2016

# A brain-machine interface for assistive robotic control

BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

# A BRAIN-MACHINE INTERFACE FOR ASSISTIVE ROBOTIC CONTROL

by

**BYRON V. GALBRAITH**

B.S., University of Illinois at Chicago, Chicago Illinois, 2006
M.S., Marquette University and the Medical College of Wisconsin, Milwaukee Wisconsin, 2010

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2016

Approved by

First Reader

Frank H. Guenther, Ph.D.
Professor, Department of Speech, Language, and Hearing Sciences

Second Reader

Massimiliano Versace, Ph.D.
Research Assistant Professor, Center for Computational Neuroscience and Neural Technology

Third Reader

Deniz Erdogmus, Ph.D.
Associate Professor, Electrical and Computer Engineering
Northeastern University, College of Engineering

## ACKNOWLEDGMENTS

I would like to thank the following people for their contribution to the completion of this dissertation. My advisors, Frank Guenther and Max Versace, provided invaluable guidance, mentorship, and perspective during the course of my research. My committee chair, Dan Bullock, offered support and assistance throughout my time in the Cognitive and Neural Systems program. My officemates and colleagues at Boston University assisted, encouraged, and motivated me to work harder.

I would like to especially thank my wife, Karen, for her patience and support through the entire process and my children, Zephan and Tamzin, for the joy they brought me every day.

This work was supported in part by the Center of Excellence for Learning in Education, Science, and Technology, a National Science Foundation Science of Learning Center (NSF SMA-0835976).

**A BRAIN-MACHINE INTERFACE FOR ASSISTIVE ROBOTIC CONTROL**

**BYRON V. GALBRAITH**

Boston University Graduate School of Arts and Sciences, 2016

Major Professor: Frank H. Guenther, Ph.D. Professor, Department of Speech, Language, and Hearing Sciences

ABSTRACT

Brain-machine interfaces (BMIs) are the only currently viable means of communication for many individuals suffering from locked-in syndrome (LIS) – profound paralysis that results in severely limited or total loss of voluntary motor control. By inferring user intent from task-modulated neurological signals and then translating those intentions into actions, BMIs can enable LIS patients increased autonomy. Significant effort has been devoted to developing BMIs over the last three decades, but only recently have the combined advances in hardware, software, and methodology provided a setting to realize the translation of this research from the lab into practical, real-world applications. Non-invasive methods, such as those based on the electroencephalogram (EEG), offer the only feasible solution for practical use at the moment, but suffer from limited communication rates and susceptibility to environmental noise. Maximization of the efficacy of each decoded intention, therefore, is critical.

This thesis addresses the challenge of implementing a BMI intended for practical use with a focus on an autonomous assistive robot application. First an adaptive EEG-based BMI strategy is developed that relies upon code-modulated visual evoked potentials (c-VEPs) to infer user intent. As voluntary gaze control is typically not

available to LIS patients, c-VEP decoding methods under both gaze-dependent and gaze-independent scenarios are explored. Adaptive decoding strategies in both offline and online task conditions are evaluated, and a novel approach to assess ongoing online BMI performance is introduced.

Next, an adaptive neural network-based system for assistive robot control is presented that employs exploratory learning to achieve the coordinated motor planning needed to navigate toward, reach for, and grasp distant objects. Exploratory learning, or "learning by doing," is an unsupervised method in which the robot is able to build an internal model for motor planning and coordination based on real-time sensory inputs received during exploration.

Finally, a software platform intended for practical BMI application use is developed and evaluated. Using online c-VEP methods, users control a simple 2D cursor control game, a basic augmentative and alternative communication tool, and an assistive robot, both manually and via high-level goal-oriented commands.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

AAC ..... Alternative and Augmentative Communication

API ..... Application Programming Interface

BCI ..... Brain Computer Interface

BMI ..... Brain Machine Interface

c-VEP ..... Code-modulated Visual Evoked Potential

CCA ..... Canonical Correlation Analysis

CoCoRo ..... Cognitive Co-Robot

EEG ..... Electroencephalography

ERP ..... Event-Related Potential

ITR ..... Information Transfer Rate

LDA ..... Linear Discriminant Analysis

LIS ..... Locked-In Syndrome

LSL ..... Lab Streaming Layer

RBF ..... Radial Basis Function

RMSE ..... Root Mean Square Error

SNR ..... Signal-to-Noise Ratio

SSVEP ..... Steady State Visual Evoked Potential

TLX ..... Task Load Index

# 1. INTRODUCTION

## 1.1. Problem Statement

Brain-machine interfaces, also called brain-computer interfaces (BCI), are the only currently viable means of communication for individuals suffering from locked-in syndrome (LIS) – profound paralysis that results in severely limited or total loss of voluntary motor control. By inferring user intent from task-modulated neurological signals and then translating those intentions into actions, BCIs can enable LIS patients increased autonomy. Significant effort has been devoted to developing BCIs over the last three decades, but only recently have the combined advances in hardware, software, and methodology provided a setting to realize the translation of this research from the lab into practical, real-world applications. Non-invasive methods, such as those based on the electroencephalogram (EEG), offer the only feasible solution for practical use, but suffer from limited communication rates. Maximization of the efficacy of each decoded intention, therefore, is critical. This can be achieved through BCI-controlled mobile robots with reaching capabilities that can autonomously translate a limited set of high-level commands into complex environmental interactions.

## 1.2. Contribution

The contribution of the work presented herein is fourfold. First, adaptive code-modulated visual evoked potential (c-VEP) BCI methods were developed and evaluated for online practical use. Second, a control system for an assistive robot was created that

imbued the robot with autonomous capabilities for reaching and grasping remote objects. Third, the online c-VEP BCI method was used to control and direct a robot embodied with the autonomous control system to navigate toward distant targets. Finally, in order to support both the BCI research and the development of BCI applications such as the robot interface, the Unlock framework, a Python-based BCI software platform, was developed.

### 1.3. Organization

The rest of this document is organized as follows. Chapter 2 describes a novel method for an adaptive c-VEP BCI geared toward practical applications, while Chapter 3 describes a neural-network based system for embodying an assistive robot with autonomous reaching and navigation capabilities. In Chapter 4, the Unlock framework is described in detail and the online control of user applications, such as directing an autonomous robot to navigate toward distant objects, is demonstrated using the c-VEP BCI method. Finally, Chapter 5 concludes the dissertation with a summary of all the work presented herein and identifies potential areas of future work and direction.

# 2. AN ADAPTIVE CODE-MODULATED VISUAL BCI METHOD FOR PRACTICAL APPLICATIONS

## 2.1. Introduction

### 2.1.1 Brain-Computer Interfaces

A brain-computer interface (BCI) is a system that acquires neural activity from a user, processes that signal to identify and extract relevant features, classifies those features to decode user intent, and then translates that decoded intent into a computerized action (He et al., 2012). For individuals suffering from locked-in syndrome (LIS), which involves intact cognition with near or total loss of voluntary motor control, a BCI may be their only means of communication. These individuals do not live in research labs, so in order for them to gain real-world benefit from advances in BCI technology, the BCI must be made practical and robust enough for home use.

For a BCI to be practical it needs to address issues other than maximizing information transfer rate (ITR), such as maintaining a high level of decision accuracy over extended periods of time, operating in noisy, non-clinical environments, and supporting easy maintenance by caregivers and other non-BCI experts. Adaptive BCI decoders and tasks offer great promise for practical BCI, as they can be designed to automatically respond to changes in signal quality brought on by both environmental noise and internal state of the user.

### 2.1.2. Visual Evoked Potentials

A common BCI paradigm is to detect sensory evoked potentials in electroencephalography (EEG) data (He et al., 2012). These events result from a variety of sensory stimuli including visual, auditory, and somatosensory. In each case a detectable change in EEG can be correlated to a particular attended stimulus delivered via one of the above modalities. For example, in steady state visual evoked potentials (SSVEP), continual attention to a visual stimulus flickering at a fixed rate above 4Hz entrains the visual cortex to the flicker pattern, resulting in a corresponding pattern of activity in the EEG signals recorded over the visual cortical region.

SSVEP-based methods can be divided into two categories based on the stimulus presentation paradigm: frequency and phase. Frequency (f-VEP) methods (Middendorf et al., 2000) present multiple stimuli that each flicker at a different, constant rate (e.g. 12, 13, 14, 15 Hz). Like f-VEP, phase (p-VEP) methods (Jia et al., 2011) present multiple stimuli that flicker at a constant rate, though instead of varying the frequency, a single frequency is used with varying phase offsets.

Alternatives to steady state presentation methods are impulse-like methods that evaluate repeatability of the EEG response in the time domain. Code-modulated (c-VEP) methods use maximum length sequences (m-sequences) (Sutter, 1992) to describe a pseudorandom flashing pattern presented at a fixed display frequency. A key property of m-sequences is that they have an autocorrelation function that approximates an impulse signal. This makes them attractive for stimulus flicker patterns in VEP-based BCI, as any phase shifts in the presentation cycle will correspond to equal shifts in the resulting EEG

response pattern. This is exploited by presenting multiple targets starting at different time lags of the m-sequence, and then using the linear cross-correlation of the samples with a template to determine the offset corresponding to the attended target. The template is generated during a training phase, where the EEG responses to several m-sequence presentation cycles at a fixed phase offset are obtained and used to train a classifier. In addition to using a single m-sequence with several phase-shifted targets, a method using multiple, concurrent m-sequences has also been demonstrated (Nezamfar et al., 2011). This multi-sequence c-VEP paradigm is the basis for the methods explored throughout this work.

### 2.1.3. Gaze Independence

Gaze-dependence, an implicit requirement of most visual-based BCI, renders the BCI impractical for real-world use. If the subject has reliable gaze control, an eye tracker is significantly more reliable than a BCI (Pasqualotto et al., 2015), while if they do not, the BCI may not work at all. To offer a practical solution, visual BCIs need to assume gaze-independence. Gaze-independence in the context of BCI takes on two different forms. First, there are gaze-fixed, covert attention paradigms where individual target stimuli are attended to in the visual periphery while gaze is directed elsewhere, typically at a central fixation point (Kelly et al., 2005). Second and less frequent, there are non-spatially selective paradigms, where the stimuli either alternate over the fixation point or take up the most of the field of view, such as rapid serial visual presentation (Hild et al.

2011; Acqualagna and Blankertz, 2013) and using overlapping stimuli (Allison et al., 2008; Zhang et al., 2010), respectively.

VEPs are strongest when both gaze and attention is directed at the target stimulus. When covertly attending to stimuli, the evoked potentials are still detectable, albeit at lower signal-to-noise ratio (SNR) (Walter et al., 2012). Finding ways to boost these signals is thus critical for reliable covert attention BCI. Co-adaptive methods that employ both machine learning on the signal processing side and user guidance via feedback on the task presentation side are one way to boost SNR in BCI (Vidaurre et al., 2011). Performance feedback allows the BCI user to learn attentional strategies that may produce better results, while adaptive decoding methods can provide greater robustness to nonstationarities that exist in the signals across users and sessions.

In this work we explore different attentional paradigms for c-VEP BCI with a goal toward practical application. We evaluate gaze-directed overt attention, gaze-fixed covert attention, and gaze-constrained non-spatially selective overlapped target attention tasks. We compare several different spatial filter strategies for each and evaluate different confidence threshold mechanisms. We then compare an adaptive online classification method to a static one, and present a performance analysis method that takes into consideration the effect online trial rejection has on traditional classification accuracy for practical use considerations.

The rest of this chapter is arranged as follows. Section 2 describes the experimental design and decoding methods. Section 3 presents the results of the three c-VEP task experiments. Section 4 compares the methods and results to previous work,

comments upon the sources of error, and discusses key findings from the experiments. Finally, Section 5 concludes the chapter with a recap of the presented work and suggested future directions to investigate.

## 2.2. Methods and Materials

### 2.2.1. Data Acquisition

Eight subjects (2 females, aged 21-38) were recruited to perform the experiments, with data collected over a single session. All gave informed consent for the study, which had been approved by the Boston University Institutional Review Board. Two subjects had prior BCI experience, though none had experience with c-VEP BCI. All subjects had normal or corrected to normal vision, and none reported a history of epilepsy or indicated sensitivity to rapidly flicking lights.

EEG was recorded using the Enobio 8 (Neuroelectrics, Barcelona, Spain), an eight-channel wireless EEG recording system. Electrodes were placed at locations PO7, O1, Oz, O2, PO8, PO3, Pz, and PO4 according to the 10-20 international system with reference CMS and DRL electrodes placed over the right mastoid. EEG was digitized at 500Hz and transmitted via a Bluetooth connection to the BCI computer, where it passed through the Neuroelectrics NIC software to the Unlock application and saved to disk.

The experimental sessions were conducted in in an office-like environment lit with fluorescent ceiling lights. Subjects were seated in a comfortable chair approximately 70cm from an LCD computer monitor with a display resolution of 1920x1080 pixels operating at a 120Hz refresh rate. A Tobii EyeX eye tracker mounted just under the

screen was used to track the subjects' gaze during trials to ensure proper gaze fixation. Prior to beginning the experiment, subjects created eye tracking calibration profiles. Individual alpha frequency data was also obtained by instructing subjects to close their eyes three times for approximately five seconds at a time.

Data collection and task presentation were performed on the same computer using the Unlock software. EEG and gaze data were streamed into Unlock using the Lab Streaming Layer (LSL) library (available: https://github.com/sccn/labstreaminglayer). Event markers, such as the start of a c-VEP presentation cycle, were generated in software by Unlock and synchronized with the EEG and eye gaze data streams using the relative timestamps generated by LSL.

### 2.2.2. Experimental Design

The experiment was split into three phases: training, testing, and application control. In the training and testing phases, three different c-VEP tasks were performed: overt, covert, and overlapped. For the overt task, subjects were directed to gaze at and attend directly to a specific flickering target. In the covert task, subjects gazed at a central fixation point while attending to a specific target in their peripheral vision. Finally, the overlapped task had subjects gazing in the vicinity of a fixation target while attending to one of two overlapping checkerboard patterns that filled the entire screen. The application control phase was based on the overt task, with full procedure and results described in Chapter 4.

All tasks involved attending to stimuli flickering according to one of four possible 31-bit m-sequence-based patterns (Figure 2.1). A one in the sequence corresponded to the stimulus being on, or visible, while a zero corresponded to the stimulus being off, or hidden. The sequence progressed at a rate of 30Hz, requiring 1.034 seconds to complete one full presentation of the pattern.

The four m-sequences were the same as in (Nezamfar et al., 2011) and were chosen to have near-zero Pearson correlation coefficients:

$$\boldsymbol{P} = \begin{pmatrix} 1 & -0.033 & -0.033 & -0.044 \\ -0.033 & 1 & -0.033 & -0.044 \\ -0.033 & -0.033 & 1 & -0.044 \\ -0.044 & -0.044 & -0.044 & 1 \end{pmatrix}$$

The correlation coefficient matrix $\boldsymbol{P}$ of a sample matrix $\boldsymbol{X}$, where each row is a different variable and each column is an observation, is defined as

$$\boldsymbol{\rho_{ij}} = \frac{\boldsymbol{C_{ij}}}{\sqrt{\boldsymbol{C_{ii}C_{jj}}}} \qquad (\mathbf{2.1})$$

where $\boldsymbol{C_{ij}}$ is the covariance between the $\boldsymbol{i}^{\text{th}}$ and $\boldsymbol{j}^{\text{th}}$ rows of $\boldsymbol{X}$.

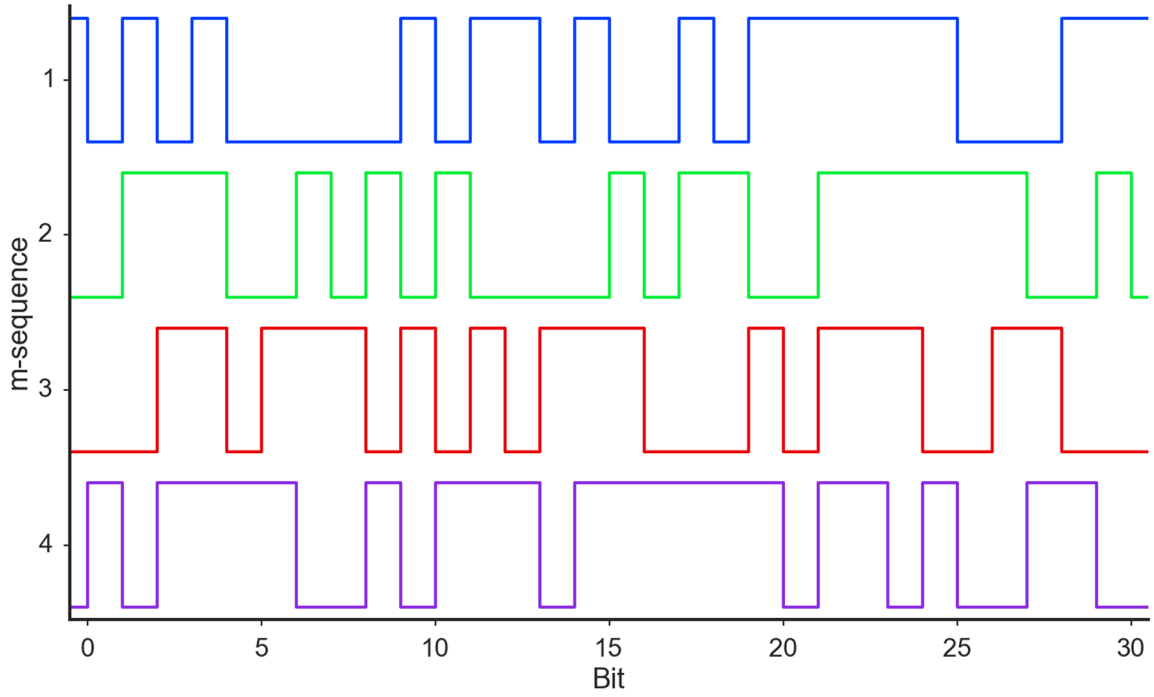


**Figure 2.1. M-sequence flicker patterns.**
In the overt and covert trials, m-sequences 1, 2, 3, and 4 were the basis for the flicker patterns presented by the up, down, left, and right targets, respectively. In the overlapped trials, m-sequences 2 and 3 were used for the green and magenta targets, respectively.

In both the overt and covert attention tasks, four white squares, 180x180 pixels in size (4.7cm, 3.86° visual angle), were centered 360 pixels (9.2cm, 5.81° visual angle to the inner edge) above, below, to the left, and to the right, respectively, from the center of the screen in front of a black background. This arrangement was chosen to reserve sufficient room in the center of the screen for the user application workspace.

The subject was instructed, through an onscreen prompt in the form of an arrow and fixation indicator (a cross) to look at and directly attend to the indicated target. An additional prompt had them look only at the center fixation point as a null-class reference.

