed by a statement about seeking simplicity within complex systems. The chapter ends with a discussion of what the successful conclusion of our experiment means for BCI experimentation in the future.

9.1 Real-World BCI and Neuroscience

Something that is missing from this dissertation is a detailed analysis of what exactly occurred in the brain of the monkeys in the process of developing BCI control. Does the process of watching the robot arm move during decoder calibration elicit activity in the brain that echoes signals that control the monkeys' own arm? What changes in the cortex allowed control skill to develop during practice? While questions like this have been investigated with a great deal of energy, they are very difficult to answer with the data that has been collected in this experiment. The openness and uncontrolled nature of the experiment, intended to make the BCI function in a realistic setting, also make the resulting data very difficult to analyze. Some of these challenges include:

1. The monkey was moving the robotic arm at the same time as its own arm, head,

and eyes.

2. The BCI task was difficult to complete, and task completion took a relatively long time; at any moment the monkey may or may not have been attempting to directly complete the task.
3. The long trial duration led to a relatively small number of trials.
4. The 6-DoF task used 36 targets to sample the control space evenly; this made averaging of trajectories for analysis difficult.
5. Robot movement did not always reflect intent exactly; actual movements were either the result of a BCI command, or the result of interaction between the robot and its environment.
6. The process of developing the system provided a moving target for data analysis. At the same time, full control of the device would have never occurred without this process.

In many ways, the data recorded in this experiment resemble a recording of a monkey freely behaving more than it does many of the highly constrained 2-D and 3-D BCI cursor movement experiments of the past. Nonetheless, the development of the ability to control the device was obvious and direct to the long-term observer; at the end of the training process, one could watch the monkey perform full control of the robot when sufficiently motivated. In the end, we can describe what occurred, but still have a difficult time explaining why.

BCI experiments that involve real-life tasks will probably suffer from similar problems. Affording natural control of a fully articulated robot can conflict with the process of understanding how this is done. Moving forward, it will be important to develop experimental paradigms that preserve the complexity and richness of natural movements, but provide more controlled experimental conditions that facilitate analysis.

9.1.1 Einstein's Razor

Everything should be made as simple as possible, but no simpler.

The working BCI system described here represents years of development work built on the foundation provided by previous BCI control experiments. Its development was the result of equal parts systematic design and trial and error. Many

versions of each of the subsystems of the experiment had been built and used over periods of time, with a great deal of work going into efforts that ultimately were not used in the final product. What has emerged from this experience has been a system that is in many ways simpler than that which was originally envisioned, but that also is more capable than one we could have originally conceived.

Because of the overall complexity of the task that we were trying to accomplish, when faced with one facet of the problem (e.g. "find a useful representation of rotation"), there was a constant tendency to want to do one of two things: (1) design very particular and complicated solutions to the immediate problem ("let's use wrist joint angles along with the Cartesian position"), or (2) reduce the specific complexity of the task in a way that makes it easier, but is not helpful in the long run ("let's start with 1-D axial rotation and go from there"). For the example of rotation, the simple solution of rotational velocity emerged after realizing that the monkey did not need to consciously understand the relationship between a rotation and the direction of that rotation's axis. Processes such as this one that enabled many aspects of the system to work seemed to be one of emerging simplicity, where solutions that were complicated and particular to our robot or experiment were passed over for ones that were simpler and more general but not initially apparent. These solutions were not adopted because they were simpler, but because they tended to work better. In the end, however, I think that we have arrived on a general model that neatly solves a lot of difficult problems.

Something that can be of value moving forward is to keep this experience in mind while designing future BCI systems. It is tempting to rely on the closed-loop nature of BCI and expect the subject to compensate for problems introduced due to overly complex or degenerate systems. In the end, however, these approaches almost never work. On the other hand, clever solutions to problems that were developed in isolation, without incrementally trying them with a subject, rarely seemed to work either. Successful progression of control occurred through a balance between systematic design and test iterations over the problem. If this entire process were repeated, many of these efforts would now be more quickly understood to be blind alleys; more rapid development may be possible if more principled approaches are taken from the beginning, as long as they can be incrementally tested.

9.2 Building Blocks for BCI

9.2.1 Brain Control of Robotic Dextrous Grasping

The brain-computer interface architecture presented in this dissertation was intended from the beginning to apply to the control of even more complex robots, specifically for dextrous robotic grasping. A large part of our development effort has been to generalize elements from 3-D linear control into concepts that can be applied to the control of a system that is potentially more complex than the one that we actually used. During the design and creation of the system described in this dissertation, it was not known what robotic hardware would be available for control. While only the 4-DoF gripper (mapped by 1-DoF of brain control) described in this work was available when the project was initiated, the intent was for later incorporation of a more highly configurable hand robot that could exhibit meaningful differences in hand shape. Therefore, each element of the system was designed to work seamlessly to include the control of dextrous finger movement. Aspects of the system that were intended for application to dextrous grasping include:

Expansion of cortical decoding While the neural cosine-tuning model was originally developed for the analysis of translational movements of the hand, the simplicity of its underlying equations made its application to the control of rotation straightforward. By thinking about high-dimensional decoding in terms of estimating a command based on a population of projected movement estimations in some vector space, the concise decoding system that evolved can be applied to finger movements just as well as hand rotation.

Observation-based decoder calibration While observation-based decoder calibration was useful for operation of the 7-DoF arm, it will be even more necessary for the operation of a BCI dextrous hand. While many specific robotic arm structures can obtain similar sets of Cartesian end-effector poses, the way that a hand interacts with the world is a direct product of both its specific morphologic structure and its set of joint poses. Therefore, a method that goes beyond thinking of a BCI robotic hand as equivalent to the natural hand of the subject is needed to provide the link to establish control. An observation-based approach provides a convenient mechanism to do this.