The covert task was identical to the overt task, except that subjects were instructed to maintain gaze fixation at the center of the screen while only attending to the cued stimulus in their periphery.

The third task evaluated a strategy for non-spatially selective BCI. The entire screen was filled with overlapping green-clear and clear-magenta checkerboard patterns over a black background. The individual checkerboard tiles were square with side lengths of 108 pixels (2.84cm, 2.33° visual angle), or one-tenth the height of the screen. The colors had their alpha channel set to 0.75, providing some level of transparency. The green tiles flickered according to one m-sequence pattern while the magenta ones flickered according to another. The green tiles were shifted six pixels down and to the right while the magenta tiles were shifted six pixels up and to the left for a total offset of twelve pixels from each other, which, combined with the transparency effect, helped to create a sense that the checkerboards were on different depth planes, with the magenta tiles appearing slightly in front of or on top of the green tiles. The subject was directed to gaze in the vicinity of one of five different fixation points and attend to the green tiles, the magenta tiles, or just the fixation point.

The color, density, and offsets of the checkerboards were determined through a pilot study of one subject, with the above described combination of properties producing the greatest classification accuracy. Green and magenta are complementary colors in the RGB additive color model used by computer screens and provide high contrast while reinforcing brightness. Other complementary or opponent colors, such as red-cyan, red-

blue, and blue-yellow, were evaluated but did not perform as well and were reported to not be as distinct from one another as the green-magenta combination.

### 2.2.3. Procedure

Both the training and testing portions of the experiment followed the same basic trial layout (Figure 2.2). The different task paradigms were presented in blocks. At the start of each block, the subject was directed to press the space bar on the keyboard to start the block. The name of the task was presented for 2.0s then the trials began. Each trial started with a cue to the attentional target that lasted 0.5s, followed by a preparation period in which only the fixation point was visible for another 0.5s. Next came the actual task period when the stimuli were active on the screen – the number of stimuli and duration of flicker were determined by the particular phase of the experiment. After the task period ended, a feedback period occurred lasting 0.5s in which a visual indicator of the result was displayed. Finally, a rest period occurred containing no visual display and lasting another 0.5s.

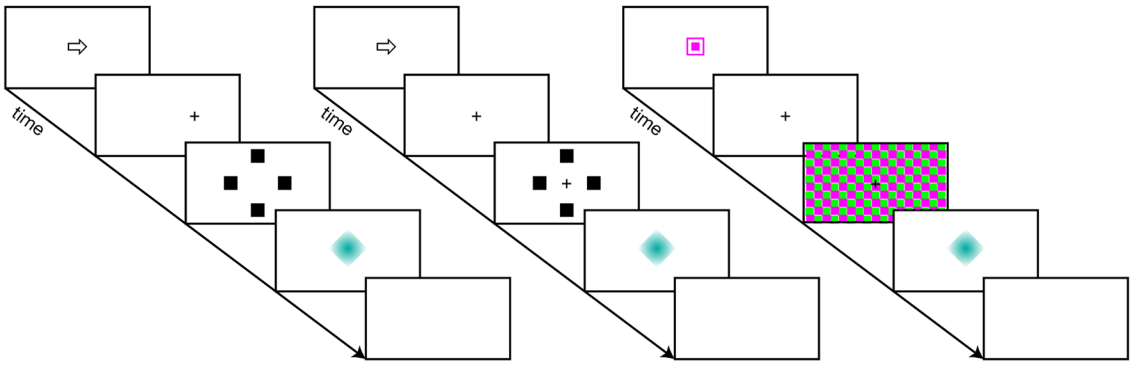


**Figure 2.2. Procedure schematic for the three c-VEP tasks.**
The overt (left), covert (center), and overlapped (right) tasks all followed the same general presentation pattern of cue, preparation, stimulation, feedback, and rest. While the timings of some segments varied between training and testing phases, they were the same across all paradigms.

The decoding method used relied on template matching, so an initial training phase was required to collect and build the m-sequence response templates for each subject before moving on to the testing phase. All subjects performed training in the same order: first the overt task, then the covert task, and finally the overlapping task.

### 2.2.3.1. Training

At the start, each potential target was presented twice and in isolation for 5.5s. An additional two null class cases were also presented, in which all targets were active while the subject gazed at the fixation point but was asked to not attend to anything in particular or "zone out". The order of the target presentation was randomized and no feedback was provided during this initial phase. In the case of the overlapped task, the subject was told that they did not have to maintain fixed gaze on the fixation point but maintain gaze in the near vicinity.

In the second training phase, all targets were present and active on the screen. Each target, including the null target, was cued four times in random order. Each trial lasted 5.5s. In the overt and covert trials, an additional feedback gradient image was placed near the intended target (Figure 2.3). The brightness of this gradient image changed after each full sequence cycle, or about every 1.034s, to reflect the relative strength of the online decoding attempts. The brighter the gradient became, the more highly correlated the last trial was with the current template for that target. Brightness was determined by the following equation:

$$\boldsymbol{b} = \left\lfloor \frac{\mathbf{255}}{\mathbf{1} + \boldsymbol{e}^{-\mathbf{5}(\|\boldsymbol{\rho}\| - \mathbf{0.15})}} \right\rfloor \qquad (\mathbf{2.2})$$

where $\boldsymbol{b}$ is an integer in [0, 255) and $\boldsymbol{\rho}$ is the correlation coefficient between the trial and the template for the cued target. As the stimulus filled the entire screen during the overlapped trials, a large central feedback gradient image was presented at the end of the trial instead.

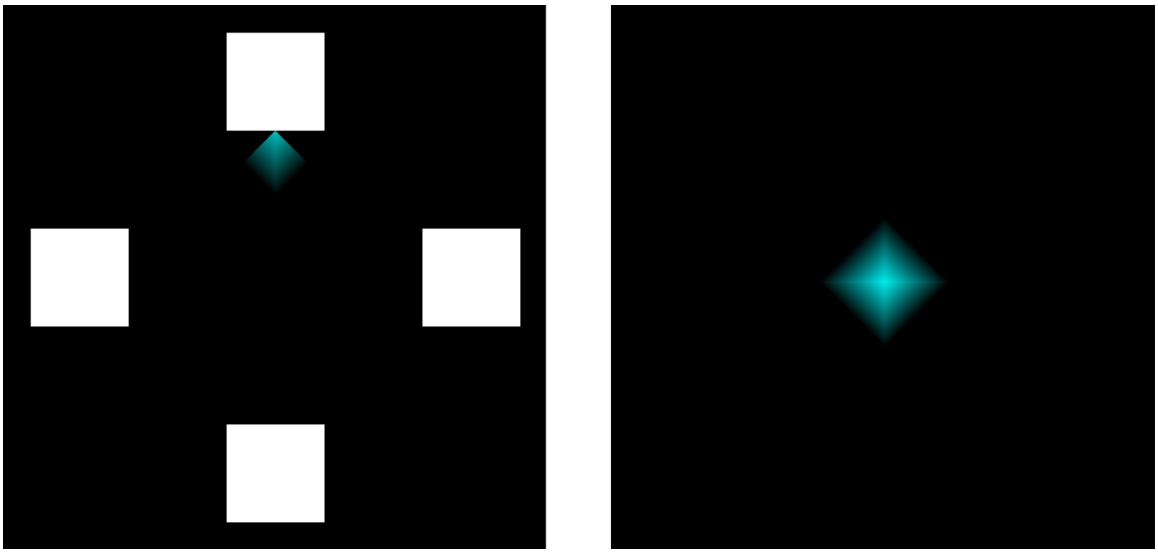

**Figure 2.3. Feedback gradients.**
Two types of feedback were provided to subjects. During the training phase trials of the overt and covert tasks, the colored feedback indicator appeared on the inner edge of the cued stimulus (left). A large, centrally placed feedback indicator appeared immediately proceeding the training phase trials of the overlapped task and all task trials of the testing phase (right).

#### 2.2.3.2. Testing

After completing the training phase, subjects then performed the testing phase of the experiment. Here, subjects were presented with three blocks of trials per task for a total of nine blocks. The blocks were presented in such a way that no paradigm would appear three times in a row and no subject received the same block order as any other. The presentation order was counterbalanced across all subjects so that the distribution of task block occurrences was roughly equal for a given order position.

A test block consisted of ten cues per target, including the null target, presented for 3.3s in order to capture three full m-sequence cycles. No feedback gradient appeared during trials, but instead appeared immediately afterward as a centered gradient image as

in the second overlapped training phase (Figure 2.4). In the covert trials, gaze fixation on the center of the screen was enforced through eye tracking. If the subject's gaze was detected to move greater than 1.8° from center, the trial ended with a "bad gaze" feedback indicator and the trial was repeated. The overlapped trials also moved the fixation point to one of five locations: centered and the for corners, placed at relative screen positions of (0.5, 0.5), (0.2, 0.2), (0.2, 0.8), (0.8, 0.2), and (0.8, 0.8), respectively.

In order to incentivize subjects to attend to the targets, an oddball stimulus would flicker during non-null target cues in approximately 10% of the trials. The oddball stimulus would appear randomly between 0.5s and 1.0s into the trial and remain visible for 0.25s. In the overt and covert trials, the oddball was white, the same size as the target stimuli and appeared over the cued target. In the overlapped trials, the oddball was the color of the cued target, the size of a single tile, and appeared over one of the appropriately colored tiles near the directed gaze point. Subjects were instructed to press the space bar when they saw the oddball stimulus appear. Their accuracy at detecting the oddballs was confirmed at the end of the testing phase.

Before the start of the experiment proper, subjects were shown practice demonstration versions of both the training and testing procedures for all three tasks, during which the instructions for the tasks were given. This ensured familiarity with the tasks and allowed for any questions regarding the procedure to be answered.

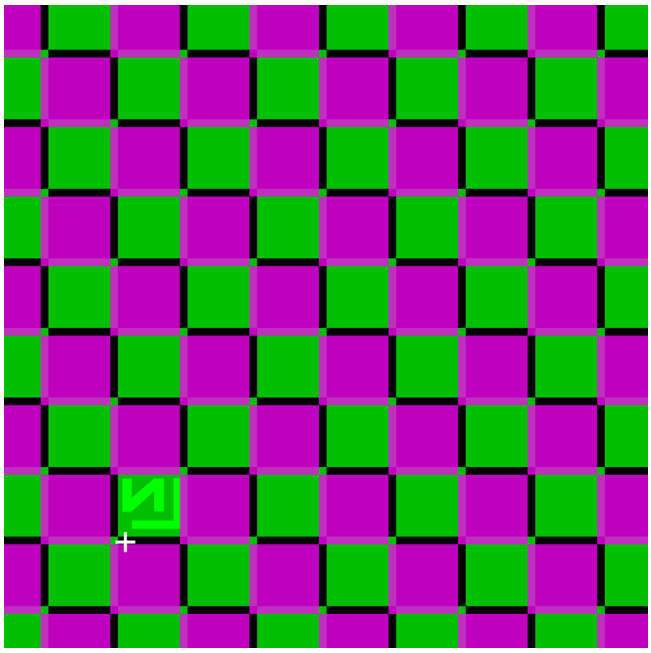

**Figure 2.4. Oddball stimulus during an overlapped trial.**
This is part of a screen capture taken the moment when the oddball stimulus is visible during an overlapped task trial.

### 2.2.4. Cognitive Workload

The NASA Task Load Index (NASA-TLX) (Hart and Staveland, 1988) survey provides a quantitative measure of relative cognitive workload experienced during execution of a particular task or set of related tasks. It consists of two parts. First, subjects rate six factors: mental demand, physical demand, temporal demand, performance, frustration, and effort on a 20-point scale for each task performed. Each point of the scale is worth five points. Second, the subject performs a pair-wise comparison of all factors, for a total of 15 comparisons, selecting which of the two factors contributed more to the experienced workload. This comparison is done once for the entire set of tasks. The number of times each factor was selected is summed and then divided by 15 to compute

that subject's personal weights for each factor. The cognitive workload score for each task is then the weighted average of the factors, producing a value between 0-100, with higher values indicating greater cognitive workload.

The pen and paper version of the NASA-TLX survey was administered to the subjects after completing the testing phase of the experiment. Each subject completed the survey for the overt, covert, and overlapped tasks, in that order. They then performed the factor comparison portion of the survey, with the pair-wise comparisons presented randomly.

### 2.2.5. Analysis

#### 2.2.5.1. Signal Preprocessing

First, each channel was adaptively demeaned using an exponential weighted moving average filter according to the following formula:

$$\boldsymbol{\mu_{i,0} = x_{i,0}} \tag{2.3}$$

$$\boldsymbol{\mu_{i,t} = \alpha x_{i,t} + (1-\alpha)\mu_{i,t-1},\ \ t > 0} \tag{2.4}$$

$$\boldsymbol{y_{i,t} = x_{i,t} - \mu_{i,t}} \tag{2.5}$$

where $\boldsymbol{i}$ corresponds to the $\boldsymbol{i^{th}}$ channel, $\boldsymbol{x}$ is the raw sample, $\boldsymbol{\mu}$ is the adaptive mean, $\boldsymbol{\alpha}$ is the smoothing factor, and $\boldsymbol{y}$ is the demeaned sample. Here, $\boldsymbol{\alpha}$ = 0.05 following (Vidaurre et al., 2011). This has the effect of acting as a high-pass filter.

EEG data related to task trials were detected and isolated by the presence of markers in a separate channel inserted via software. The trial data was extracted and resampled to 64 samples using the FFT method. The choice to resample rather than apply

a low-pass filter was two-fold. First it significantly reduced the total number of features from 517 to 64 per channel that had to be considered by downstream feature extraction and classification processes. Second, the actual number of samples in a trial could vary slightly due to temporal jitter that caused slight desynchronization between presentation and data acquisition, so resampling forced all trials to have the same number of samples which also aided in the downstream processing. The number of samples was set at 64 via analysis of pilot data from two subjects. Given a trial length of 1.034s this produced an effective new sample rate of approximately 62 Hz which placed it above the Nyquist rate needed for the 30Hz display frequency used by the stimuli.

#### 2.2.5.2. Feature Extraction

The features used for building average response templates came from the spatially-filtered EEG recorded during each trial. Seven different spatial filters associated with EEG-based BCI were evaluated in offline analysis (Table 2.1). First was a single channel filter that only used the signal from Oz, which has been shown to be effective for overtly attended c-VEP (Nezamfar et al., 2011). Second was a bipolar channel filter that used the difference between Pz and Oz, previously used in an adaptive SSVEP study (though they used POz instead of Pz) (Fernandez-Vargas et al., 2013). Next was a common average reference (CAR) filter along with three different discrete Laplacian filters centered on Oz. Whereas the bipolar channel represents the first spatial derivative, Laplacian channel filters represent the second spatial derivative. Both CAR and Laplacian filters have been used in motor imagery tasks (McFarland et al., 1997), but not

typically in VEP studies. The three Laplacian filters evaluated were a 1D Small Laplacian using O1, Oz, and O2; a 1D Large Laplacian using PO7, Oz, and PO8; and a 2D Large Laplacian using PO7, Oz, PO8, and Pz. Normally a 2D Laplacian filter would have five channels, but as the the center channel Oz was already at the boundary of the electrode montage, a truncated version was used. Finally, filters were produced on a per-subject, per-target bases using Canonical Correlation Analysis (CCA). CCA-based filters have proven to be quite effective in c-VEP BCI (Bin et al., 2011; Spüler et al., 2013; Waytowich and Krusienski, 2015).

CCA produces a set of column vectors $\boldsymbol{w_x}$ and $\boldsymbol{w_y}$ that maximize the correlation between two matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ via:

$$\max_{w_x, w_y} \rho(x, y) = \frac{w_x^T X Y^T w_y}{\sqrt{w_x^T X X^T w_x \cdot w_y^T Y Y^T w_y}} \qquad (2.6)$$

In the context of BCI classification, $\boldsymbol{X}$ is the preprocessed EEG signal from all channels and $\boldsymbol{Y}$ is a reference signal of interest. In this work, $\boldsymbol{X}$ is created by concatenating m n-channel trials of length k in the evaluation data set together producing a matrix of shape (n, m x k). $\boldsymbol{Y}$ is obtained by taking the median average channel response across the same m trials, then repeating that result m times to produce a matrix of shape (1, m x k). The resulting vector $\boldsymbol{w_x}$ will then have shape (n, 1) and act as the spatial filter across channels. This procedure is repeated to produce a separate filter for each target in the task.

| Filter Name | Label | Weights |
|---|---|---|
| Single | Oz | [0, 0, 1, 0, 0, 0, 0, 0] |
| Bipolar | Pz-Oz | [0, 0, -1, 0, 0, 0, 1, 0] |
| Common Average Reference | CAR | [1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8] |
| 1D Small Laplacian | 1DSLAP | [0, 1, -2, 1, 0 ,0 ,0 ,0] |
| 1D Large Laplacian | 1DLLAP | [1, 0, -2, 0, 1, 0, 0, 0] |
| 2D Large Lalpacian | 2DLLAP | [1, 0, -3, 0, 1, 0, 1, 0] |
| Canonical Correlation Analysis | CCA | Subject and target specific |

**Table 2.1. c-VEP spatial filters.**

#### 2.2.5.3. Classification

Existing c-VEP methods use template matching for classification purposes. This is based on the observation that the sequence and magnitude of VEPs elicited by the flicker pattern is generally consistent for each presentation cycle. A template is created from a training data set and then used as the basis for comparison during classification of a testing data set. Template generation methods range from simple averaging (Bin et al, 2011) to more advanced machine learning approaches, such as using one-class support vector machines (Spüler et al., 2012). In this work, templates are created by taking the median average feature vector of the training set produced after preprocessing and feature extraction as in (Nezamfar et al., 2011).

In phase-based c-VEP methods, only a single template is required as all target stimuli will have the same pattern, just shifted by a predetermined phase. Classification is then accomplished by computing the offset of the maximum linear cross-correlation value between the test sample and the template, then matching that to the target stimulus associated with that offset. In this work, each stimulus had its own distinct m-sequence, so each target stimulus required its own template. The linear correlation coefficient of the test sample with each template is computed, and the template that has the highest correlation is chosen as the determined class.

A weakness of the median average template generation and correlation coefficient classification methods are that they are both highly susceptible to temporal drift. Any jitter in the alignment of the EEG signal with the stimulus presentation can introduce offsets that can then greatly impact the sharpness of the templates or the determined correlation coefficients. Temporal jitter is a known problem with high temporal precision presentation systems that use multi-tasking operating systems (Straw, 2008) and one reason BCI display systems turn to microcontroller-controlled LEDs or hardware-based triggers using photodiodes. During pilot data analysis, it was determined that this offset was occurring in data collected through the Unlock software. A software-based solution to the jitter problem was employed by adjusting the classification method to assume slight offsets occur. Correlation coefficients for all templates at five different phase shifts of the test signal, in this case rolling the feature vector by 0, 1, 2, 3, and 4 points forward, respectively. The template with the greatest score amongst all templates and signal shifts was then selected as the classified target.

In addition to fixed-template classifiers evaluated offline, an adaptive classifier was used online during the training and testing phases to drive the feedback indicators. After each single-cycle trial, the template for the cued target was updated by adding the extracted features from the trial data to a buffer and recomputing the median average template. This was performed regardless of whether the decoder correctly classified the trial or not. During training, the template was computed from a growing number of samples, reaching a total of 30 per target by the end of the phase. The set of training samples was carried over to the testing phase, where each new trial replaced the oldest from the set, resulting in the template being computed from the 30 most recent target trials. The 2DLLAP spatial filter was chosen for this online adaptive template method based on initial pilot studies.

#### 2.2.5.4. Confidence Thresholding

In order to compensate for bad signals or potentially identify when a user was not actually paying attention to any target, a confidence threshold was added to determine if the classifier's result was acceptable. Trials that did not meet this confidence threshold were rejected. From an online BCI perspective, this meant that the decoder returned a null result, or "no decision," for that trial.

Some classifiers, such as LDA, have classification probabilities built into them which could be used to determine confidence. For the correlation-based template matching decoder used here, another strategy was required. In order to evaluate confidence metric candidates, templates were generated from the training phase and used

to classify the trials from the testing phase. All template correlation scores and estimated alpha band power over Oz for each trial were recorded and partitioned based on whether the trial classification was correct or not. Four different potential metrics were evaluated: the predicted target score ("winner"), the difference between the predicted target and the next closest target ("diff2"), the z-score of the predicted target computed from all target scores ("zscore"), and the relative alpha band power ("alpha").

#### 2.2.5.5. Reliability

When reporting on the performance of an online BCI task, most studies simply list the overall classification accuracy and likely the associated ITR of the decoding method. ITR is flawed as a metric for practical online BCI evaluation as it only considers raw information throughput under certain preconditions (Yuan et al., 2013), many or all of which may not hold in more user-centric BCI designs that employ adaptive or asynchronous methods. Also, with the exception of very high accuracies which imply continual success, raw classification accuracy fails to capture how that performance was spread across the duration of the task.

The challenge of quantifying BCI performance is further compounded by decoders that employ trial rejection, such as the confidence threshold described in this work. The ITR calculation does not have a way to adequately incorporate these null results, though from a user standpoint, the occasional "no decision" would be preferable to an erroneous one. One approach is to treat these null results the same as trials rejected due to artifact contamination and not count them in the final accuracy calculation.

However, a decoder that achieves a 100% classification accuracy by omitting 90% of all trials under consideration due to rejection is not usable either, so raw classification accuracy is also not enough.

In order to address these problem, an analysis method was developed that attempts to quantify the relative performance of a BCI with online trial rejection over time. This metric, termed reliability, has a value in [-1,1] that is recomputed after every trial, with values of 1, -1, and 0 corresponding to reliably accurate, reliably inaccurate, and unreliable, respectively.

Reliability was computed using the following algorithm. First, $\boldsymbol{a}$, the expected accuracy due to chance for the task, was determined, e.g. $\boldsymbol{a} = 0.5$ for a two-choice task, $\boldsymbol{a} = 0.25$ for a four-choice task. Then, starting with the first decoded trial, and for each trial thereafter, the trial accuracy was scored as +1 for a correct classification and -1 for an incorrect classification. Null results score based on how many consecutive rejections have occurred according to the following:

$$\boldsymbol{s_0 = max(-t_0 a, -1)} \qquad \boldsymbol{(2.7)}$$

where $\boldsymbol{t_0}$ is the number of consecutive trials prior to the current one that also had a rejected trial. Negative scores were assigned to errors and null results because, from a practical application standpoint, an error is frequently costly, requiring additional corrective actions to achieve the desired outcome. The fractional and increasingly negative score for a null result was to reflect that, while the occasional rejected trial is acceptable and even desirable, too many continuous rejections would have a negative impact on usability. The score was added to a running tally, $\boldsymbol{s}$. The pre-scaled reliability,

$\hat{\boldsymbol{r}}$, was obtained by dividing the cumulative score by the $\boldsymbol{t}$ number of trials observed at that point

$$\hat{\boldsymbol{r}} = \frac{\boldsymbol{s}}{\boldsymbol{t}} \tag{2.8}$$

Next, let $\boldsymbol{c} = \mathbf{1} - \mathbf{2}/\boldsymbol{n}$ be an alignment factor, where $\boldsymbol{n}$ is the number of choices in the task. For $\boldsymbol{n} > \mathbf{2}$, the rescaled reliability score, $\boldsymbol{r}$, was then computed as

$$\boldsymbol{r} = \begin{cases} \dfrac{\hat{\boldsymbol{r}} + \boldsymbol{c}}{\boldsymbol{c}}, \hat{\boldsymbol{r}} \leq \boldsymbol{c} \\ \dfrac{\hat{\boldsymbol{r}} + \boldsymbol{c}}{\mathbf{1} + \boldsymbol{c}}, \hat{\boldsymbol{r}} > \boldsymbol{c} \end{cases} \tag{2.9}$$

This scaling method ensures that a reliability score of zero corresponds to the theoretical chance level for the BCI task, which can provide for quick relative comparison between tasks that have different numbers of targets. It does, however, have the effect of magnifying worse-than-chance behavior while compressing better-than-chance behavior.

Theoretically, reliability differs from ITR in a number of ways. First, ITR is an information theoretic approach that describes performance in terms of bits per choice. In order to do this, it treats the recorded classification accuracy as a binary probability distribution. Reliability, on the other hand, has neither the notion of bits of information nor having a probability measure, so it is able to incorporate dynamic weighting of each choice prediction to incorporate other potential factors such as usability. Second, both reliability and ITR peg chance level accuracy at zero. Doing worse than chance under ITR, however, produces a non-zero score. While this is intuitive in the sense of information, it can raise issues in simply comparing ITR numbers. For instance, the ITR

for a four choice task is approximately equal at 1% and 58% classification accuracy. To avoid this confusion, ITR is typically not reported for classification accuracy below chance. Third, reliability is bounded between [-1, 1], regardless of the number of choices or time of choice, making it a more qualitative metric for comparing usability of a variety of BCIs. ITR provides a more direct comparison for the theoretical decision throughput a particular BCI could achieve.

## 2.3. Results

### 2.3.1. Spatial Filters

The seven different spatial filters were evaluated offline using single-cycle trials collected from the testing phase. Null-class trials were ignored and no trial rejection due to artifact contamination was performed. Templates were evaluated using 5-fold cross validation with 66% of the trials (n=240 for overt and covert, n=120 for overlapped) used for training and the remaining 33% (n=120/n=60) used for validation. In the following figures, the color scale is centered around chance-level accuracy (0.25 for overt/covert, 0.5 for overlapped). Darker hues of red indicate greater than chance accuracy, while darker hues of blue indicate worse than chance accuracy.

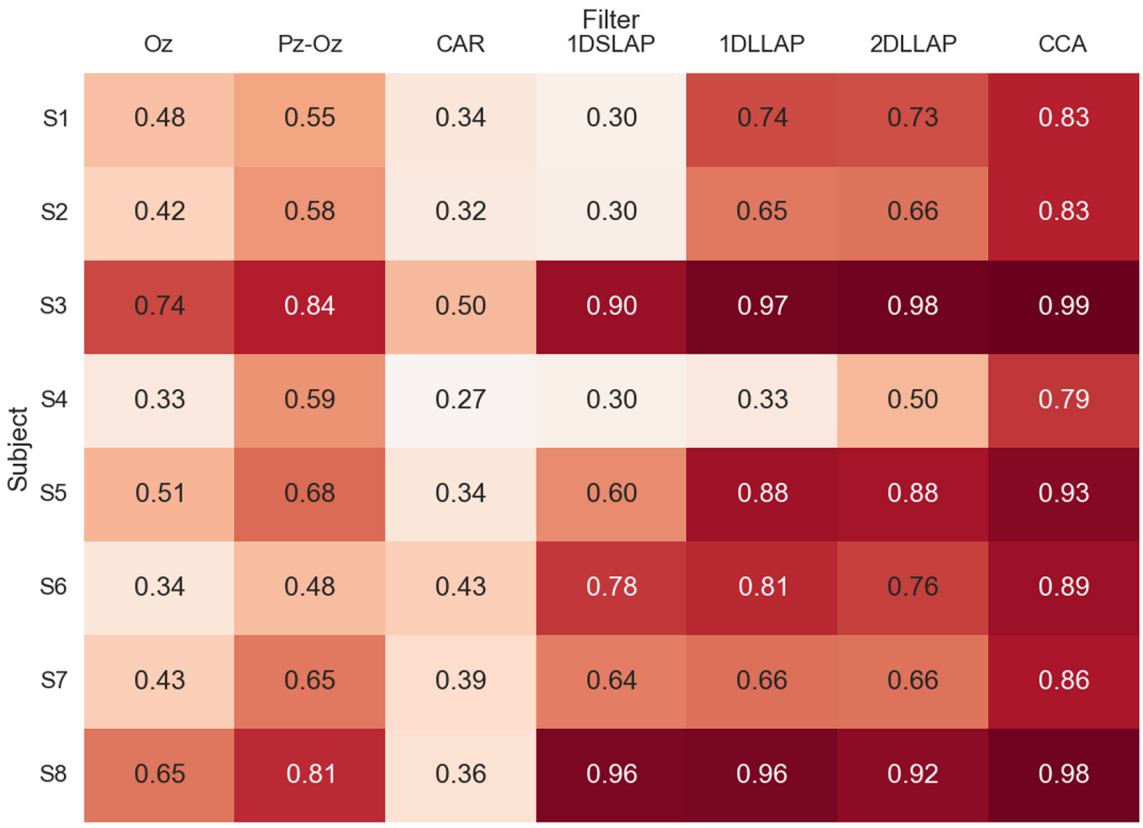


**Figure 2.5. Overt task classification accuracies for various spatial filters.**

Overt task performance (Figure 2.5) ranged from chance levels in some cases to near perfect performance in others. For the general filters, accuracy was typically higher the more differential channels that were incorporated into the filter. The CAR filter was the worst performing across all subjects. The user- and target-specific CCA filters outperformed all others, demonstrating a 20% increase over the next best performing option in two subjects (S4, S7). S4, in particular, has relatively poor performance across all general filters. This suggests noisy channels, either from Oz itself or from one of the others included in the filters. CCA's success in this case comes from its ability to effectively minimize the contribution of bad channels.

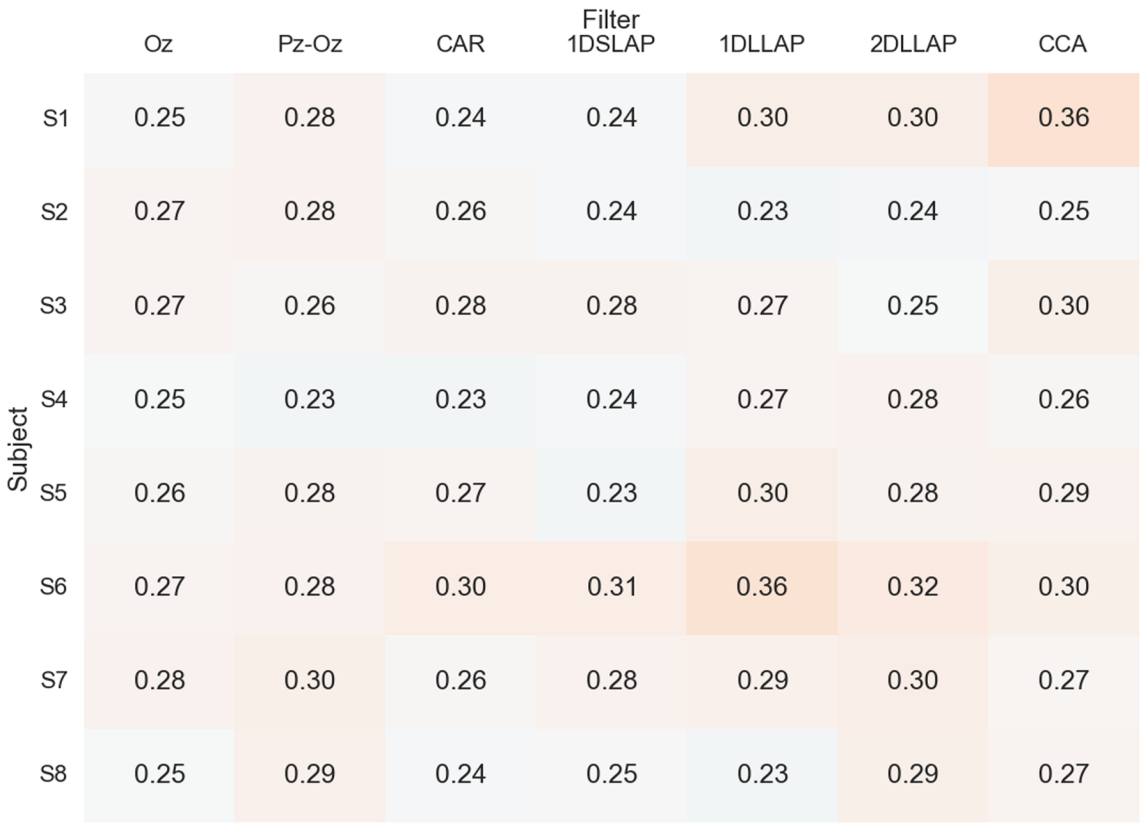


**Figure 2.6. Covert task classification accuracies for various spatial filters.**

Covert performance, on the other hand, was no better than chance for almost all subjects and filters (Figure 2.6). Only CCA for subject S1 ($p < 0.01$) and the Large Laplacian filters for subject S6 (1DLLAP, $p < 0.01$; 2DLLAP, $p < 0.05$) were able to achieve better than chance accuracies.

The overlapped task performance (Figure 2.7) appeared to be more dependent on the subject than on any particular filter. Half of the subjects (S2, S4, S5, S8) were unable to achieve significant performance over chance for any filter, while three subjects (S3, S6, S7) were able to cross the usability threshold of 70% (Kübler et al., 2004). Unlike in

the overt task, CCA did not appear to offer an advantage over the Large Laplacian methods.

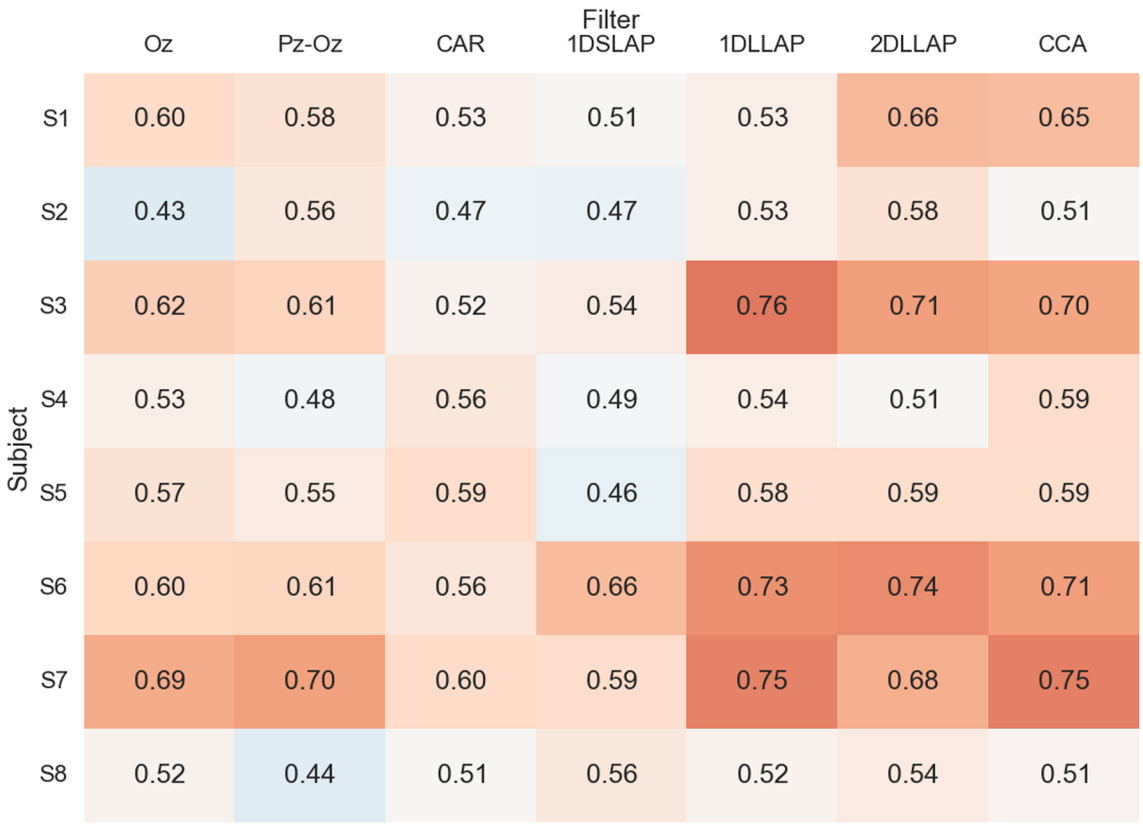


**Figure 2.7. Overlapped task classification accuracies for various spatial filters.**

In addition to computing accuracies, the ITR values of these results were also obtained. Notable is that the best ITR performance for the overt task (Figure 2.8) was an order of magnitude greater than that of the overlapped task (Figure 2.9), which itself was an order of magnitude greater than the covert task (Figure 2.10)


**Figure 2.8. Overt task ITR for various spatial filters.**


**Figure 2.9. Overlapped task ITR for various spatial filters.**

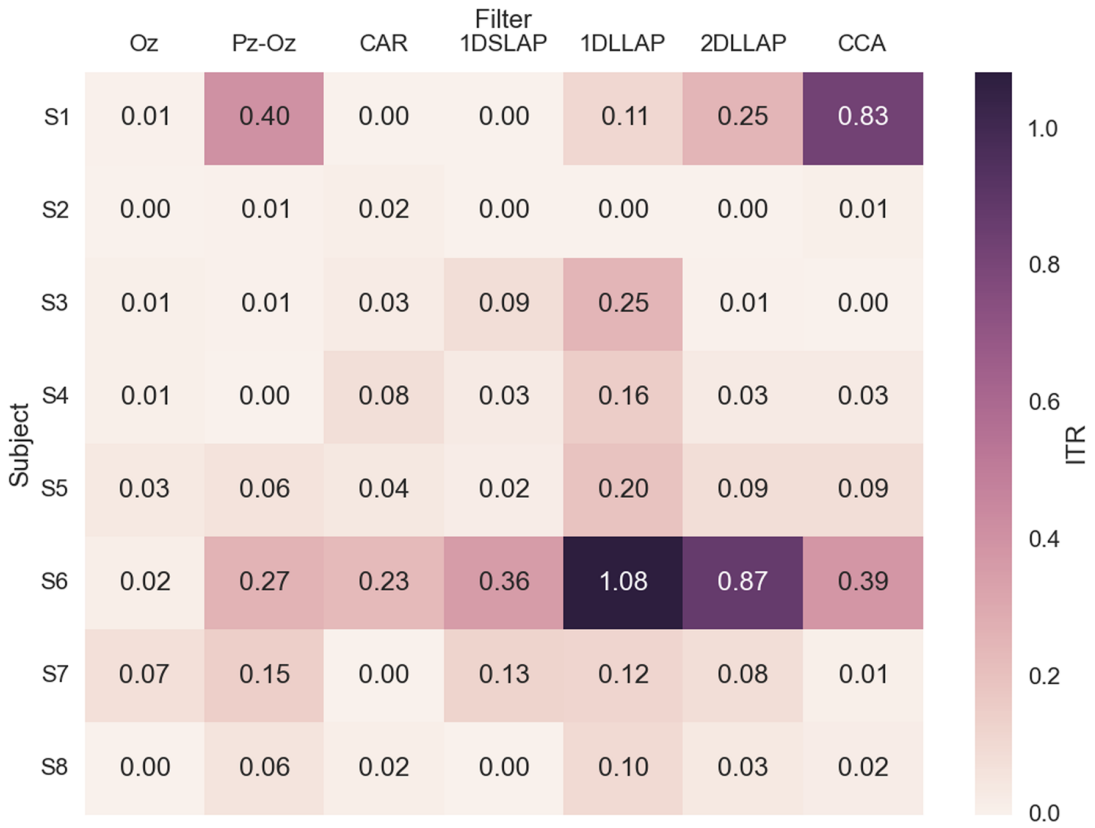


**Figure 2.10. Covert task ITR for various spatial filters.**

### 2.3.4. Filters and Templates

Example spatial filters and resultant templates generated via CCA from the training data are shown for two subjects, S3 and S1 in Figure 2.11 and Figure 2.12, respectively. CCA clearly picks out Oz as the most important channel for S3 in both the overt and overlapped tasks, regardless of target. It also exhibits spatially distinct filters for the covert task targets, with inverted weights between the left and right targets, and alternating emphasis placed on Pz vs Oz for the up and down targets, respectively. The

overt templates are sufficiently uncorrelated with each other, while the overlapped templates are almost all identical, contributing to the relatively poor performance in that task. Despite distinct and somewhat uncorrelated templates for the covert task, performance is still very low, suggesting that there may not be enough consistency in single-cycle trials.