Multiple target pose-based shared control The passive fixturing-based shared control mechanism described in this dissertation was initially designed to assist

with the control of dextrous grasping. The hand is capable of grasping virtually any small object in a wide range of hand poses, corresponding to the "point cloud" concept of motion targeting within the shared control algorithm. The integration of an asynchronous grasp planning system in this experiment, while trivial in this example, is designed to be replaced by robotic grasp planning software that can represent a rich set of grasping and potentially manipulation behaviors.

Joint-space velocity-impedance control The low-level robot control system presented here is designed to work as well for joint-space control as Cartesian endpoint control. It is also easily extensible to branched kinematic chains so that it is easy to imagine the use of this method to provide controlled compliance during BCI grasping, and extend into a robotic hand control system capable of producing simultaneous control of contact forces and finger movement during manipulation.

The extensibility of this system will be put to the test using a robotic arm and hand system that we have recently acquired for use in a new set of BCI experiments. The Johns Hopkins University Modular Prosthetic Limb (MPL) system (Figure 9.1) will add 10+-DoF finger control to the range of brain-controlled movements described in this dissertation. To be ultimately useful in a clinical context, the ability to move a robotic prosthetic device must be initiated in a patient with no ability to move and end with elaborate control of a fully articulated prosthetic that can allow full interaction between the device and the real world. Extension of BCI to dextrous finger movements using the MPL is an important step in this process.

9.2.2 Patterns for Future BCI Systems

Many aspects of this work that were developed on the path to 7-DoF BCI control represent independent advances in clinically relevant brain-computer interface and robotic systems. The use of completely automated cortical unit discrimination methods suggests that a time-intensive spike sorting process is not required for provision of useful clinical BCIs. The use of recordings mainly from the side of the brain ipsilateral to the robot supports extension of sophisticated BCI control to patients with stroke and other neurologic conditions affecting areas normally used for limb control. The observation-based model calibration procedure provides a model for controlling movements that have not been studied equivalently in natural motion. Advances in

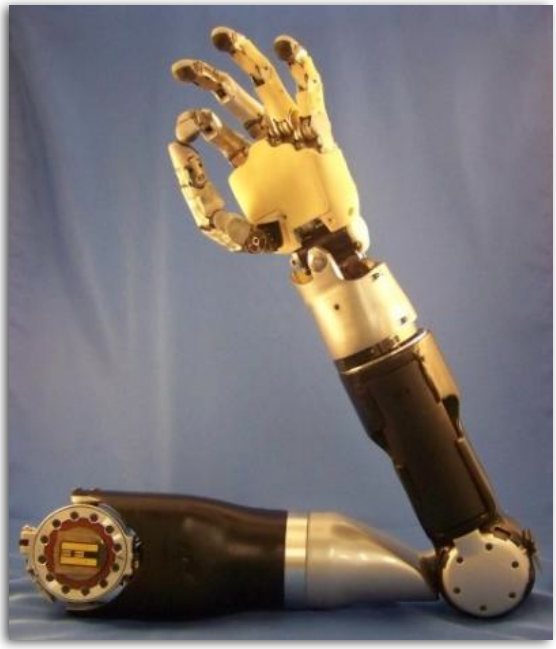


Figure 9.1: The Johns Hopkins University Applied Physics Lab Modular Prosthetic Limb (MPL).

robot control algorithms made effective control of these DoF possible in a task that simulated realistic situations in allowing the robot to freely interact with solid objects. These methods and others prototype future systems that may allow patients to realistically and practically use a BCI device during daily life.

Some themes have emerged in our research that have implications for both how complex BCI problems can be approached and what BCI device control may have to offer in the future. Our approach to cortical decoding of a large number of movement parameters simultaneously was not to build a much more sophisticated model for extraction of movement intent from cortical activity. The relationship between cortical activity and natural rotational hand motion as well as the transition from kinematic to dynamic control on physiologic motion is currently poorly understood. Instead, our focus was on the use of biologically inspired yet simple control models that were learnable by the subject. Instead of viewing the use of a BCI prosthetic as a replacement for natural control of a monkey's real arm, it is instead the introduction of a foreign object into a new control loop in the brain. This foreign object may possess certain features in common with movement of the body that may be more easily incorporated into familiar neural control mechanisms, but at its root the

brain must learn how to operate a new and complex tool. The process that allowed successful control to take place in our experiment could not have been too closely related to the usual control of each monkey's own arm; the robot arm used in this task was differently constructed and much larger than the monkeys' own.

Similarly, it is possible to create very sophisticated models for specific arm kinematics or dynamics based on primate musculoskeletal structure or neurophysiology that are intended for use in BCI. Even if these models are successfully built, they cannot be used efficiently if the design of the prosthetic robot differs much from the structure of the natural limb. As the envelope of what can be controlled by BCI has moved from the upper arm to include the wrist, we have already encountered a mismatch between robot and subject mechanics (WAM wrist) that would make specific physiology-based models difficult to use. This problem will only be compounded when moving from the control of the arm to the control of finger movements of the hand, as grasping and manipulation behaviors depend even more closely on the specific mechanical structure of the effector.

At the same time, progress toward full control of the device seemed to be encouraged by the continuous process of developing robot control algorithms that made the robot move more like a living thing. Instant advances in brain-control skill and attention to the task seemed to occur when, for instance, the robot impedance control parameters were modified to increase the fluidity of robot motions. Similarly, performance was improved when control of the redundant degree-of-freedom (elbow swing) of the robot constrained it to poses that appeared to be more physiologic. While difficult to prove conclusively, this experience suggests that it may be beneficial to carry over certain salient features of natural motor movements to promote the process of embodiment that occurs as control skill is developed. Similarly, improvements in extraction algorithms to better capture natural responses to observations of robot movement seemed to enhance control. Performance appeared to be instantly enhanced by use of the bias-reducing OLE algorithm instead of PVA to translate observation-based calibrated decoding parameters to parameters governing active intentional control. When errors were made in the cortical extraction, the resulting decoders were often similar to but not the same as ones accurately based on observation-based activity. Brain-control sessions using these parameters were never successful and usually ended with monkeys giving up on the process early in the session. This suggests as well that the process occurring during BCI skill acquisition in this case is more than one of pure operant conditioning, and instead one that can be connected at some level to

integration of the robot into a physiologic body control model.