The spatial filters and templates for S1 are presented in comparison to show that not all subjects had clean results. Here, CCA has identified that the O1 channel was bad, as it is weighted near zero across all three tasks. Indeed, analysis of S1's channel variance showed O1 to be quite noisy compared to the others channels.

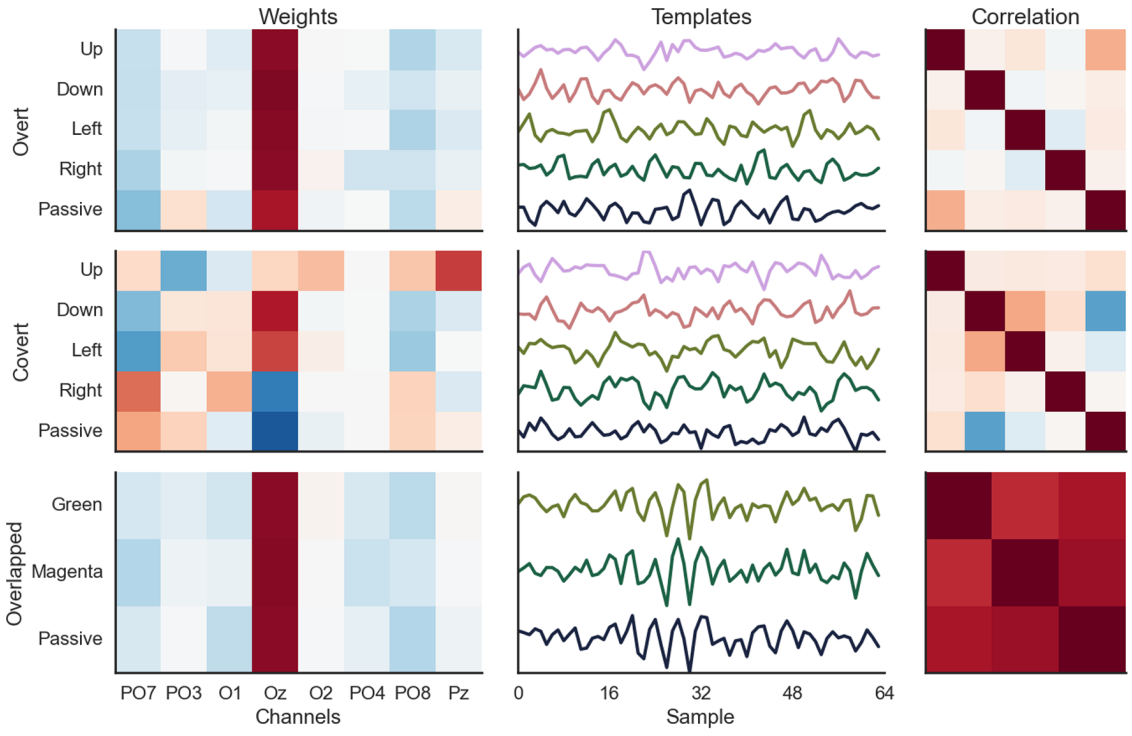


**Figure 2.11. Training results using CCA for subject S3.**
The spatial filters (left), resultant templates (center), and template correlations (right) produced using CCA for subject S3. The color scale ranges from -1 (dark blue) to 0 (light gray) to 1 (dark red). Templates are offset vertically to align with their associated target weights.

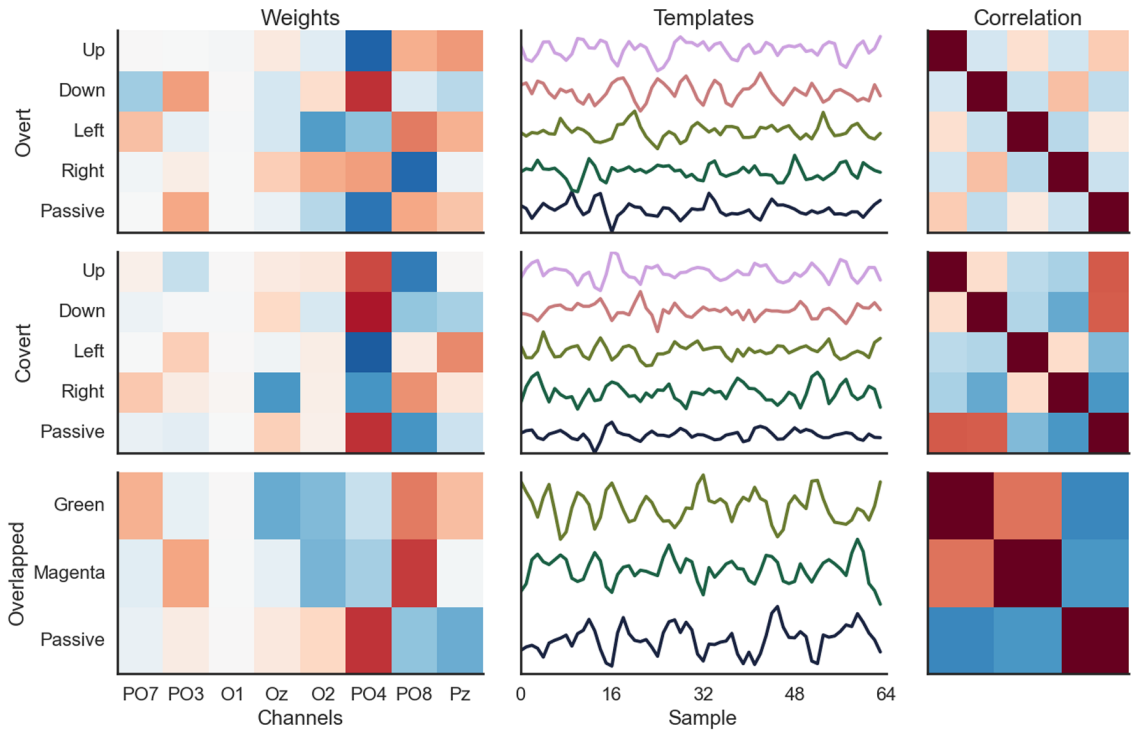


**Figure 2.12. Training results using CCA for subject S1.**

### 2.3.4. Confidence Metrics

Based on the previous analysis, CCA was chosen as the spatial filter to evaluate further for confidence thresholding. Only the overt task produced any distinct differences between the four metrics evaluated (Figure 2.13). The most discriminable metric is "winner", which reaffirms pilot study results that suggest the score of the predicted class alone is sufficient for the correlation-based classifier. There is also a noticeable difference in relative alpha band power, with good trials exhibiting greater alpha suppression indicating users were more attentive during these trials.

None of the metrics produced significant differences between good and bad trials in the covert or overlapped tasks. This suggests that, unsurprisingly, thresholding based on the classifier score will only benefit classifiers that are already operating at a reasonably high level of accuracy.

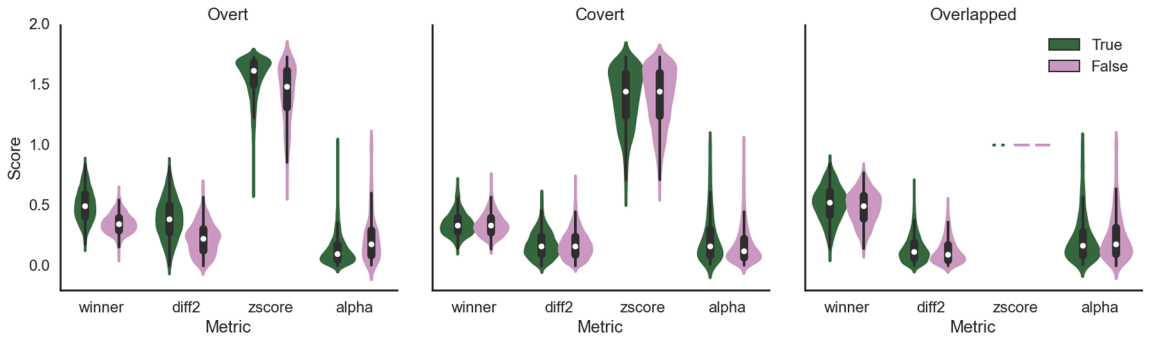


**Figure 2.13. Comparison of potential confidence threshold metrics.**
The dark violin plots reflect the correctly classified (true) trials across all subjects while the light plots are from the incorrect (false) trials. As overlapped trials only have two evaluated templates, z-scores were always equal to 1.

### 2.3.5. Adaptive Online Performance

Online performance for adaptive and fixed 2DLLAP- and CCA-based decoders were evaluated (Figure 2.14 and Figure 2.15, respectively). All used a confidence threshold strategy of rejecting any trials with target correlation scores less than 0.3. The confidence threshold was only applied to the output of the overt task decoder, as earlier analysis indicated that it would have little to no beneficial effect on the covert and overlapped tasks.

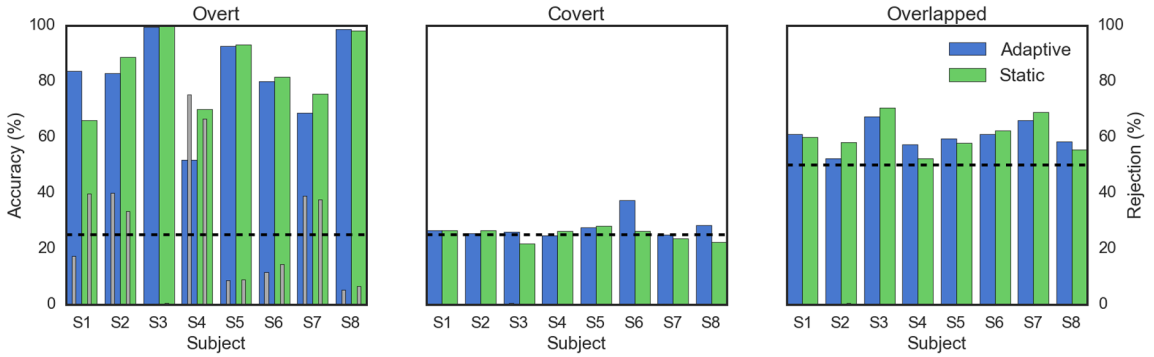


**Figure 2.14. Online c-VEP classification accuracies using 2DLLAP.**
The online classification accuracies across the three tasks were computed for both the adaptive templates (green) and static templates (orange) ignoring null class trials. The percent of trials rejected for failing the confidence threshold test for each method is indicated in gray. Accuracy due to chance for each task is indicated by the dashed lines.

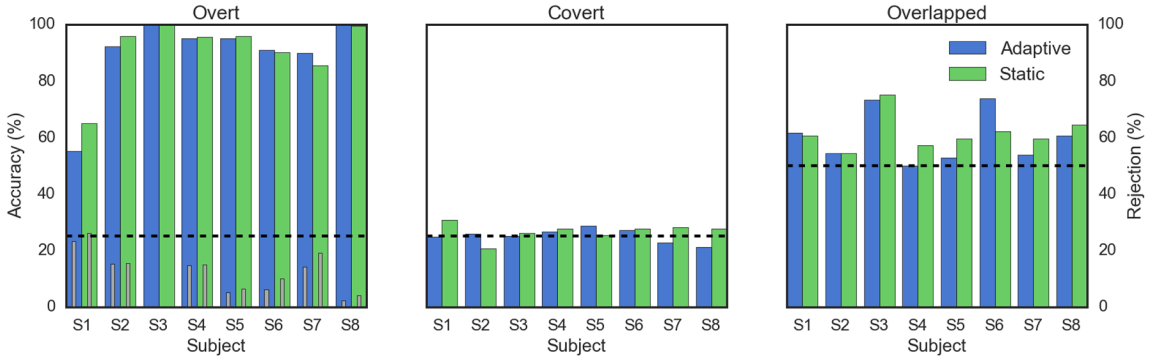


**Figure 2.15. Online c-VEP classification accuracies using CCA.**

Subjects S3 and S8 both performed exceptionally well on the overt task, so the choice of method or type of decoder had little effect. Generally, CCA demonstrated equal or better accuracy with significantly fewer rejected trials. S4 had a significant improvement, both in accuracy and reduction of rejected trials. S1, on the other hand, actually showed a significant decline in performance in the adaptive CCA decoder –

adaptive 2DLLAP was the best, followed by no difference in either static 2DLLAP or CCA, followed by adaptive CCA.

Covert task performance was flat at chance levels across the board for all decoder configurations, with the exception of S6 using adaptive 2DLLAP. It was not enough to raise the performance to usability level, but it does suggest some consistency was found in the trials. Similarly, overlapped task performance did not see much difference between the decoders other than adaptive CCA for S3 and S6.

Reliability scores were computed and plotted over time for both adaptive 2DLLAP and CCA decoders as well (Figure 2.16 and 2.17, respectively). As neither covert nor overlapped tasks incorporated trial rejection, their reliability score was a scaled version of raw classification accuracy.

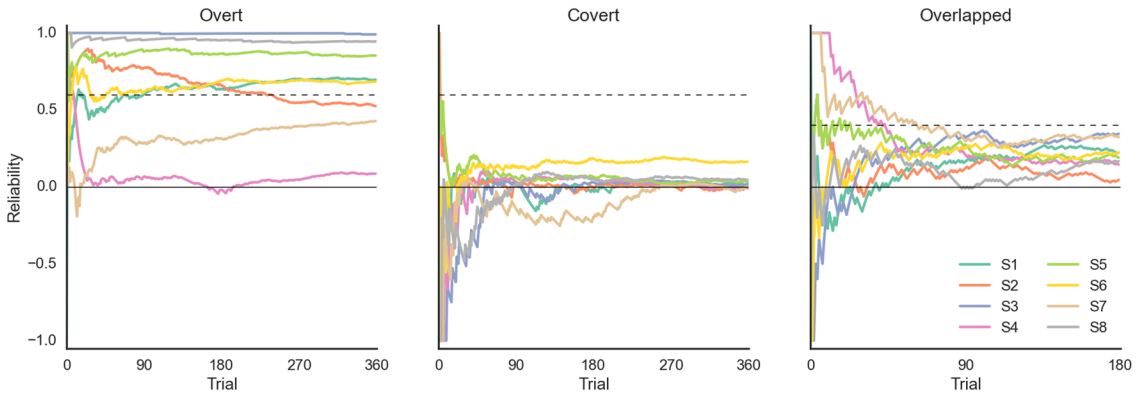


**Figure 2.16. Reliability over time of adaptive 2DLLAP.**
The reliability score was computed for each trial over all trials presented to the decoders. The dashed line corresponds to the scaled equivalent of the 70% usability threshold.

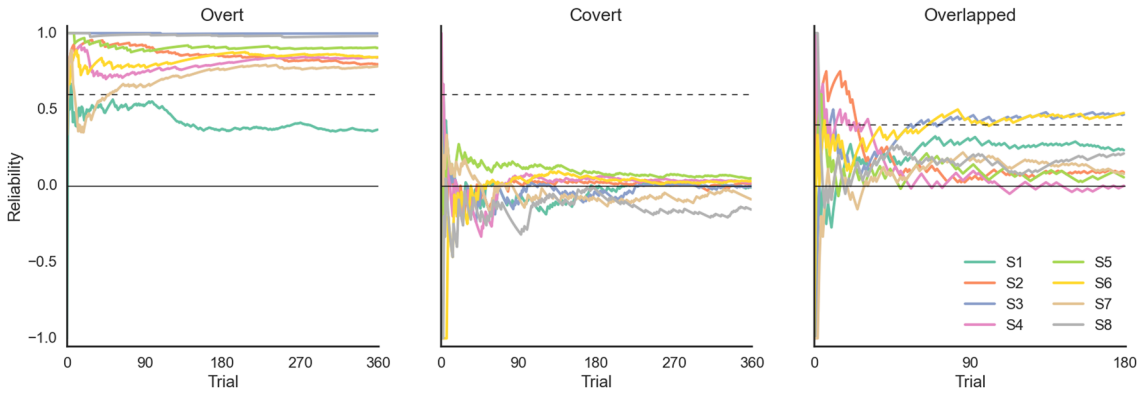


**Figure 2.17. Reliability over time of adaptive CCA.**

Using the reliability score over time, it is possible to see that S1's CCA performance on the overt task seems to suffer a major hiccup around trial 120, which happens to correspond to the start of a new block of trials. S4's 2DLLAP reliability on the overt task demonstrates how this analysis methods departs from just reporting classification and rejection rates. Using only classification accuracy, S4 would report 52% accuracy with a rejection rate of 75%. The reliability score indicates that this BCI would be no better than chance.

### 2.3.6. Cognitive Workload

The individual NASA-TLX components were averaged (Table 2.2) and the scaled cognitive workload scores were computed for each subject for each of the three tasks, with median scores of 33.00 (overt), 56.00 (covert), and 25.35 (overlapped). A Wilcoxon signed-rank test was performed on each pair of task responses. Cognitive workload experienced during the overt task was less than that during the covert task ($T = 0$, $P =$

0.01), while neither overt vs overlapped (T = 13, P = 0.87) nor covert vs overlapped (T = 6, P=0.09) were significantly different. Given the relatively small sample size, it is possible that significance between covert and overlapped task workload would be observed if more subjects were run.

The clear difference between overt and covert tasks can most likely be attributed to the added challenge of maintaining gaze fixation on one point while attending to the target. In addition to the more difficult task of attending to something peripherally instead of directly, healthy subjects must also suppress the desire to look directly at the target.

| | Overt | Covert | Overlapped | Contribution |
|---|---|---|---|---|
| Mental | 38.13 | 60.63 | 38.75 | 2.88 |
| Physical | 28.13 | 35.63 | 32.50 | 1.13 |
| Temporal | 35.63 | 40.00 | 34.38 | 1.38 |
| Performance | 25.63 | 40.00 | 29.38 | 4.25 |
| Effort | 46.25 | 68.75 | 45.00 | 4.13 |
| Frustration | 15.63 | 39.38 | 21.25 | 1.25 |

**Table 2.2. Average scores of individual NASA-TLX factors.**

### 2.4. Discussion

In this chapter, work toward implementation and evaluation of an adaptive c-VEP BCI for practical application was presented. Data was collected and analyzed from subjects participating in online-based gaze-dependent and gaze-independent tasks.

Subjects received relative qualitative feedback instead hit or miss indicators after trials to encourage them to find adapt their attentional strategies. Offline, common static spatial EEG filters were compared with the user- and target-adaptive CCA-based filters for use in the template matching decoder, with CCA broadly performing equal to or better than any other method. The only consistent contenders to CCA were the Large Laplacian filters, which tended to do slightly better for some users in the gaze-independent tasks. Next, a trial rejection mechanism based on confidence thresholding of the BCI classifier was developed. Four different metrics were considered, with the raw correlation coefficient of the predicted target showing the best discriminability between correct and incorrect trials. Third, adaptive template matching methods based on CCA and 2DLLAP were evaluated in simulated online BCI conditions, replaying the data collected from the subjects as if it were live. CCA again proved superior for overt tasks, but did not have a significant advantage over 2DLLAP in the gaze-independent tasks. Finally, a novel performance method, termed BCI reliability, was developed to provide a more meaningful representation of practical BCI performance when the BCI contains elements such as null results and adaptation.

### 2.4.1. Qualitative Online Feedback

It has been shown that offering only positive or qualitative feedback during online training trials can improve BCI performance, as subjects who feel like the BCI is working stay engaged with it. Both (Faller et al., 2012) and (Marchetti et al., 2013) provided "perfect feedback" to the users, letting them believe they were achieving 100% accuracy

in motor imagery and covert attention P300 tasks, respectively. In an adaptive SSVEP paradigm study (Fernandez-Vargas et al., 2013), users are given modulating auditory cues during training to indicate how well they are doing at the task. This work is similar in that it modulates visual cues designed in such a way as to be minimally distracting yet still salient. Unlike Fernandez-Vargas et al., the qualitative feedback is provided throughout training and testing sessions, rather than limited to just the training period. Also, by presenting a simple qualitative cue, i.e. relative brightness of a gradient image, the testing phase becomes more like a game than a test. This is especially important when collecting online environment trials of paradigms that may not have good working decoders at the time, such as the gaze-independent tasks. It provides a means to keep the subject engaged without discouraging them early on. While no formal irritation survey was conducted, informal queries about the tasks after the session found most subjects had little difficulty staying on task, with some even expressing confidence that they were getting better at the covert trials, despite the decoder itself operating effectively at chance levels.

### 2.4.2. Gaze-Dependent Task Performance

The performance of the overt task decoder was comparable to that reported by other gaze-dependent c-VEP studies. (Bin et al., 2011), (Spüler et al., 2012), and (Waytowich and Krusienski, 2015) all use phase-shifted versions of a single 63-bit m-sequence displayed at 60Hz and CCA-based spatial filters over 9, 32, and 16 channels, respectively. The first two demonstrated control of a 32-target speller, while the latter

investigated eight- and four-target setups arranged in an annulus. All reported average classification accuracies across all subjects in the high 90's. (Nezamfar et al., 2011) also used a four-choice task with four distinct, 31-bit m-sequences displayed at 30Hz and was able to report offline accuracy from single channels in the high 90's as well.

The slightly lower performances reported here, especially for subject S1, can be attributed to a few different sources of error. First and foremost was precise synchronization between the stimulus presentation and EEG data. Both the averaging method use to create templates and the linear correlation-based template matching decoder are highly susceptible to desynchronization. Attempts to correct these synchronization issues have all been in software thus far, which have partially addressed the problem but still seem to not fully compensate for it. Second, the EEG equipment used was generally effective but more susceptible to environmental noise given its portable design. Third, some subjects, such as S1, ended up having a bad channel which dramatically impacted the quality of the data. CCA was able to mitigate that to some extent but more advanced channel selection and weighting methods should be explored. Fourth, the environment the subjects performed the task in was intentionally not controlled for optimal visual BCI performance. Fluorescent overhead lights limited the full contrast available from the display while ambient noise and glossy, reflective display screens could be distracting. These sources of error largely stemmed from tradeoffs made in aiming for a BCI that could operate in more typical environments, rather than a highly controlled lab setting with stationary EEG equipment.

More fundamentally, the usage of four distinct m-sequence patterns instead of a single one that was phase-shifted introduced additional complexity into the decoding process that in turn also created greater chances for error. In phase-shifted c-VEP, subjects typically only need to attend to a single stimulus to build a template, then that template is phase shifted accordingly to match all the available targets. With just a single template to construct, training can progress faster or more training trials can be obtained in a fixed time frame. Likewise, determining phase offset against a single template signal can be more robust to slight temporal desynchronization (provided there is enough space between peak offsets), whereas this is not true with the current rolling correlation coefficient method here. The tradeoff with this approach is that testing of the sample signal could, theoretically, be evaluated continually producing even faster decoding times when using multiple m-sequences, whereas this is not possible with the phase-shifted approach.

### 2.4.3. Gaze-Independent Task Performance

There is only one published covert attention study based on c-VEP that we are aware of. In addition to the overt task mentioned above, (Waytowich and Krusienski, 2015) looked at near-foveal (1° visual angle away) and parafoveal (4° visual angle away) target stimuli. Instead of having users fixate on a central point and attend to one of multiple equidistant targets, however, this study has the users attend to fixation points close to but not on top of a region of a large annulus-based stimulus. They were able to achieve group average accuracy above 80% for a trials based on 6 cycles (6.3s) and at

least 70% for some individuals on only single-cycle classification attempts. While this indicates that c-VEP is generally detectable with covert attention, it does so in a paradigm that is still actually gaze-dependent. Furthermore, it is unclear how well this approach would stand up in even a binary discrimination task if both stimuli were within the same visual angle at the same time.

Covert visual attention BCI studies based on an SSVEP paradigm have been reported as well (Kelly et al., 2005; Walter et al., 2012). The effects of covert attention were detectable and had contralateral representations in left-right binary tasks. However, the power of the detected signals was much smaller and resulted in around a 20% drop in classification accuracies compared to overtly-attended versions of the same task.

The covert c-VEP methods attempted here failed to produce any usable level accuracies and only a few instances barely managed to show significance above chance with one exception from the adaptive online 2DLLAP scenario, though that was still well below usability threshold. One challenge was the placement of the stimuli themselves. In an attempt to align the stimuli with the BCI application workspace, they were placed farther out (5.7° to inner edge) than has been reported for covert VEP studies, which range from 1° to 4.9°. This is, however, well within the range of covert vision-based event-related potential (ERP) BCI studies (Treder et al., 2011; Marchetti et al., 2013; Martel et al., 2014), which placed the center of the stimuli up to 10° from center fixation. Moving the stimuli closer to center may help improve the SNR, though would also likely require a redesign of how the application workspace was laid out. From a different perspective, it may have been that the stimuli were actually too close together. With all

four stimuli visible and occupying roughly the same amount of visual space, only top-down attentional effects were likely introducing differences in the trials from the combined flicking activity from all stimuli. Trying to remove this baseline through subtractive means proved to actually decrease performance, as the subtraction of a common vector from each template only increased their correlations with each other, thus making correlation-based discrimination even more difficult.

Another challenge was the limited duration of the trials used in building and evaluating the classifiers, as only single-cycle trials of 1.034s were evaluated. The experiment was designed to enable up to three-cycle trials in online mode, so further analysis needs to be done to see if using two- or three-cycle trials can improve performance.

The overlapped c-VEP task showed some promise, as three subjects were able to perform greater than the usability threshold of 70% accuracy in offline analysis while two of them maintained that level online as well. While the average group accuracy was not high, the fact that some subjects were able to successfully perform the task at a usable level suggests that this method bears further investigation.

A major challenge to this approach is that, due to the fact that both stimuli were present at the same time regardless of gaze, templates were highly correlated, even more so than those generated for the covert task. Some form of decorrelation method needs to be applied prior to classification, or a non-correlative classifier employed to overcome this issue. Likewise, only singe-cycle trials were evaluated, so extending the analysis to two- or three-cycle trials may prove fruitful as well.

Many covert visual attention BCI studies refer to the work as an independent BCI because gaze control is not required. However, these studies assume that, while reliable gaze movement is unavailable, gaze will remain fixed at a set location. Some motor impaired patients may have involuntary gaze movement or may not be able to maintain gaze directed squarely at a central fixation point. Some work has gone into non-spatial visual selective attention where the whole screen is used to present two overlapping SSVEP stimuli. Zhang et al. demonstrate a method where gaze is fixed at a central spot while attention is directed to one of two overlapping stimuli: red and blue dots that flicker at different rates and rotate around the fixation point counterclockwise and clockwise, respectively (Zhang et al. 2010). The rotation allows for the perception of two separate planes, similar to the checkerboard offset and transparency method used here. They were able to achieve an average accuracy of 72% across 18 subjects. Allison et al. also explored overlapping stimuli for SSVEP, with alternating red and green lines. They did not constrain eye gaze, but were also unable to find high performance accuracy.

This work is the first we are aware of to explore not only a gaze-independent c-VEP paradigm, but also potentially the first of any reported visual BCI paradigm that allows the gaze to be anywhere on the task space. Having the entire display taken up by flickering checkerboards does present a challenge for practical application use, however. The intent is that, by using slightly transparent stimuli, application-specific information can be displayed underneath the stimuli and so still be visible.

### 2.4.4. Adaptive Online Decoding

Similar to (Spüler et al., 2012), two types of decoder adaptation were demonstrated. First was the use of CCA as a spatial filtering method, as this produced subject- and target-specific filters. Unlike that study however, these filters were only computed once based on the data collected during the training phase and never updated afterwards. The templates generated from these filters were continually updated after each trial using the result regardless of whether it was correctly classified or passed the confidence threshold.

In general, the time-evolving templates did not have a substantial impact on overall online performance. Only one subject in the overlapped task showed a substantial increase in overall accuracy using the adaptive CCA-based classifier. For most subjects, the addition of adapting templates either was no different than static templates, or slightly worse performing. This is likely due to the fact that temporal jitter introduced a smoothing and readjustment period to the templates as the average signal offset was corrected. While this would potentially be a beneficial correction to changes in the signals received, the rolling correlation classifier already largely accounted for this.

Finally, the BCI reliability metric presented here offered another means of quantifying the performance of BCI methods beyond simple classification accuracy or ITR. For online BCI studies that lack adaptation or null results, plotting performance over time may serve only diagnostic purposes, so very few report these results. Those studies that do use adaptation still only report straight ITR (Spüler et al., 2012) or,

acknowledging a problem with the base definition, attempt an adaptive ITR calculation (Fernandez-Vargas et al., 2013).

## 2.5. Conclusion

While the goal of developing an effective gaze-independent c-VEP BCI was not achieved with covert-based attention, the overlapped paradigm presented some intriguing possibilities. Online overt c-VEP was demonstrated for the first time using four distinct m-sequences instead of the more common phase-based c-VEP approach. Also, while CCA has been highly successful in visual BCI, the use of Laplacian filters for c-VEP, commonly associated with motor imagery BCI, presents an interesting and computationally simpler alternative. The use of a confidence threshold method for rendering a "no decision" result for online BCI has also only had limited exploration, the evaluation of which prompted the development of the reliability score for online BCI.

Future work will focus on four key areas. First is improving the experimental apparatus to minimize temporal desynchronizations. This is absolutely critical to ensure averaging and correlation-based methods perform at a high level. Second is to investigate multi-cycle classification paradigms for the gaze-independent tasks, as there does not seem to be enough consistency with the single-trial approach. Third is to explore alternate, non-correlation-based decoding strategies. One possible area to explore is techniques used in audio fingerprinting, which also looks for features in the broadband temporal signal, but is based on their relative positions to one another instead of correlating the entire sample. Finally, should that method prove fruitful, work would be

done on developing a continuous decoder that does not require lockstep with the stimulus presentation cycles.

# 3. A NEURAL NETWORK-BASED EXPLORATORY LEARNING AND MOTOR PLANNING SYSTEM FOR CO-ROBOTS

## 3.1. Introduction

Co-robots, collaborative robots that work alongside humans to perform assistive tasks, are becoming more prevalent, notably in the healthcare and telepresence spaces (Kristoffersson et al., 2013). A major challenge for co-robots is the need to make decisions on how to operate in dynamic environments with other autonomous agents (Hayes and Scassellati, 2013). This includes using onboard sensors to detect and avoid obstacles or finding, reaching for, and grasping objects. Embodying the co-robot with some sense of spatial awareness is critical for it to make appropriate decisions on how to proceed with its tasks.

Spatial awareness here refers to the combination of sensory inputs, such as visual and proprioceptive, to construct an egocentric coordinate system for objects in the immediate vicinity of the co-robot. The sensory processing, decision-making, and motor planning components of the task process all share this reference frame in order to achieve effective coordination. For instance, the co-robot needs to know where its body and arm are relative to a visually identified target object in order to plan and execute the appropriate motor actions needed to achieve its goal of grasping the object. If the robot is too far away to reach for a target object from its current position, it will have to move its body closer until the target is within range.

A common first step in developing co-robot control models is to employ simulations and virtual environments to evaluate which strategies and methods have a chance of working in the real world. By avoiding issues such as battery charge and wear and tear of robot parts in simulations, multiple models can be evaluated rapidly without fear of damage to physical components. The main drawback to relying on virtual environments is that many challenges faced in the real world are difficult to simulate accurately without significant effort. Perfectly aligned idealized components of a robotic limb in the virtual environment will have isotropic movement behavior, while in the real world, compliance in the mounting joint and inconsistent servo performance will result in anisotropic movements. Even more challenging is the reliance on data from actual sensors, which are susceptible to noise and artifacts, whereas simulated models frequently use perfect information and highly constrained environments.

These variances between idealized models and physical reality may not be describable analytically, which poses a significant challenge in translating theoretical control systems to practical application. One solution is to embody the co-robot with an adaptive system that integrates and learns actual sensory and behavioral data. By using exploratory learning methods, the robotic agent is able to use a form of unsupervised learning where it gains an operational model of its capabilities by observing the results of its own actions. As the co-robot performs and observes the results of endogenous random movements, i.e. motor babbling, it learns how to link sensory information with motor actions. Once these causal relationship models are built, the co-robot can then transition from passively observing undirected actions to actively planning goal-directed actions.

In this work we present such a system using an adaptive neural network-based controller that employs exploratory learning to enable a hardware robot to autonomously search for, navigate towards, and pick up a distant object as specified by a remote operator. In order to evaluate the viability of the learning, sensory integration, and decision-making models required for these tasks in both virtual and hardware versions of the Calliope robot, we created the CoCoRo (Cognitive Co-Robot) control system. Using CoCoRo, we demonstrate that through motor babbling of its wheels and arm, the Calliope is able to learn how to relate visual and proprioceptive information to achieve the hand-eye-body coordination required to complete its intended tasks.

The rest of this paper is arranged in the following way. Section 2 describes the CoCoRo architecture, the Calliope robotic platform used to evaluate the system, and a detailed description of the components used to achieve hand-eye-body coordination. Section 3 presents the results of several experiments conducted to validate the reaching, navigation, and distant object retrieval goals. In Section 4, the methods and experimental results are discussed and compared to previous work. The paper concludes in Section 5 with a summary of the key contributions.

## 3.2. Methods and Materials

### 3.2.1. CoCoRo Architecture

CoCoRo uses a modular, synchronous architecture. It defines four types of system components: executive agent, sensorimotor devices, cognitive processes, and working memory. Each component in the system is chained together in serial with data flowing

from one component to the next via a data structure termed a cognitive packet. A single iteration through all components is referred to as a cognitive cycle (Figure 3.1).

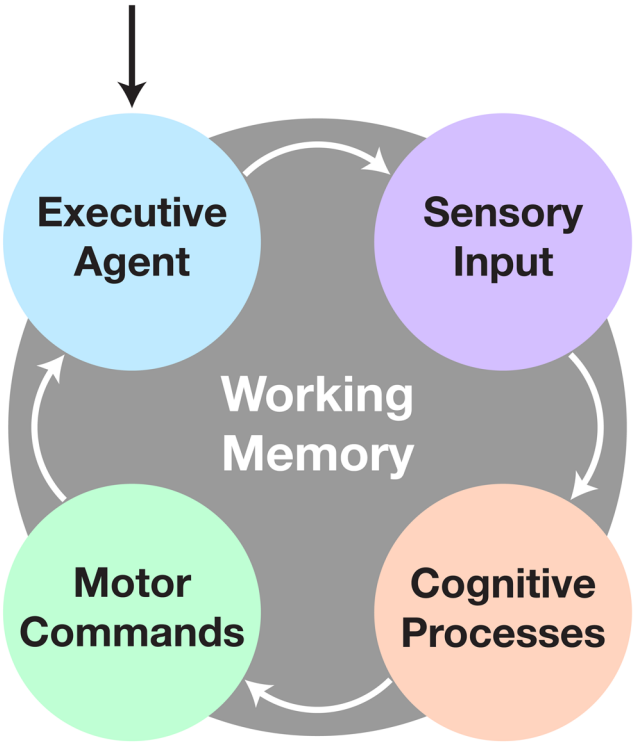


**Figure 3.1. The cognitive cycle.**
After initialization, the cognitive cycle runs until the user halts the robot. The executive agent checks for changes in goal directive, followed by acquisition of sensory data. Next comes processing of the data to fulfill the current objective. Finally, any new motor commands are sent to the appropriate devices and the process repeats. All communication during and persistence across cycles is handled by the working memory system.