Viewing BCI prosthetic control as a process somewhere between learning how to use a tool and learning how to use an arm frees system designers to devise BCI control schemes for things like hands without following anatomy too closely. While we rely on observation-based activity as a mechanism to calibrate our neural decoder, we already take advantage of activity that is elicited when a monkey views a robot arm that is very different from its own. The control that develops from this process is at first approach just the control of a machine that happens to look like an arm. However, features that have been observed in monkey behavior that are unnecessary for control of the robot, such as the bell-shaped velocity profile of movements, suggest a deeper connection to how brain control of physical devices occurs. The boundaries of complexity and type of devices that could potentially be controlled with a BCI have yet to be discovered.

Bibliography

[1] Jake Abbott, Panadda Marayong, and Allison Okamura. Haptic Virtual Fixtures for Robot-Assisted Manipulation. In Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte, editors, *Springer Tracts in Advanced Robotics*, pages 49–64. Springer Berlin / Heidelberg SN -, Berlin, Heidelberg, 2007. ISBN 978-3-540-48110-2. doi: 10.1007/978-3-540-48113-3{_}5. 4.1

[2] S Adee. Winner: the revolution will be prosthetized. *IEEE spectrum*, 2009. 5.1

[3] Yashar Ahmadian, Jonathan W Pillow, and Liam Paninski. Efficient Markov chain Monte Carlo methods for decoding neural spike trains. *Neural Computation*, 23(1):46–96, jan 2011. doi: 10.1162/NECO{_}a{_}00059. 2.3.2

[4] Alin Albu-Schaffer, Christian Ott, and Gerd Hirzinger. A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots. *The international journal of robotics research*, 26(1):23–39, jan 2007. doi: 10.1177/0278364907073776. 6.1

[5] P Andersen, P J Hagan, C G Phillips, and T. P. S. T3 Powell. Mapping by Microstimulation of Overlapping Projections from Area 4 to Motor Units of the Baboon's Hand. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 188(1090):31–60, jan 1975. 5.2

[6] Eishi Asano, Csaba Juhasz, Aashit Shah, Otto Muzik, Diane C Chugani, Jagdish Shah, Sandeep Sood, and Harry T Chugani. Origin and Propagation of Epileptic Spasms Delineated on Electrocorticography. *Epilepsia*, 46(7): 1086–1097, 2005. doi: 10.1111/j.1528-1167.2005.05205.x. 2.1

[7] A Bettini, P Marayong, S Lang, A.M. Okamura, and G.D. Hager. Vision-assisted control for manipulation using virtual fixtures. *Robotics, IEEE Transactions on*, 20(6):953–966, 2004. doi: 10.1109/TRO.2004.829483. 4.1, 4.2

[8] Niels Birbaumer, Ander Ramos Murguialday, and Leonardo Cohen. Brain–computer interface in paralysis. *Current Opinion in Neurology*, 21(6):634–638,

dec 2008. doi: 10.1097/WCO.0b013e328315ee2d. 2.2

[9] M.J. Black, E Bienenstock, J P Donoghue, M. Serruya, Wei Wu, and Yun Gao. Connecting brains with machines: the neural control of 2D cursor movement. In *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*, pages 580–583. IEEE, 2003. ISBN 0-7803-7579-3. doi: 10.1109/CNE.2003.1196893. 2.1

[10] Benjamin Blankertz, Klaus-Robert Müller, Dean J Krusienski, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlögl, Gert Pfurtscheller, José del R Millán, Michael Schröder, and Niels Birbaumer. The BCI competition. III: Validating alternative approaches to actual BCI problems. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 14(2):153–159, jun 2006. doi: 10.1109/TNSRE.2006.875642. 2.2

[11] A E Brockwell. Recursive Bayesian Decoding of Motor Cortical Signals by Particle Filtering. *Journal of neurophysiology*, 91(4):1899–1907, apr 2004. doi: 10.1152/jn.00438.2003. 2.3.2

[12] Peter Brunner, Anthony L Ritaccio, Joseph F Emrich, Horst Bischof, and Gerwin Schalk. Rapid Communication with a "P300" Matrix Speller Using Electrocorticographic Signals (ECoG). *Frontiers in neuroscience*, 5:5, 2011. doi: 10.3389/fnins.2011.00005. 2.2

[13] Jose M Carmena, Mikhail A Lebedev, Roy E Crist, Joseph E O'Doherty, David M Santucci, Dragan F Dimitrov, Parag G Patil, Craig S Henriquez, and Miguel A L Nicolelis. Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS biology*, 1(2):E42, nov 2003. doi: 10.1371/journal.pbio.0000042. 2.1, 2.3.1, 2.3.3, 5.1, 5.2, 7.1

[14] Hubert Cecotti. A self-paced and calibration-less SSVEP-based brain-computer interface speller. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 18(2):127–133, apr 2010. doi: 10.1109/TNSRE.2009.2039594. 2.2

[15] E K Chadwick, D Blana, J D Simeral, J Lambrecht, S P Kim, A S Cornwell, D M Taylor, L R Hochberg, J P Donoghue, and R F Kirsch. Continuous neuronal ensemble control of simulated arm reaching by a human with tetraplegia. *Journal of neural engineering*, 8(3):034003, may 2011. doi: 10.1088/1741-2560/8/3/034003. 2.1

[16] John K Chapin, Karen A Moxon, Ronald S Markowitz, and Miguel A L

Nicolelis. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *nature neuroscience*, 2(7):664–670, jul 1999. doi: 10.1038/10223. 2.1, 2.3.1

[17] Steven M Chase, Andrew B Schwartz, and Robert E Kass. Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of spiking-based brain-computer interface algorithms. *Neural networks : the official journal of the International Neural Network Society*, 22(9): 1203–1213, nov 2009. doi: 10.1016/j.neunet.2009.05.005. 2.3.3, 3.3.2, 3.3.2, 7.3