The cognitive cycle consists of four phases: executive, sensory, cognitive, and motor. In the executive phase, a cognitive packet is generated by the working memory component, which includes persistent information from the last cycle, time elapsed since the beginning of the previous cycle, and any commands from the executive agent. Next, in the sensory phase, all sensorimotor devices are polled to retrieve new raw sensory data.

Then, in the cognitive phase, cognitive processes act on the sensory and memory data. Finally, in the motor phase, the sensorimotor devices execute any relevant motor commands generated from the previous phases. Finally, the executive agent is given the opportunity to store or transmit any data from the cognitive packet before the next one is generated and the cycle repeats.

The executive agent determines the broad goal objective and task the co-robot will perform. This could arise endogenously through a default behavior pattern or exogenously through commands received from a remote operator. The executive agent also has the ability to store or transmit data for later analysis or telepresence capabilities. Sensorimotor devices are elements that produce sensory data and/or execute motor commands, such as capturing image data from a camera or setting velocity commands to wheel motors. Cognitive processes are intended to be discrete, single purpose functions, such as detecting objects in a visual scene or planning the motor actions needed to articulate a limb toward a desired target. These processes operate on either raw sensory data or the outputs of upstream processes. They then either output intermediate data for use by downstream processes or drive behavior in the form of motor commands. Finally, working memory retains persistent information over the duration of the designated task operation, such as what the goal target is, where it was last seen, and whether certain actions should be enabled or inhibited.

The CoCoRo architecture separates out the realization of a specific robotic platform from the cognitive control model by defining an API for writing the robot control system component modules and runtime programs. Using this approach, cognitive

processes evaluated in a virtual environment can be directly applied to a real world robot without code changes – only the CoCoRo runtime, including operational parameters, and the sensorimotor device modules need be specific to a particular robot environment. Additionally, a common reference frame for working with various coordinate systems in three dimensions is also defined as part of this API to ensure consistent operation between components (Figure 3.2). All code was implemented using the Python programming language.

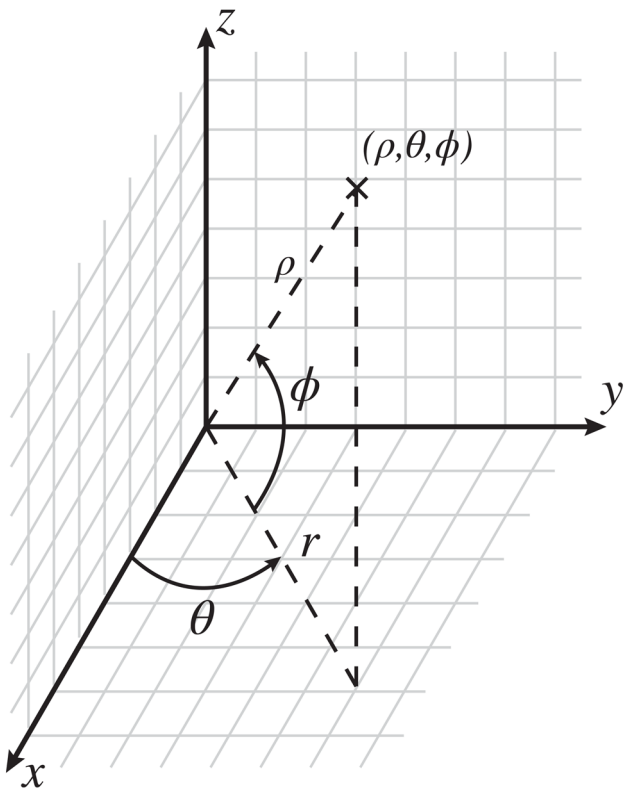


**Figure 3.2. The CoCoRo common coordinate reference frame.**
The origin is defined as the center of the robot's head. In Cartesian space, ***x*** is in front of the robot, with positive values going outward, ***y*** is the horizontal plane, with positive values going to the left, and ***z*** is the vertical plane, with positive values going up. In spherical space, $\boldsymbol{\rho}$ is the distance from the origin to a given point, $\boldsymbol{\theta}$ is the counterclockwise azimuth angle in radians, and $\boldsymbol{\phi}$ is the inclination angle upward from the horizontal plane in radians.

### 3.2.2. Robot Platform

The robot platform used in this study is the RoPro Calliope (Figure 3.3), a reference robot designed for the Tekkotsu robotics development environment (Tira-Thompson and Touretzky, 2011). The Calliope is a multimodal system consisting of an iRobot Create robot base mounted with a 7-degree-of-freedom (DOF) robotic limb and a Microsoft Kinect. All hardware components of the Calliope are centrally controlled via a laptop running Linux (Ubuntu 14.04) resting on top of the Create.

**Figure 3.3. The Calliope robot.**
The RoPro Calliope mobile robot (left) and its virtual counterpart in the Webots (http://www.cyberbotics.com/) robotics simulator (right).

The Create is a differential-drive robot with two drive wheels capable of up to 500 mm/s either forward or reverse and a third balancing wheel. The limb is constructed from

Robotis Dynamixel servos and separated into a 4-DOF arm with horizontal shoulder, vertical shoulder, elbow, and wrist pitch joints on one servo network and a 3-DOF hand with wrist roll and two claws on another network. Each servo has 1024 addressable positions covering 300 degrees. The servos are controlled through a USB-to-TTL interface. The Kinect has a 640x480 32-bit color camera and a 640x480 12-bit depth camera. The cameras have a field of view of 1 radian horizontal and 0.75 radians vertical. The depth camera has an effective sensing range of 0.5m to 3.5m. Pan and tilt control of the Kinect is provided by two additional Dynamixel servos also on the arm servo network. Power for the Kinect and arm servo network comes from a battery pack mounted on the back of the Create, while the hand servo network is powered from the Create's own battery. When fully assembled, the Calliope weighs 10.34 kg.

To enable safe testing and evaluation of the CoCoRo control system and component modules, a virtual representation of the Calliope was developed in Webots (Michel, 2004), a commercial mobile robot simulation software package. Webots allows for robot controllers to be written in a variety of languages including Python, which made it ideal for testing and evaluating the various CoCoRo components.

### 3.2.3. System Implementation

On top of the base CoCoRo platform we developed the components necessary to embody the Calliope with the ability to reach and grasp distant, visually identified objects. This task required the co-robot to perform the following coordination of subtasks: identify and localize objects in the environment, visually search for a desired

object, navigate toward the object, reach for the object, and finally grasp the object in its hand.

The CoCoRo components created to fulfill this task include an executive agent that supported remote operator control; four sensorimotor device interfaces for the Calliope's Kinect, servos, and wheels; and multiple cognitive processes to perform decision-making and coordination for the various subtasks. The full cognitive cycle implementation is depicted in Figure 3.4.

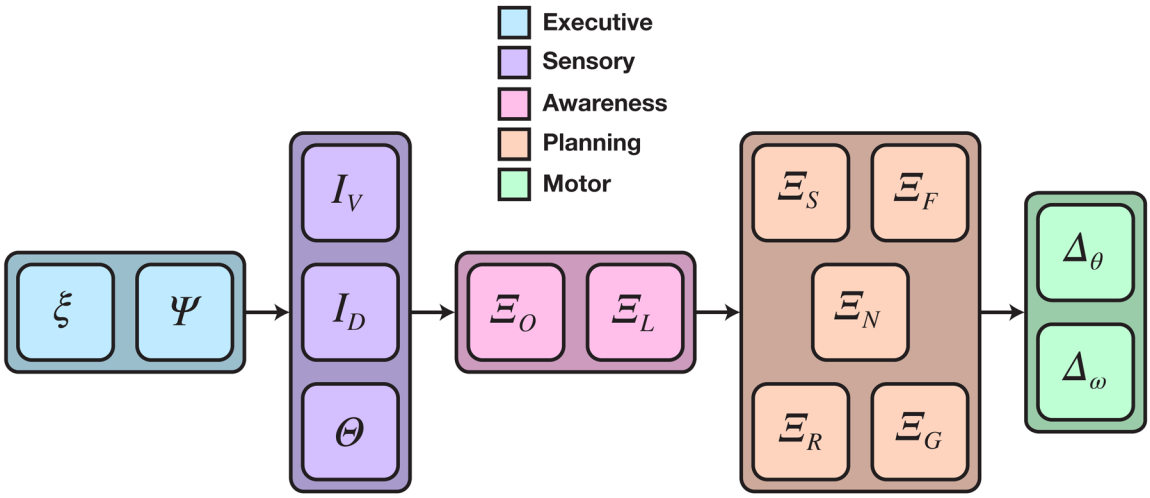


**Figure 3.4. Detailed cognitive cycle model for reaching and grasping distant objects.** Data flows from left to right. Vertically aligned components could execute in parallel, though in practice all components execute in a single serial chain. The cognitive process phase was divided into two sub-phases: object awareness and motor planning. $\boldsymbol{\xi}$ is the cognitive packet, $\boldsymbol{\Psi}$ is the executive agent, $\boldsymbol{I}$ and $\boldsymbol{\Theta}$ are data from camera and joint position sensors, respectively, $\boldsymbol{\Xi}$ is a cognitive process, and $\boldsymbol{\Delta}$ is a motor command expressed as a joint or wheel velocity.

The executive agent was implemented using the Asimov middleware system (Galbraith et al., 2011) to send and receive data between the Calliope and a remote operator. Operators were able to send goal directives and manual motor commands. They

could also optionally receive video frames from the Calliope's camera. Additionally, the agent had the capability to store the contents of each cognitive packet to disk after the end of a cycle for later offline analysis.

Four sensorimotor devices were created: one for the Kinect, one for the Create, and one for each of the two servo networks. The Kinect device captured and provided the raw RGB and depth images while the Create device accepted and issued changes in wheel velocity. The servo devices, corresponding to the arm/neck and hand servo networks, provided the current positions of the joints, set the joint velocities and goal positions, and translated between CoCoRo's common reference frame and the internal Dynamixel reference frame.

The cognitive processes were divided into two functional groups: object awareness and motor planning. Object awareness consisted of two steps: detecting known objects in the visual scene and then localizing them in reference to the body. Motor planning contained the processes for generating and coordinating joint and wheel velocities to control head position, navigation, reaching, and grasping.

As employing robust computer vision methods to object detection was outside the scope of this work, we intentionally chose a simplistic approach. The robot used a color threshold method to detect predefined objects in a constrained environment. Objects were monochromatic cylinders and spheres defined by channel ranges in the CIELAB color space. CIELAB was chosen over RGB due to its greater robustness to changes in luminance. First the raw RGB image was converted to CIELAB using OpenCV and then segmented into a 5x5 grid of tiles. For each known object, the tile with the most matching

pixels that fell into that object's color range was selected. The object was considered present if the pixel count exceeded a threshold of 64 pixels. The centroid of the object was then computed by taking the median x and y image coordinate values of all matching pixels. The depth value was selected by taking the corresponding pixel location from the depth image. Finally, these pixel values were added to the cognitive packet along with the object's label.

Object localization converted all detected objects from raw image coordinates ($\boldsymbol{I_x}$, $\boldsymbol{I_y}$, $\boldsymbol{I_z}$) into relative egocentric locations ($\boldsymbol{\rho}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$). The angular coordinates of each object were computed using the following transforms:

$$\boldsymbol{\theta} = \left(\frac{\mathbf{1}}{\mathbf{2}} - \frac{\boldsymbol{I_x}}{\boldsymbol{I_w}}\right)\boldsymbol{F_h} + \boldsymbol{\theta_p} \qquad (\mathbf{3.1})$$

$$\boldsymbol{\phi} = \left(\frac{\mathbf{1}}{\mathbf{2}} - \frac{\boldsymbol{I_y}}{\boldsymbol{I_h}}\right)\boldsymbol{F_v} + \boldsymbol{\theta_t} \qquad (\mathbf{3.2})$$

Here, $\boldsymbol{I_w}$ and $\boldsymbol{I_h}$ were the image width and height in pixels, $\boldsymbol{F_h}$ and $\boldsymbol{F_v}$ were the horizontal and vertical fields-of-view, and $\boldsymbol{\theta_p}$ and $\boldsymbol{\theta_t}$ were the positions of the pan and tilt joints. This had the effect of converting raw pixel locations into retinotopic coordinates and then adjusting them based on the head position.

For the Kinect, $\boldsymbol{I_z}$ ranged from 0 to 2047, with 0 corresponding to >3.5m, 2046 corresponding to approximately 0.5m, and 2047 corresponding to an error code meaning no depth information was obtained. If an error code was detected, no value was set for $\boldsymbol{\rho}$, otherwise it was computed by:

$$\boldsymbol{\rho} = \boldsymbol{D}(\boldsymbol{I_z}) + \boldsymbol{l_n}\,\boldsymbol{sin}(-\boldsymbol{\theta_t}) \qquad (\mathbf{3.3})$$

The first part transformed the Kinect pixel values to depths given in meters using function $\boldsymbol{D}$ adapted from (Miller, 2010). The second part adjusted for the tilt of the head away from center, where $\boldsymbol{l_n}$ = 0.05m was the length of the neck.

Once objects were detected and localized, they were passed on to the motor planning processes. Head position was determined by whether or not the goal object was detected in the visual scene. When the target was not detected, joint commands were generated to rotate the head in a fixed sweeping pattern to scan the environment until the target was found. Otherwise the robot fixated on the target by generating joint commands to position the head such that the target was held in the center of vision. For the scope of this work, no additional seeking behavior was implemented, so the robot remained stationary while scanning the environment indefinitely if the target could not be detected.

#### 3.2.3.1. Reaching

Motor planning for reaching is based on the DIRECT model (Bullock et al., 1993; Guenther and Micci Barreca, 1997), which belongs to the class of psuedoinverse control methods for redundant manipulators (Klein and Huang, 1983). These methods solve the inverse kinematics problem of choosing appropriate joint velocities that achieve desired end-effector movement by computing the generalized psuedoinverse of the manipulator's Jacobian matrix.

There are two challenges to implementing this solution in practice. First is that the Jacobian matrix must be computable for all possible joint configurations. In stick models or simulations where the robot is treated as a rigid body and the exact geometry of the

arm is known, the solution can be computed directly. For instance, the Calliope's limb (Figure 3.5) has the following ideal relationship between joint configuration and end effector's egocentric location:

$$\boldsymbol{x_e = x_0 + cos\,\theta_1\,(l_1 + l_2\,cos\,\theta_2 + l_3\,cos(\theta_2 + \theta_3) + l_4\,cos(\theta_2 + \theta_3 + \theta_4))} \quad \mathbf{(3.4)}$$

$$\boldsymbol{y_e = y_0 + sin\,\theta_1\,(l_1 + l_2\,cos\,\theta_2 + l_3\,cos(\theta_2 + \theta_3) + l_4\,cos(\theta_2 + \theta_3 + \theta_4))} \quad \mathbf{(3.5)}$$

$$\boldsymbol{z_e = z_0 + l_2\,sin\,\theta_2 + l_3\,sin(\theta_2 + \theta_3) + l_4\,sin(\theta_2 + \theta_3 + \theta_4)} \quad \mathbf{(3.6)}$$

where ($x_0$, $y_0$, $z_0$) is the location of the base of the arm in the CoCoRo common reference frame, $l_i$ is the length of the $i$th arm segment, and $\theta_i$ is the position of the $i$th joint. Using this, the Jacobian matrix and psuedoinverse can be easily derived and computed.

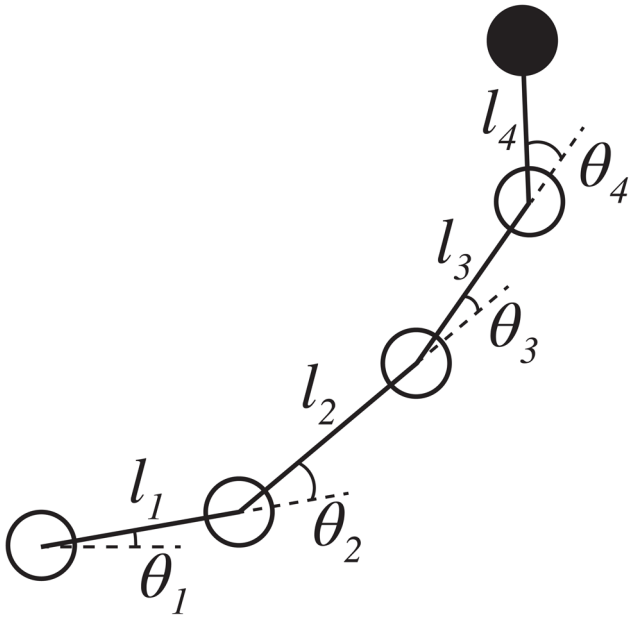


**Figure 3.5. Stick model of the Calliope arm.**
The Calliope arm has four revolute joints arranged in a linear chain. The first joint represents horizontal shoulder movement and rotates about the z-axis. The other three joints, vertical shoulder, elbow, and wrist pitch, respectively, rotate about the y-axis. The limb segment lengths are 0.11m, 0.145m, 0.138m, and 0.135m, respectively.

In the real world, however, the Calliope is susceptible to deviations from this model due to the invalidation of the rigid body assumption, operational limitations, and minor manufacturing defects. As such the error between the actual and computed Jacobian will vary in an inconsistent fashion across the workspace. This is further compounded by the second challenge to using the inverse kinematic model, which is determining where the hand is relative to the desired location.

Obtaining the value for the desired end-effector displacement, $\Delta \boldsymbol{x}$, in a simulation could be as straightforward as tracking the allocentric coordinates of both end effector and desired target and then computing their difference. In an embodied system, where the robot can only act upon data from its sensors, arriving at an appropriate value for $\Delta \boldsymbol{x}$ is non-trivial. The desired reach target is located through the visual system, whereas the hand can be located through vision or, failing that, through an estimate achieved via proprioception. This latter modality is especially important, as the robot's hand may not be visible when reaching is initiated towards a target. Good hand-eye coordination, i.e. agreement between visual and proprioceptive position estimates, is important for obtaining consistent values of $\Delta \boldsymbol{x}$ and thus for maintaining smooth and effective reaching trajectories.

DIRECT addresses both the determination of the Jacobian and achieving good hand-eye coordination through neural network-based exploratory learning mechanisms. By motor babbling the joints in the arm and observing the resulting position of the end effector, the DIRECT neural network is able to learn the relationship between the visual

and proprioceptive inputs. Using this method accounts for deviations from the idealized model by using actual data instead of theoretical predictions.