[18] M.T. Ciocarlie, S.T. Clanton, M.C. Spalding, and P.K. Allen. Biomimetic grasp planning for cortical control of a robotic hand. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2271–2276. IEEE, sep 2008. ISBN 978-1-4244-2057-5. doi: 10.1109/IROS.2008.4651179. 8.3.5

[19] C Cipriani, F Zaccone, S Micera, and M.C. Carrozza. On the Shared Control of an EMG-Controlled Prosthetic Hand: Analysis of User-Prosthesis Interaction. *Robotics, IEEE Transactions on*, 24(1):170–184, feb 2008. doi: 10.1109/TRO.2007.910708. 5.1

[20] P I Corke. A robotics toolbox for MATLAB. *Robotics & Automation Magazine, IEEE*, 3(1):24–32, mar 1996. doi: 10.1109/100.486658. 6.2.1, 6.2.1, 6.1, 6.5

[21] M Csikszentmihalyi. *Learning, flow and happiness*. Invitation to lifelong learning, 1982. 5.5.2

[22] Janis J Daly and Jonathan R Wolpaw. Brain-computer interfaces in neurological rehabilitation. *The Lancet Neurology*, 7(11):1032–1043, nov 2008. doi: doi: 10.1016/S1474-4422(08)70223-0. 2.1

[23] Mina N. Evangeliou, Vassilis Raos, Claudio Galletti, and Helen E. Savaki. Functional Imaging of the Parietal Cortex during Action Execution and Observation. *Cerebral Cortex*, 19(3):624–639, jan 2009. doi: 10.1093/cercor/bhn116. 5.2

[24] Maddalena Fabbri-Destro and Giacomo Rizzolatti. Mirror Neurons and Mirror Systems in Monkeys and Humans. *Physiology*, 23(3):171–179, jan 2008. doi: 10.1152/physiol.00004.2008. 3.4

[25] R Featherstone and D. Orin. Robot Dynamics: Equations and Algorithms. In *2000 ICRA. IEEE International Conference on Robotics and Automation*, pages 826–834. IEEE, 2000. ISBN 0-7803-5886-4. doi: 10.1109/ROBOT.2000.844153. 6.2.1, 6.3

[26] E E Fetz. Operant Conditioning of Cortical Unit Activity. *Science*, 163(3870): 955–958, feb 1969. doi: 10.1126/science.163.3870.955. 2.1, 5.2

[27] E E Fetz and M. A. Baker. Operantly conditioned patterns on precentral unit activity and correlated responses in adjacent cells and contralateral muscles. *Journal of neurophysiology*, 36(2):179–204, jan 1973. 5.2

[28] Eberhard E. Fetz and Dom V. Finocchio. Operant Conditioning of Specific Patterns of Neural and Muscular Activity. *Science*, 174(4007):431–435, jan 1971. doi: 10.1126/science.174.4007.431. 2.1

[29] George W Fraser, Steven M Chase, Andrew Whitford, and Andrew B Schwartz. Control of a brain-computer interface without spike sorting. *Journal of neural engineering*, 6(5):055004, oct 2009. doi: 10.1088/1741-2560/6/5/055004. 8.3.1

[30] V Gallese, L Fadiga, L Fogassi, and G Rizzolatti. Action recognition in the premotor cortex. *Brain : a journal of neurology*, 119 (Pt 2):593–609, apr 1996. 3.4, 5.2

[31] A Georgopoulos, A Schwartz, and R Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, sep 1986. doi: 10.1126/science.3749885. 2.3.2, 3.1.1

[32] A P Georgopoulos, J F Kalaska, R Caminiti, and J T Massey. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 2(11):1527–1537, nov 1982. 3.1, 5.2, 7.1

[33] Mark Guadagnoli and Timothy Lee. Challenge Point: A Framework for Conceptualizing the Effects of Various Practice Conditions in Motor Learning. *Journal of Motor Behavior*, 36(2):212–224, jul 2004. doi: 10.3200/JMBR.36.2.212-224. 5.5.2

[34] RS Hartenberg. A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied mechanics*, 77(2):215, 1955. 6.2.1

[35] Nicho Hatsopoulos. Personal Communication. jul 2011. 8.2.1

[36] Nicholas G Hatsopoulos and John P Donoghue. The Science of Neural Interface Systems. *Annual review of neuroscience*, 32(1):249–266, jun 2009. doi: 10.1146/annurev.neuro.051508.135241. 2.2

[37] S Q He, R P Dum, and P L Strick. Topographic organization of corticospinal projections from the frontal lobe: motor areas on the lateral surface of the hemisphere. *The Journal of neuroscience : the official journal of the Society*

for Neuroscience, 13(3):952–980, mar 1993. 7.3

[38] Rodolphe Héliot, Karunesh Ganguly, Jessica Jimenez, and Jose M Carmena. Learning in closed-loop brain-machine interfaces: modeling and experimental validation. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 40(5):1387–1397, oct 2010. doi: 10.1109/TSMCB.2009.2036931. 2.3.3, 5.2

[39] S H Hendry, C R Houser, E G Jones, and J E Vaughn. Synaptic organization of immunocytochemically identified GABA neurons in the monkey sensory-motor cortex. *Journal of neurocytology*, 12(4):639–660, aug 1983. 8.2.1

[40] Leigh R Hochberg, Mijail D Serruya, Gerhard M Friehs, Jon A Mukand, Maryam Saleh, Abraham H Caplan, Almut Branner, David Chen, Richard D Penn, and John P Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171, jul 2006. doi: 10.1038/nature04970. 2.1, 2.3.1, 5.2

[41] N Hogan. Impedance control: an approach to manipulation—part II: implementation. *Journal of Dynamic Systems, Measurement, and Control*, 1985. 6.1, 6.4, 7.4.1

[42] Neville Hogan. Impedance Control: An Approach to Manipulation. In *American Control Conference, 1984*, pages 304–313. IEEE, 1985. 6.3, 6.4

[43] R E Isaacs, D J Weber, and A B Schwartz. Work toward real-time control of a cortical neural prothesis. *Rehabilitation Engineering, IEEE Transactions on*, 8 (2):196–198, jun 2000. 2.1

[44] Herbert Jasper and Wilder Penfield. Electrocorticograms in man: Effect of voluntary movement upon the electrical activity of the precentral gyrus. *European Archives of Psychiatry and Clinical Neuroscience*, 183(1):163–174, jan 1949. doi: 10.1007/BF01062488. 2.1

[45] GG Judge, WE Griffiths, and RC Hill. *Theory and practice of econometrics*. 1985. 3.3.2

[46] Shinji Kakei, Donna S. Hoffman, and Peter L. Strick. Muscle and Movement Representations in the Primary Motor Cortex. *Science*, 285(5436):2136–2139, jan 1999. doi: 10.1126/science.285.5436.2136. 3.2, 5.2

[47] Thomas R. Kane and David A. Levinson. The Use of Kane's Dynamical Equations in Robotics. *The international journal of robotics research*, 2(3):3–21, jan 1983. doi: 10.1177/027836498300200301. 6.2

[48] Sang Hoon Kang, Maolin Jin, and Pyung Hun Chang. A Solution to the Accuracy/Robustness Dilemma in Impedance Control. *IEEE/ASME Transactions on Mechatronics*, 14(3):282–294. doi: 10.1109/TMECH.2008.2005524. 6.1

[49] Robert E Kass, Valérie Ventura, and Emery N Brown. Statistical Issues in the Analysis of Neuronal Data. *Journal of neurophysiology*, 94(1):8–25, jan 2005. doi: 10.1152/jn.00648.2004. 3.3.1

[50] P R Kennedy and R A Bakay. Restoration of neural output from a paralyzed patient by a direct brain connection. *Neuroreport*, 9(8):1707–1711, jun 1998. 2.1

[51] Philip R. Kennedy, Suzanne S. Mirra, and Roy A.E. Bakay. The cone electrode: Ultrastructural studies following long-term recording in rat and monkey cortex. *Neuroscience letters*, 142(1):89–94, aug 1992. doi: doi:10.1016/0304-3940(92)90627-J. 2.1

[52] RE Kettner, AB Schwartz, and AP Georgopoulos. Primate motor cortex and free arm movements to visual targets in three- dimensional space. III. Positional gradients and population coding of movement direction from various movement origins. *The Journal of Neuroscience*, 8(8):2938–2947, jan 1988. 3.1

[53] O Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The international journal of robotics research*, 5(1):90–98, mar 1986. doi: 10.1177/027836498600500106. 5.1

[54] H.K. Kim, J Biggs, W Schloerb, M Carmena, M.A. Lebedev, M.A.L. Nicolelis, and M.A. Srinivasan. Continuous shared control for stabilizing reaching and grasping with brain-machine interfaces. *Biomedical Engineering, IEEE Transactions on*, 53(6):1164–1173, jun 2006. doi: 10.1109/TBME.2006.870235. 5.1, 5.3.1

[55] Sung-Phil Kim, John D Simeral, Leigh R Hochberg, John P Donoghue, and Michael J Black. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *Journal of neural engineering*, 5(4):455–476, dec 2008. doi: 10.1088/1741-2560/5/4/010. 2.1

[56] Shinsuke Koyama, Steven M Chase, Andrew S Whitford, Meel Velliste, Andrew B Schwartz, and Robert E Kass. Comparison of brain-computer interface decoding algorithms in open-loop and closed-loop control. *Journal of Computational Neuroscience*, 29(1-2):73–87, aug 2010. doi: 10.1007/s10827-009-0196-9. 2.3.3, 3.3.2

[57] D Kragic, P Marayong, M Li, A.M. Okamura, and G.D. Hager. Human-Machine Collaborative Systems for Microsurgical Applications. *The international journal of robotics research*, 24(9):731–741, jan 2005. doi: 10.1177/0278364905057059. 4.1, 4.2, 4.2

[58] H I Krebs, J J Palazzolo, L Dipietro, M Ferraro, J Krol, K Rannekleiv, B T Volpe, and N Hogan. Rehabilitation Robotics: Performance-Based Progressive Robot-Assisted Therapy. *Autonomous Robots*, 15(1):7–20, jan 2003. doi: 10.1023/A:1024494031121. 5.5.2

[59] Hermano Igo Krebs, Bruce Volpe, and Neville Hogan. A working model of stroke recovery from rehabilitation robotics practitioners. *Journal of neuroengineering and rehabilitation*, 6:6, 2009. doi: 10.1186/1743-0003-6-6. 5.5.2

[60] T A Kuiken, G Li, B A Lock, R D Lipschutz, L A Miller, K A Stubblefield, and K B Englehart. Targeted Muscle Reinnervation for Real-time Myoelectric Control of Multifunction Artificial Arms. *JAMA: The Journal of the American Medical Association*, 301(6):619–628, feb 2009. doi: 10.1001/jama.2009.116. 5.2

[61] Rajesh Kumar, Gregory Hager, Aaron Barnes, Patrick Jensen, and Russell Taylor. An Augmentation System for Fine Manipulation. In Scott Delp, Anthony DiGoia, and Branislav Jaramaz, editors, *Lecture Notes in Computer Science*, pages CH404–CH404. Springer Berlin / Heidelberg SN -, Berlin, Heidelberg, 2000. ISBN 978-3-540-41189-5. doi: 10.1007/978-3-540-40899-4{_}99. 5.1

[62] Gert Kwakkel, Boudewijn J. Kollen, and Hermano I. Krebs. Effects of Robot-Assisted Therapy on Upper Limb Recovery After Stroke: A Systematic Review. *Neurorehabilitation and Neural Repair*, 22(2):111–121, sep 2007. doi: 10.1177/1545968307305457. 5.5.2

[63] H C Kwan, J T Murphy, and Y C Wong. Interaction between neurons in precentral cortical zones controlling different joints. *Brain research*, 400(2): 259–269, jan 1987. 8.2.1