Our version of DIRECT is similar to that described in (Guenther and Micci Barreca, 1997) as we also use a hyperplane radial basis function (RBF) network (Du and Swamy, 2014; Stokbro et al., 1990) as our choice of neural network. However, we do not attempt to learn the inverse map, but instead only learn the forward map and then use it to numerically approximate the instantaneous Jacobian matrix. This is accomplished by querying the trained model for expected changes in end effector position due to slight perturbations of each joint in isolation. Once obtained, the arm joint motor plan is computed using the psuedoinverse method.

In addition to learning how to articulate its limb to reach for a particular location, the robot also needs to determine if that location is actually within its immediate reach, a task outside the scope of the DIRECT model. We have developed a solution to this reachability problem using the same motor babbling process employed by DIRECT. The reachability of a desired object is whether or not the robot can move its end effector to that exact location from its current position. Both the geometry of the robot's arm and the persistent features of its operational environment determine the reachable workspace of the robot, such as the robot's own body morphology and the relative position of the floor. An object is labeled as reachable if it is contained within a manifold encompassing all points that the end effector can move through. Defining this manifold is not achievable through simple polyhedral, however. Every place the hand can go is considered a reachable location; therefore, all recorded locations of the hand are collected into a point

cloud that represents a sampling of the reachability manifold. A Delaunay triangulation, a mesh of adjacent simplices, is then constructed from this set of points, which creates a convex approximation of the manifold. Additionally, like the RBF network, the Delaunay triangulation algorithm supports incremental update allowing it to be used in both offline and online learning scenarios. The test for reachability of an object becomes whether or not its location would fall within the boundaries of any simplex in the mesh. When a goal object is outside the range of reachability, the navigation system is disinhibited allowing wheel commands to be generated to move the robot toward the target as described in the next subsection. As soon as the object is deemed to be within reachable range, the navigation system is inhibited, preventing any further wheel movements.

Limited grasping capabilities were also implemented. For the purposes of this work, the actual grasping problem was reduced from 3DOF to 1DOF by making all grasping targets vertically aligned cylinders e.g. soda cans. The wrist pose never had to change as it was always aligned for vertical targets, and the finger and thumb motor actions were treated as one synchronous motion to jointly open or close. The distance vector between the location of the hand and the target object that was computed during the reaching task was evaluated each cycle against a minimum grasping threshold. Once the hand was determined to be within this threshold for grasping the target, motor commands were issued to both close the hand at a fixed velocity and cease any new reaching-related joint velocities.

#### 3.2.3.2. Motor Babbling

Motor babbling is an exploratory-based learning strategy for sensorimotor control. Through repeated execution of the action-perception cycle, an agent is able to build an internal model of how its motor behavior corresponds to sensory observations. The babbling aspect is that random actions are generated to explore and discover the range of possible outcomes with limited or no prior knowledge of what is actually possible. This strategy has been successfully used in neural network-based embodied learning for navigation (Zalama et al., 1995) and reaching (Bullock et al., 1993) using endogenously generated pseudorandom joint velocities. A drawback of those approaches, however, is that there is no active exploration of the workspace. Instead they passively rely on a large number of trials to fully cover the space. Recent approaches have explored an active form of motor babbling that either uses a confidence metric in accuracy to direct babbling to less confident regions (Saegusa et al., 2009) or a curiosity-driven reinforcement learning method that seeks out unexplored regions (Frank et al., 2014).

For this work, a semi-active approach was utilized. Endogenous random joint or wheel velocities were generated as in the passive case, but Sobol sequences (Sobol, 1976) were used instead of uniformly distributed pseudorandom numbers. A Sobol sequence is a set of quasi-random numbers designed to evenly cover a space for given sequence length. This provides a semi-active solution, as although it is still largely random, it is guaranteed that the babbling phase will result in actions that explore the entire worksace, thus reducing the number of training iterations required.

### 3.2.3.3. Navigation

The Calliope, owing to the iRobot Create base, uses a differential drive form of locomotion. Like with reaching, in order to navigate toward a desired target, the robot needs to solve the inverse kinematics problem of determining the wheel velocities that will move it to the appropriate location. Typically solved in allocentric, Cartesian space (Dudek and Jenkin, 2010), we present an egocentric, polar space solution that produces smooth trajectories.

Assuming constant wheel velocities ($\boldsymbol{v_R}$, $\boldsymbol{v_L}$) with no slippage over a fixed time interval, the inverse kinematic model is initially given as

$$\begin{bmatrix} \boldsymbol{v_R} \\ \boldsymbol{v_L} \end{bmatrix} \boldsymbol{\Delta t} = \begin{bmatrix} \mathbf{1} & \frac{\boldsymbol{d_w}}{\mathbf{2}} \\ \mathbf{1} & -\frac{\boldsymbol{d_w}}{\mathbf{2}} \end{bmatrix} \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{\theta_R} \end{bmatrix} \tag{3.7}$$

where $\boldsymbol{d_w}$ is the distance between the wheels and $\boldsymbol{s}$ is the desired trajectory arc length with angle of rotation $\boldsymbol{\theta_R}$. Determining ($\boldsymbol{s}$, $\boldsymbol{\theta_R}$) is challenging when working in allocentric coordinates, where the robot must have a sense of the target location and its own relative to a fixed origin in the environment. This problem is avoided when working in egocentric coordinates, where the robot views everything in relationship to itself (Figure 3.6). The relationship between egocentric coordinates in the horizontal plane ($\boldsymbol{r}$, $\boldsymbol{\theta}$) and the associated trajectory arc is

$$\boldsymbol{s} = \frac{\boldsymbol{\theta r}}{\boldsymbol{\sin \theta}} \tag{3.8}$$

$$\boldsymbol{\theta_R} = \mathbf{2}\boldsymbol{\theta} \tag{3.9}$$

By combing Equations 7-9 the egocentric inverse kinematics model is obtained:

$$\boldsymbol{v_R \Delta t = \left( \frac{\theta r}{\sin \theta} + \theta d_w \right)} \quad (\mathbf{3.10})$$

$$\boldsymbol{v_L \Delta t = \left( \frac{\theta r}{\sin \theta} - \theta d_w \right)} \quad (\mathbf{3.11})$$

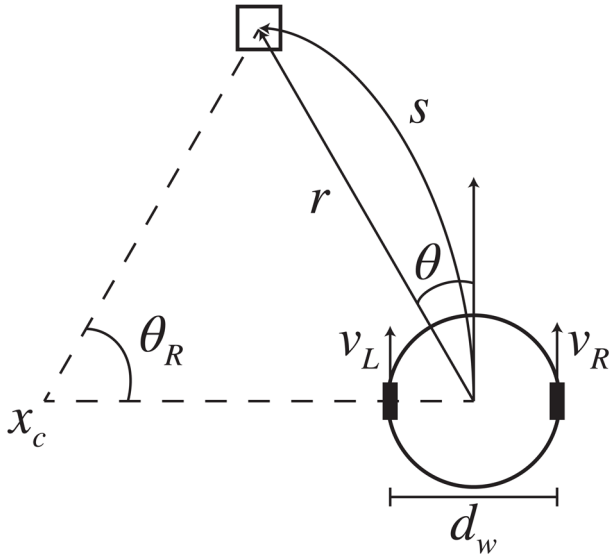


**Figure 3.6. Differential-drive kinematic model.**
Based on the visually determined relative location of the desired target (***r***, ***θ***), the robot generated wheel velocities (***v_L***, ***v_R***) to produce the trajectory arc that would reach the target. The arc has length ***s*** and angle of rotation ***θ_R*** about point ***x_c***.

In practice, however, the wheel velocities have maximum speeds (***v_Rmax***, ***v_Lmax***) that this model does not accommodate; simply capping or scaling velocities that exceed these limits is insufficient as the difference between ***v_R*** and ***v_L*** is central to the desired trajectory movement and must be preserved. Let ***Δt*** = 1s, ***v_Rmax*** = ***v_Lmax*** = ***v_max***, and

$$\boldsymbol{\delta = \frac{v_R - v_L}{2} = \theta d_w} \quad (\mathbf{3.12})$$

then considering the imposed requirement of non-negative velocities, the wheel velocities are given by

$$v_R = max\left(min\left(\frac{\theta r}{sin\,\theta}, v_{max}\right) - |\delta| + \delta, 0\right) \quad (3.13)$$

$$v_L = max\left(min\left(\frac{\theta r}{sin\,\theta}, v_{max}\right) - |\delta| - \delta, 0\right) \quad (3.14)$$

In egocentric space, the relative position of the target is continually changing while the robot is moving, so new velocities are generated every cycle. As no distinction needs to be made between stationary and moving targets as long as they can be localized, this method can produce smooth trajectories for both approaching a fixed location and pursuing a mobile object.

### 3.3. Results

The hand-eye-body coordination tasks were evaluated in three broad task areas: hand-eye coordination, egocentric navigation, and grasping distant objects (Figure 3.7). These experiments were conducted in both virtual and real world environments.

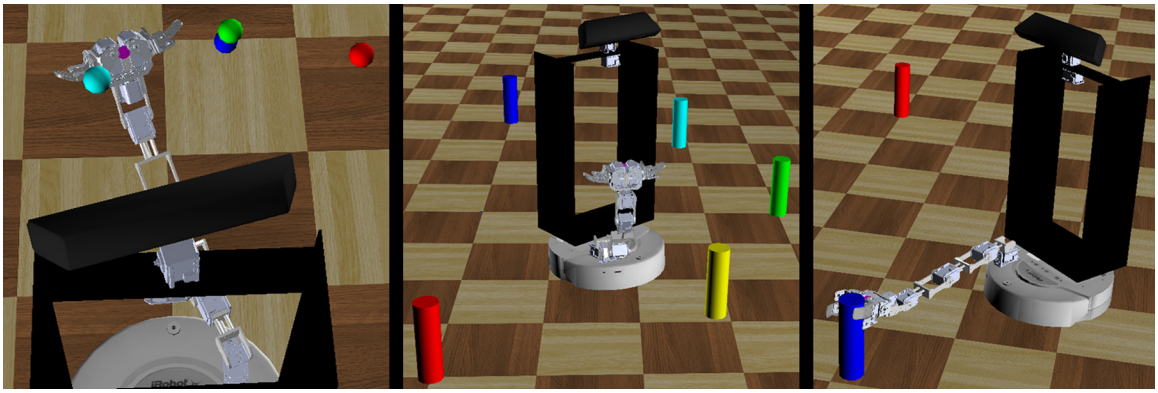

**Figure 3.7. Three robot behavioral experiments.**
The robot performed a series of behavioral tasks to evaluate the feasibility of the motor babbling approach. These tasks included repeatedly reaching to a series of targets in space (left), navigating toward a target and stopping within a set distance threshold (center), and grasping distant objects (right).

### 3.3.1. Hand-Eye Coordination

The co-robot performed arm motor babbling to learn both the relationship between proprioceptive inputs of joint positions to the visual inputs of end-effector position and an approximation of the reachability manifold of the arm. Random target joint positions were generated over [-2.62, 2.62] radians per joint with velocities chosen to require ten cognitive cycles to reach the new position. During this motor babbling phase, the co-robot fixated on its hand, identified by either a magenta circle (virtual) or red foam ball (real) attached to the end effector. If the end effector was visually located during a cognitive cycle, the arm joint positions and target location were recorded.

After the motor babbling phase ended, an offline training phase was conducted. Data outliers due to noise from the real world cobot were identified and rejected by detecting target positions with a nearest neighbor distance greater than 2.5cm. A Delaunay triangulation was constructed from this data to approximate the reachability manifold.

A hyperplane RBF network was trained to learn the forward proprioceptive map. First a grid search was conducted using the collected data to determine the number of bases, Gaussian width, and learning rate to use for the network – the Gaussian centers were spread evenly across the joint input space of [-2.62, 2.62] radians per joint. Next, 10,000 distinct evenly spaced joint configurations and associate hand positions were generated from the rigid-body model of the arm (Equations 4-6) and used to prime the network. Finally, the network was trained on the collected data. To imitate online learning, data points were presented sequentially and only once.

The network parameters chosen for both virtual and real world cobot were three bases per input dimension for a total of 34 or 81 bases, $\sigma$ =1.57, and $\alpha$ = 0.025. The network trained from within the virtual environment was able to reproduce the training set target positions with $R^2$ = 0.926 and RMSE = 0.051 while the network trained on the real world Calliope achieved $R^2$ = 0.942 and RMSE = 0.044.

The efficacy of the hand-eye coordination model acquired through motor babbling was then compared to that of one based on the rigid-body model. The virtual co-robot reached toward four colored targets suspended in the space in front of it in a predetermined order. The hand was deemed to have reached the target if the difference between detected positions was within (0.02, 0.034, 0.034) spherical units. Once reached, the co-robot moved to the next target in the sequence, completing the entire cycle three times. The position of the hand as determined by the robot was recorded and plotted (Figure 3.8).

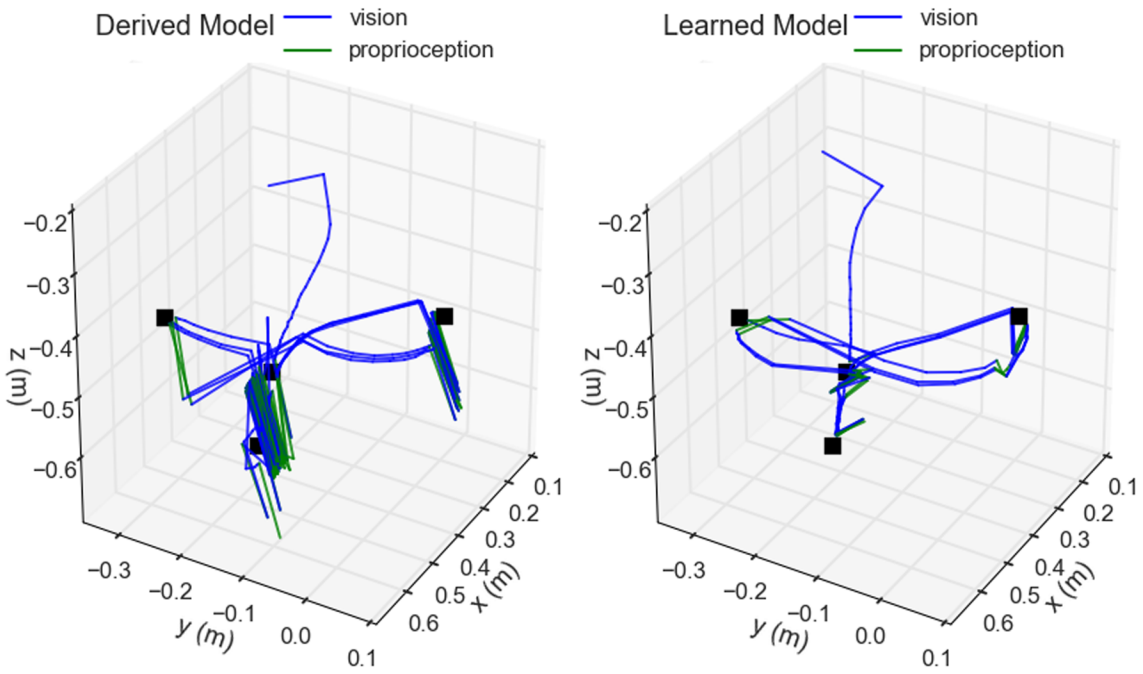


**Figure 3.8. Comparison of derived versus learned models for hand-eye coordination.** The trajectory of the hand positions as determined by the robot are shown during the execution of a reaching task cycling between four visually located targets (black). Blue components of the trace indicate when the hand was visually located, whereas green indicates when the proprioceptive model was used. Arm joint velocities were determined using Jacobian matrices either computed directly from the rigid-body model (left) or approximated from the trained neural network (right).

### 3.3.2. Egocentric Navigation

Motor babbling of the wheels allowed the robot to learn the distance between its wheels. It fixated on a target initially placed 1.5m directly in front of it, recorded the target's position provided from the visual system, then engaged each wheel at a fixed velocity selected from a Sobol sequence over [-0.15, 0.15] m/s for approximately 1 second. After the trial time had elapsed, the robot came to a halt, recorded the new relative position of the target, and computed the wheel distance estimate using

$$\overline{\boldsymbol{d_w}} = \frac{\boldsymbol{\Delta t}}{\boldsymbol{\Delta \theta}}(\boldsymbol{v_r} - \boldsymbol{v_l}) \qquad (\mathbf{3.15})$$

It repeated this process using the reverse of the previously selected velocities to return to its approximate starting position. After several trials of forward and reverse pairs were conducted, the median of the estimates was taken as the robot's learned wheel distance (Figure 9).

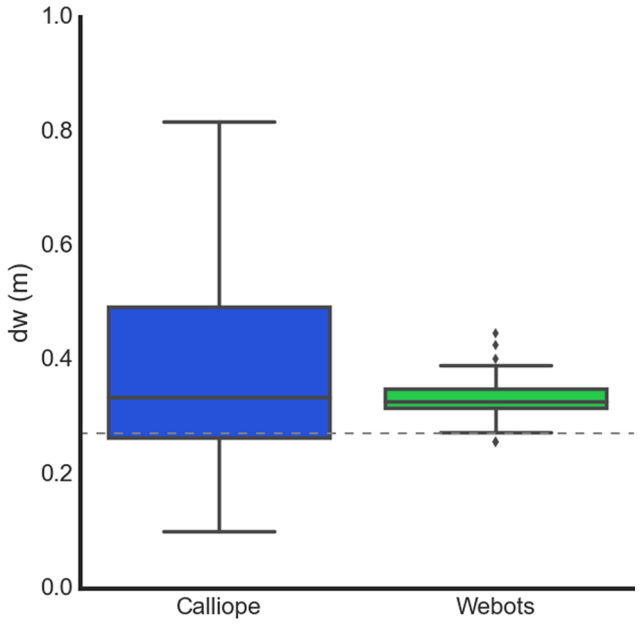


**Figure 3.9. Learning body size through motor babbling.**
The Calliope learned a $\overline{\boldsymbol{d_w}}$ of 0.336m ± 0.088m (n = 91), while the virtual robot learned a $\overline{\boldsymbol{d_w}}$ of 0.326m ± 0.014m (n = 84). The dotted line at 0.272m represents the actual distance between the wheels.

Using the learned wheel distance, the robot navigated toward targets placed approximately 1m away and at -90°, -45°, 0°, 45°, and 90° angles. The robot stopped once it determined it was within 20cm of the target. Once the robot stopped moving, the actual distance between the edge of the target and the center of the robot was measured

and recorded. The real and virtual robots achieved mean stopping distances of 22.7cm ± 0.748cm (n = 15) and 20.2cm ± 0.458cm (n = 5), respectively.

To demonstrate an example of human-robot interaction, the robot also followed a person identified by a held target object. The person started 1m directly in front of the robot, holding the identifying object approximately 0.7m off the ground. The person then walked in an 8m perimeter square pattern just fast enough to prevent the robot from catching up. This was replicated in the virtual environment by having the target object hover above the ground and move on its own. During this task, the position of the target was smoothed using an exponential weighted moving average to mitigate sensor noise. Both the virtual and real world robots maintained pursuit over traversal of the pattern (Figure 3.10).