[64] Mikhail A Lebedev and Miguel A L Nicolelis. Brain-machine interfaces: past, present and future. *TRENDS in Neurosciences*, 29(9):536–546, sep 2006. doi: doi:10.1016/j.tins.2006.07.004. 2.2

[65] Roger Lemon. The output map of the primate motor cortex. *TRENDS in Neurosciences*, 11(11):501–506, jan 1988. doi: 10.1016/0166-2236(88)90012-4. 5.2

[66] EC Leuthardt, G Schalk, and JR Wolpaw. A brain–computer interface using

electrocorticographic signals in humans. *Journal of neural ...*, 2004. 2.1

[67] Eric C Leuthardt, Gerwin Schalk, Jarod Roland, Adam Rouse, and Daniel W Moran. Evolution of brain-computer interfaces: going beyond classic motor physiology. *Neurosurgical focus*, 27(1):E4, jul 2009. doi: 10.3171/2009.4.FOCUS0979. 2.2

[68] J Y S Luh, M W Walker, and R P C Paul. On-Line Computational Scheme for Mechanical Manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, jun 1980. doi: 10.1115/1.3149599. 6.2.1, 6.2.1

[69] J.N. Mak and J R Wolpaw. Clinical Applications of Brain-Computer Interfaces: Current State and Future Prospects. *Biomedical Engineering, IEEE Reviews in*, 2:187–199, dec 2009. doi: 10.1109/RBME.2009.2035356. 2.1

[70] MT Mason. Mechanics of robotic manipulation, 2001. 4.2

[71] Dennis J McFarland, Dean J Krusienski, William A Sarnacki, and Jonathan R Wolpaw. Emulation of computer mouse control with a noninvasive brain–computer interface. *J Neural Engineering*, 5(2):101–110, mar 2008. doi: 10.1088/1741-2560/5/2/001. 2.1

[72] Daniel Moran. Evolution of brain-computer interface: action potentials, local field potentials and electrocorticograms. *Current opinion in neurobiology*, 20 (6):741–745, dec 2010. doi: doi:10.1016/j.conb.2010.09.010. 2.1

[73] Daniel W Moran and Andrew B Schwartz. Motor Cortical Representation of Speed and Direction During Reaching. *Journal of neurophysiology*, 82(5):2676–2692, jan 1999. 2.3.2, 3.1.1

[74] Lindsay M. Oberman, Joseph P. McCleery, Vilayanur S. Ramachandran, and Jaime A. Pineda. EEG evidence for mirror neuron activity during the observation of human and robot actions: Toward an analysis of the human qualities of interactive robots. *Neurocomputing*, 70(13-15):2194–2203, aug 2007. doi: doi:10.1016/j.neucom.2006.02.024. 3.4

[75] Marcia K O'Malley, Abhishek Gupta, Matthew Gen, and Yanfang Li. Shared Control in Haptic Systems for Performance Enhancement and Training. *Journal of Dynamic Systems, Measurement, and Control*, 128(1):75–85, mar 2006. doi: 10.1115/1.2168160. 4.1, 4.2

[76] DE Orin, RB McGhee, and M Vukobratovic. Kinematic and kinetic analysis of open-chain linkages utilizing Newton-Euler methods. *Mathematical ...*, 1979. 6.2.1

[77] C Ott. Cartesian impedance control of redundant and flexible-joint robots, 2008. 6.1

[78] Liam Paninski, Matthew R Fellows, Nicholas G Hatsopoulos, and John P Donoghue. Spatiotemporal Tuning of Motor Cortical Neurons for Hand Position and Velocity. *Journal of neurophysiology*, 91(1):515–532, jan 2004. doi: 10.1152/jn.00587.2002. 2.3.1

[79] Shinsuk Park, Robert Howe, and David Torchiana. Virtual Fixtures for Robotic Cardiac Surgery. In Wiro Niessen and Max Viergever, editors, *Lecture Notes in Computer Science*, pages 1419–1420. Springer Berlin / Heidelberg SN -, Berlin, Heidelberg, oct 2001. ISBN 978-3-540-42697-4. doi: 10.1007/3-540-45468-3{_} 252. 4.1

[80] P Hunter Peckham and Jayme S Knutson. Functional Electrical Stimulation for Neuromuscular Applications. *Annu Rev Biomed Eng*, 7(1):327–360, jan 2011. doi: doi:10.1146/annurev.bioeng.6.040803.140103. 2.2

[81] P Hunter Peckham, Kevin L Kilgore, Michael W Keith, Anne M Bryden, Niloy Bhadra, and Fred W Montague. An advanced neuroprosthesis for restoration of hand and upper arm control using an implantable controller. *The Journal of hand surgery*, 27(2):265–276, mar 2002. 2.2

[82] Xavier Perrin, Ricardo Chavarriaga, Francis Colas, Roland Siegwart, and José del R Millán. Brain-coupled interaction for semi-autonomous navigation of an assistive robot. *Robotics And Autonomous Systems*, 58(12):1246–1255, dec 2010. doi: 10.1016/j.robot.2010.05.010. 5.1

[83] Gail B Peterson. A day of great illumination: B. F. Skinner's discovery of shaping. *Journal of the experimental analysis of behavior*, 82(3):317–328, nov 2004. doi: 10.1901/jeab.2004.82-317. 5.5.1

[84] G Pfurtscheller, G.R. Muller-Putz, R Scherer, and C. Neuper. Rehabilitation with Brain-Computer Interface Systems. *Computer*, 41(10):58–65, oct 2008. doi: 10.1109/MC.2008.432. 2.2

[85] Tobias Pistohl, Tonio Ball, Andreas Schulze-Bonhage, Ad Aertsen, and Carsten Mehring. Prediction of arm movement trajectories from ECoG-recordings in humans. *Journal of neuroscience methods*, 167(1):105–114, 2008. doi: doi: 10.1016/j.jneumeth.2007.10.001. 2.1

[86] Andrew V. Poliakov and Marc H. Schieber. Limited Functional Grouping of Neurons in the Motor Cortex Hand Area During Individuated Finger Move-

ments: A Cluster Analysis. *Journal of neurophysiology*, 82(6):3488–3505, jan 1999. 5.2

[87] R Porter and R Lemon. Corticospinal function and voluntary movement. . Clarendon Press, 1993. 8.2.1

[88] Dane Powell and Marcia K O'Malley. Efficacy of shared-control guidance paradigms for robot-mediated training. In *World Haptics Conference (WHC), 2011 IEEE*, pages 427–432. IEEE. ISBN 978-1-4577-0299-0. doi: 10.1109/WHC.2011.5945524. 4.1

[89] HX Qi and I Stepniewska. Reorganization of primary motor cortex in adult macaque monkeys with long-standing amputations. *Journal of neurophysiology*, 2000. 8.2.1

[90] Vassilis Raos, Mina N. Evangeliou, and Helen E. Savaki. Observation of action: grasping with the mind's hand. *Neuroimage*, 23(1):193–201, 2004. doi: doi: 10.1016/j.neuroimage.2004.04.024. 5.2

[91] J A Rathelot. Muscle representation in the macaque motor cortex: An anatomical perspective. *Proceedings of the National Academy of Sciences*, 103(21): 8257–8262, may 2006. doi: 10.1073/pnas.0602933103. 5.2

[92] Gregg Recanzone, William Jenkins, Gary Hradek, and Michael Merzenich. A behavioral frequency discrimination paradigm for use in adult primates. *Behavior Research Methods*, 23(3):357–369, jan 1991. 5.5.2

[93] G. Anthony Reina, Daniel W Moran, and Andrew B Schwartz. On the Relationship Between Joint Angular Velocity and Motor Cortical Discharge During Reaching. *Journal of neurophysiology*, 85(6):2576–2589, jan 2001. 3.5.2

[94] Giacomo Rizzolatti and Laila Craighero. THE MIRROR-NEURON SYSTEM. *Annual review of neuroscience*, 27(1):169–192, jan 2011. doi: doi: 10.1146/annurev.neuro.27.070203.144230. 5.2

[95] L.B. Rosenberg. Proceedings of IEEE Virtual Reality Annual International Symposium. In *IEEE Virtual Reality Annual International Symposium*, pages 76–82. IEEE, 1993. ISBN 0-7803-1363-1. doi: 10.1109/VRAIS.1993.380795. 4.1, 5.1

[96] Emilio Salinas and L F Abbott. Vector reconstruction from firing rates. *Journal of Computational Neuroscience*, 1(1-2):89–107, jun 1994. doi: 10.1007/BF00962720. 3.3.1, 3.3.2, 7.3

[97] JC Sanchez. Brain-machine interface engineering, 2007. 2.3.1, 2.3.2

[98] Justin C Sanchez, Babak Mahmoudi, Jack DiGiovanna, and Jose C Principe. Exploiting co-adaptation for the design of symbiotic neuroprosthetic assistants. *Neural Networks*, 22(3):305–315, apr 2009. doi: 10.1016/j.neunet.2009.03.015. 5.1

[99] G Schalk, D.J. McFarland, T. Hinterberger, N Birbaumer, and J R Wolpaw. BCI2000: a general-purpose brain-computer interface (BCI) system. *Biomedical Engineering, IEEE Transactions on*, 51(6):1034–1043, 2004. doi: 10.1109/TBME.2004.827072. 2.1

[100] G Schalk, J Kubánek, K J Miller, N R Anderson, E C Leuthardt, J G Ojemann, D Limbrick, D Moran, L A Gerhardt, and J R Wolpaw. Decoding two-dimensional movement trajectories using electrocorticographic signals in humans. *Journal of neural engineering*, 4(3):264–275, sep 2007. doi: 10.1088/1741-2560/4/3/012. 2.1

[101] G Schalk, K J Miller, N R Anderson, J A Wilson, M D Smyth, J G Ojemann, D W Moran, J R Wolpaw, and E C Leuthardt. Two-dimensional movement control using electrocorticographic signals in humans. *J Neural Engineering*, 5 (1):75–84, feb 2008. doi: 10.1088/1741-2560/5/1/008. 2.1

[102] Andrew B Schwartz and Daniel W Moran. Motor Cortical Activity During Drawing Movements: Population Representation During Lemniscate Tracing. *Journal of neurophysiology*, 82(5):2705–2718, jan 1999. 2.3.2

[103] Andrew B Schwartz, Dawn M Taylor, and Stephen I Helms Tillery. Extraction algorithms for cortical control of arm prosthetics. *Current opinion in neurobiology*, 11(6):701–708, 2001. doi: doi:10.1016/S0959-4388(01)00272-0. 2.3.2, 3.1.1

[104] Eric W Sellers and Emanuel Donchin. A P300-based brain-computer interface: Initial tests by ALS patients. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, 117(3):538–548, mar 2006. doi: doi:10.1016/j.clinph.2005.06.027. 2.1

[105] EW Sellers and JR Wolpaw. The Wadsworth BCI research and development program: at home with BCI. *Neural Systems and ...*, 2006. 2.2

[106] Mijail D Serruya, Nicholas G Hatsopoulos, Liam Paninski, Matthew R Fellows, and John P Donoghue. Instant neural control of a movement signal. *Nature*, 416(6877):141–142, mar 2002. doi: 10.1038/416141a. 2.1, 2.3.1, 5.1, 5.2, 7.1

[107] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of*

the 12th annual conference on Computer graphics and interactive techniques, pages 245–254, New York, NY, USA, 1985. ACM. ISBN 0-89791-166-0. doi: http://doi.acm.org/10.1145/325334.325242. 4.3.3

[108] B Siciliano. Springer handbook of robotics, 2008. 6.1

[109] MW Spong and S Hutchinson. Robot modeling and control, 2006. 6.1, 6.2

[110] MW Spong and M Vidyasagar. Robot dynamics and control, 2008. 6.2

[111] B E Swartz and E S Goldensohn. Timeline of the history of EEG and associated fields. *Electroencephalography and clinical neurophysiology*, 106(2):173–176, feb 1998. 2.1

[112] D M Taylor, S.I.H. Tillery, and A B Schwartz. Information conveyed through brain-control: cursor versus robot. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 11(2):195–199, 2003. doi: 10.1109/TNSRE.2003.814451. 2.1, 7.1

[113] Dawn M Taylor, Stephen I Helms Tillery, and Andrew B Schwartz. Direct cortical control of 3D neuroprosthetic devices. *Science*, 296(5574):1829–1832, jun 2002. doi: 10.1126/science.1070291. 2.1, 3.1.1, 5.1, 5.2, 5.4, 7.1, 8

[114] Russell Taylor, Pat Jensen, Louis Whitcomb, Aaron Barnes, Rajesh Kumar, Dan Stoianovici, Puneet Gupta, Zhengxian Wang, Eugene Dejuan, and Louis Kavoussi. A Steady-Hand Robotic System for Microsurgical Augmentation. *The international journal of robotics research*, 18(12):1201–1210, jan 1999. doi: 10.1177/02783649922067807. 4.2

[115] Dennis Tkach, Jacob Reimer, and Nicholas G Hatsopoulos. Congruent activity during action and action observation in motor cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 27(48):13241–13250, nov 2007. doi: 10.1523/JNEUROSCI.2895-07.2007. 3.4, 5.2

[116] Dennis Tkach, Jake Reimer, and Nicholas G Hatsopoulos. Observation-based learning for brain-machine interfaces. *Current opinion in neurobiology*, 18(6): 589–594, dec 2008. doi: 10.1016/j.conb.2008.09.016. 3.4, 5.2

[117] E Todorov. Direct cortical control of muscle activation in voluntary arm movements: a model. *nature neuroscience*, 3(4):391–398, apr 2000. doi: 10.1038/73964. 7.4.1

[118] E Todorov. On the role of primary motor cortex in arm movement control. *Progress in Motor Control: Effects of age*, 2004. 2.3.2

[119] C Toledo, L Leija, R Muñoz, A Vera, and A Ramirez. 2009 Pan American Health Care Exchanges. In *2009 Pan American Health Care Exchanges*, pages 104–108. IEEE. ISBN 978-1-4244-3668-2. doi: 10.1109/PAHCE.2009.5158375. 5.1

[120] Paul Tuffield and Hugo Elias. The Shadow robot mimics human actions. *Industrial Robot: An International Journal*, 30(1):56–60, 2003. doi: 10.1108/01439910310457715. 5.1

[121] Meel Velliste, Sagi Perel, M Chance Spalding, Andrew S Whitford, and Andrew B Schwartz. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453(7198):1098–1101, jun 2008. doi: 10.1038/nature06996. 2.2, 3.1.1, 3.4, 5.1, 5.1, 5.2, 5.4, 5.5.2, 6.1, 7.1, 7.4, 8, 8.3.2

[122] J J Vidal. Toward direct brain-computer communication. *Annual review of biophysics and bioengineering*, 2:157–180, 1973. doi: 10.1146/annurev.bb.02.060173.001105. 2.1

[123] Remy Wahnoun, Jiping He, and Stephen I Helms Tillery. Selection and parameterization of cortical neurons for neuroprosthetic control. *Journal of neural engineering*, 3(2):162–171, may 2006. doi: 10.1088/1741-2560/3/2/010. 3.4

[124] M W Walker and D E Orin. Efficient Dynamic Computer Simulation of Robotic Mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205, 1982. doi: 10.1115/1.3139699. 6.5

[125] Wei Wang, Sherwin S Chan, Dustin A Heldman, and Daniel W Moran. Motor cortical representation of position and velocity during reaching. *Journal of neurophysiology*, 97(6):4258–4270, jun 2007. doi: 10.1152/jn.01180.2006. 2.3.2

[126] Wei Wang, Sherwin S Chan, Dustin A Heldman, and Daniel W Moran. Motor cortical representation of hand translation and rotation during reaching. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 30(3):958–962, jan 2010. doi: 10.1523/JNEUROSCI.3742-09.2010. 2.3.2, 3.2, 3.4

[127] N Wiener. Extrapolation, interpolation and smoothing of stationary time series. 1949. *New York (1968)*. 2.3.1

[128] Jonathan R Wolpaw, Dennis J McFarland, Gregory W. Neat, and Catherine A. Forneris. An EEG-based brain-computer interface for cursor control. *Electroencephalography and clinical neurophysiology*, 78(3):252–259, mar 1991. doi: doi:10.1016/0013-4694(91)90040-B. 2.1

[129] Jonathan R Wolpaw, Niels Birbaumer, Dennis J McFarland, Gert Pfurtscheller, and Theresa M. Vaughan. Brain-computer interfaces for communication and control. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, 113(6):767–791, 2002. doi: doi:10.1016/S1388-2457(02)00057-3. 2.1

[130] Wei Wu, Yun Gao, Elie Bienenstock, John P Donoghue, and Michael J Black. Bayesian Population Decoding of Motor Cortical Activity Using a Kalman Filter. *Neural Computation*, 18(1):80–118, jan 2006. doi: 10.1162/089976606774841585. 2.3.2

[131] M Wyndaele and J-J Wyndaele. Incidence, prevalence and epidemiology of spinal cord injury: what learns a worldwide literature survey? *Spinal cord : the official journal of the International Medical Society of Paraplegia*, 44(9): 523–529, sep 2006. doi: 10.1038/sj.sc.3101893. 7.1

[132] Jijie Xu, Garrett G Grindle, Ben Salatin, Juan J Vazquez, Hongwu Wang, Dan Ding, and Rory A Cooper. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, pages 5042–5047. IEEE. ISBN 978-1-4244-6674-0. doi: 10.1109/IROS.2010.5652775. 8.3.3