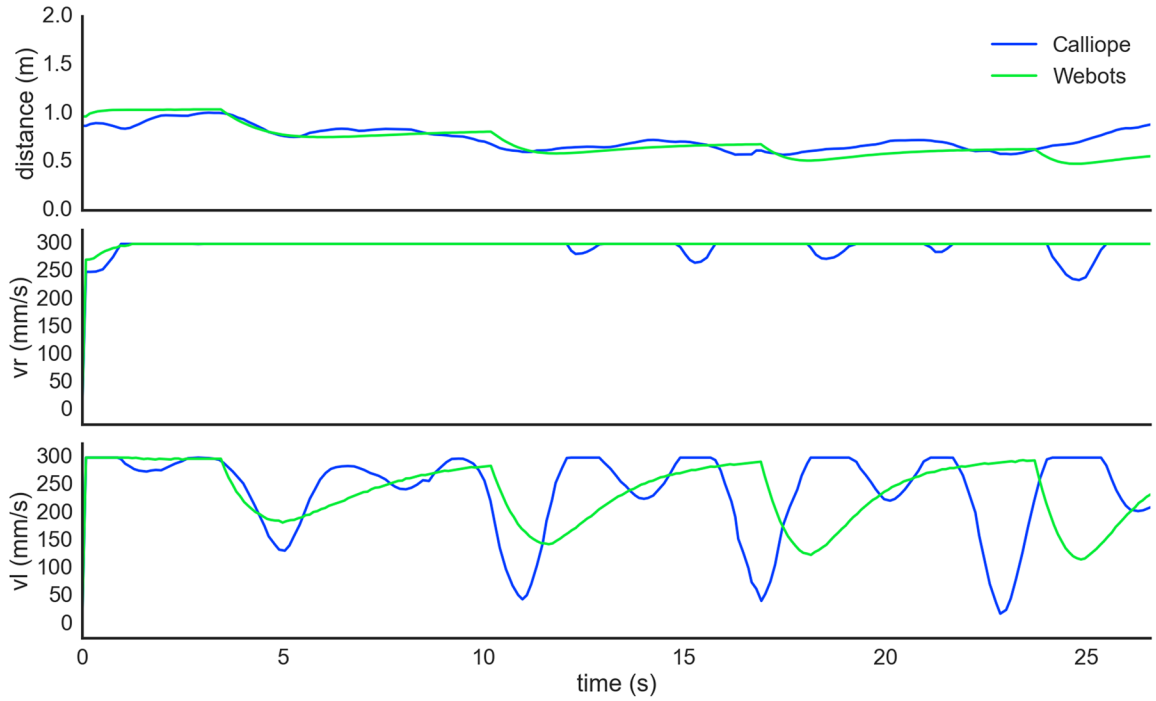


**Figure 3.10. Autonomous pursuit task.**
The robot visually tracked and pursued a target moving counterclockwise in a square pattern. The self-determined distance between the robot and target (top) slowly decreased as the robot got closer during the turns. Right wheel velocity (center) was kept at maximum while left wheel velocity (bottom) modulated during turns. The dips in both the right and left wheel velocities of the Calliope (blue) following a corner turn are from the robot overshooting and correcting itself.

### 3.3.3. Grasping Distant Objects

The coordination of reaching and navigation was demonstrated in a task where the Calliope had to pick up an operator-directed target in the environment. The Calliope was placed in an environment with two (real) or three (virtual) known objects located at (1.5m, 0°), (1.4m, -45°), and (1m, 45°) away, all outside the immediate grasping range of its arm. It was then activated and assigned one of the objects to find and pick up. The robot had to coordinate head position, wheel velocities, and arm and hand joint velocities

to complete the task successfully (Figure 3.11). The virtual robot performed one trial for each target and managed to grasp and lift each for 100% completion. The real robot performed five trials for each target and successfully completed the task 80%, 40%, and 60% of the time, respectively, for an overall completion rate of 60%. In all cases where the Calliope failed to complete the task, it was because it grazed the target object with its hand, knocking it over. It still managed to stop within reaching distance and move its hand to the correct vicinity of the target. Videos of both virtual and real robots performing the task can be found in the supplementary materials.

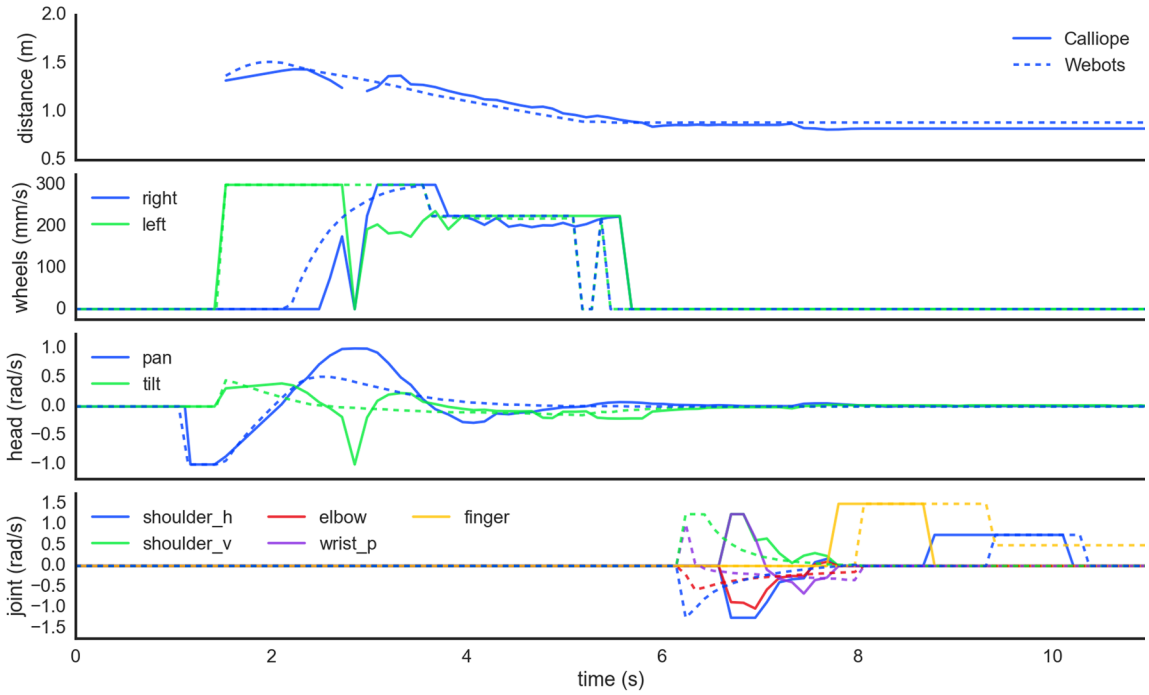


**Figure 3.11. Motor planning coordination while picking up a distant object.** In order from the top, these plots show the detected distance to the target object followed by the generated wheel velocity, head position, and limb joint commands, respectively for both real (solid) and virtual (dashed) robots. First the robots scan the scene searching for the target. At 1.5s, they locate the target to the right and navigate toward it while maintaining gaze fixation. Around the 5.5s mark, the robots determine the object is reachable, stop navigation, and ensure head position is stable before starting to reach toward the target. Grasping is initiated around 8s in and takes about 1.5s to complete before the obtained target is finally lifted off the ground.

## 3.4. Discussion

### 3.4.1. The CoCoRo Control System

One of the design choices with CoCoRo was to use a serial, synchronous data flow model. This was chosen for its relative simplicity of implementation and the ability to chain certain cognitive processes together in a defined order for coordination purposes. However, the penalty for using this architecture was that the entire cognitive cycle was

rate-limited by the slowest component. This had no impact on the virtual environment where simulation time had no bearing on real time, but it did affect the real robot, where the object identification process proved slowest due to the naïve implementation of color matching applied to the relatively large input image. Many other robot platforms, including Tekkotsu and MoBeE (Frank et al., 2012), use threaded, finite state machine architectures, which can achieve real-time performance and take advantage of concurrent and distributed processing of information. This avoids the rate-limiting problem of the serial architecture at the cost of increased system complexity. However, with the computational power inherent in modern laptops, like the one mounted on the Calliope, CoCoRo's simplistic structure did not interfere with the ability of the robot to complete tasks effectively. The Calliope operated at an average rate of 10Hz during task execution, which was sufficiently fast enough to adjust motor commands as needed for the tasks undertaken albeit with the maximum wheel and joint velocities artificially reduced. Wheel velocities were capped at 300 mm/s and arm joint velocities were capped at ±1.5 rad/s. The simulation step time in Webots was set at the default value of 32ms. As all sensor and motor component control steps must be a multiple of this simulation step, 96ms was chosen to offer a comparable decision performance rate.

An additional benefit of using the serialized data flow model was the ability to easily capture and store the cognitive packet to disk, the data structure that contained all the sensory inputs, intermediate processing, and motor outputs from a given time point. This process was used extensively for both debugging purposes and offline analysis, such as providing the data for several of the figures in this paper. A tool was also created to

reproduce robot point-of-view movies from these packets (Figure 3.12), which proved invaluable for tracking down issues with object detection and localization.

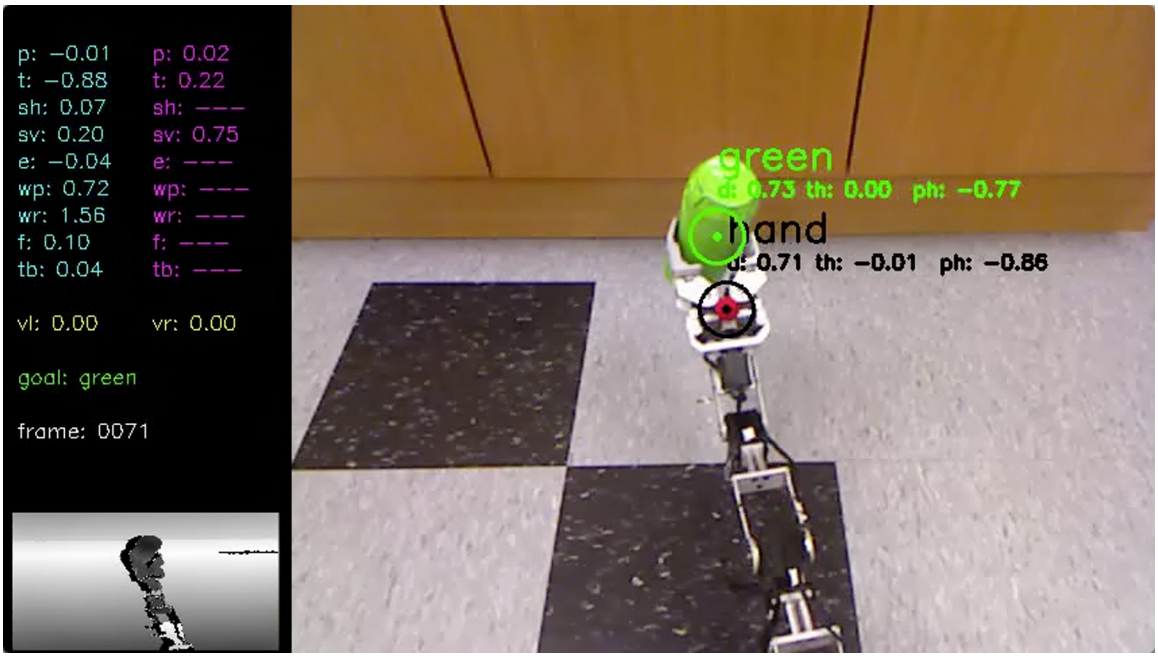


**Figure 3.12. Calliope lifting an object.**
This is a frame taken from a movie (see supplementary materials) reconstructing the Calliope's point of view during a task to grasp and lift a green object initially located 1.5m away. The movie is created from stored cognitive packets generated during the execution of the task and includes all sensory inputs, motor commands, and identified objects.

### 3.4.2. Virtual Environments

The use of simulations and virtual environments are key to developing and evaluating robotic control systems. If the virtual environment provides a good enough approximation of the real environment, certain tasks can be bootstrapped in the simulation first, such as building up the internal neural network weights for control tasks. These weights can then be transferred directly to the physical co-robot which would then need a shorter recalibration learning session than if it had started with untrained

networks. We used just such a method in training the neural network responsible for reaching. Training it first using an idealized set of inputs to outputs primed the network and provided reasonable results for locations in the reaching space that were not obtainable through motor babbling alone, i.e. where vision failed to detect the hand. The later data collected from motor babbling was then able to retrain the network to be more in line with the actual observed results instead of those generated by the rigid-body approximation.

However, we also encountered several discrepancies when moving between virtual and real sensors. The images from the virtual Kinect were always crisply rendered, whereas the images being pulled from the real Kinect were susceptible to noise. The sources of noise included motion blur introduced by movement from the body and head, and potential changes in luminance due to automatic white balancing performed by the Kinect video camera. The depth camera in the virtual environment, like the virtual video camera, was generated from the OpenGL buffer directly and did not suffer the effect of infrared shadows. These shadows were areas visible in the video image but in which no depth information could be obtained due to objects in the foreground preventing the infrared signals from reaching them. Despite these challenges in using video and depth image data in the real environment, the Calliope was still able to perform at a high level for the tasks explored, though additional checks had to be added for cases in which objects were visible but no depth information could be obtained.

Likewise, the behavior of servos varied between simulation and reality. In the virtual environment, servos would move smoothly in response to any requested velocity

within defined operational range and supported high precision positional accuracy. The real servos, on the other hand, were limited by having only 1024 addressable positions for a resolution of about 0.005 radians. This contributed to occasional jittery behavior when attempting to hold joints in a particular pose due to the effects of rounding. The real servos also did not support specifying a velocity of zero to halt movement. Instead, we had to rely on a combination of velocity and positional control to achieve a fixed joint configuration. Finally, the skeleton of the arm itself contained screws prone to loosening during continual operation, resulting in slight changes to the position of the end effector over time.

### 3.4.3. Hand-Eye Coordination

Controlling redundant joint manipulators is an open challenge in robotics, as closed-form analytic solutions to the inverse kinematics problem may not exist. Feedback-based control strategies have proven successful, but require reasonably accurate sensors to provide the needed error signals. These can be difficult to acquire for a non-planar limb outside of simulation or highly controlled workspaces. As a requirement of co-robots is to operate in largely uncontrolled environments, the control system should not rely on external sensors and fixed workspaces. We used a variant of the DIRECT model, a biologically inspired neural network approach to feedback-based control of a limb. Desirable features of DIRECT that make it useful for co-robots are that it is egocentric, so all sensor information comes from its own perspective, and it can adapt to changes in limb configuration. However, DIRECT, like many other solutions,

was validated in simulation using perfect knowledge of end-effector position and stick-model limbs. Other applications of DIRECT have been reported (Vilaplana and Coronado, 2006; Grosse-Wentrup and Contreras-Vidal, 2007; Bouganis and Shanahan, 2010), but these too were only performed in simulation with perfect positional knowledge and lack of physical constraints beyond joint rotation boundaries. Our implementation is the first instance we are aware of that demonstrates the efficacy of DIRECT using actual computer vision to determine end-effector and target localization. Furthermore, this is also the first demonstration of DIRECT embodied in a real-world robot working in a 3D workspace.

Using visual inputs from a camera and working with a physical robot presented its own set of challenges for DIRECT, computer vision not with standing. DIRECT uses motor babbling to learn the space of movements, so it must be able to observe the end-effector in order to learn how it moves in a particular part of the workspace. With a fixed camera vantage point, body components obstructing views, and limitations of the camera sensor, the Calliope had several blind spots. Our solution to this was to prime the network before motor babbling commenced using the rigid-body model of the arm to generate thousands of training points evenly spaced across the entire hypothetical workspace. The network was trained using a learning rate an order of magnitude lower than that used during motor babbling so that real observed data would take precedence.

For instance, the observed location of the robot's hand in the virtual environment displayed close similarity to that of the rigid-body model for the portion of the workspace the arm was able to reach during motor babbling (Figure 3.13). As can be seen, this

actually represented only a fraction of the theoretical range if the arm was free of any obstacles. The use of an identifying color marker on the top of the hand also produced a compressed range of visible locations. If the configuration on the arm resulted in the hand positioned upside down, for instance, it would not be recognized.

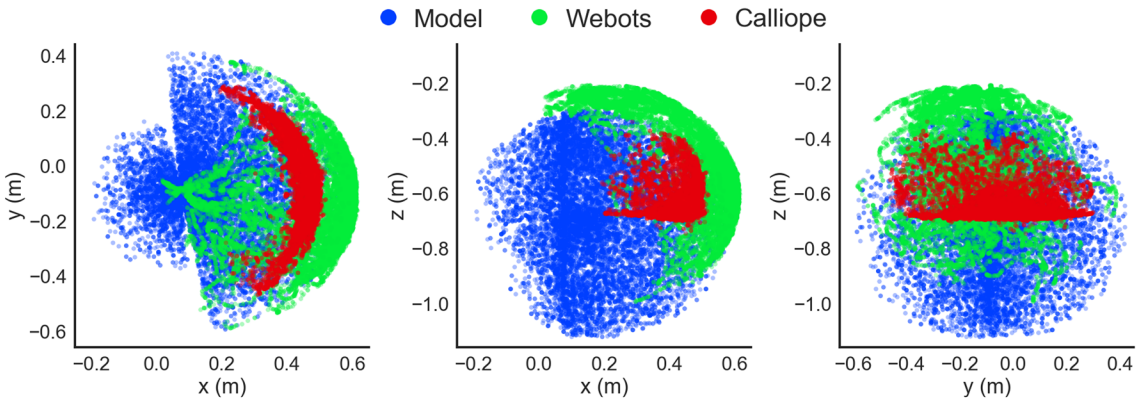


**Figure 3.13. Detected hand position during motor babbling.**
Recorded hand positions are shown in the xy- (left), xz- (center), and yz-planes (right). A kinematic stick model (blue) computed hand positions using randomly generated joint positions and the geometry of the arm, while both the Webots virtual environment simulation (green) and Calliope (red) used visual information to determine hand position during a motor babbling task.

The major difference between theoretical and detected position came in the real world Calliope, where the detected distance of the hand was almost 10cm on average closer than the model would predict. This can be attributed to two factors: a greater offset from the location of the visual marker to the end of the hand and the less precise distance estimation from the actual Kinect's depth camera versus the simulated Kinect. The observed range of motion for the real hand was even more compressed than the virtual one, however, due to the Kinect's blindness within close proximity. Relying solely on either observed data or theoretical model would have produced large gaps or erroneous

estimates, respectively. Using the theoretical model for initially priming the RBF neural network then further training with the motor babbling results provided a solution that enabled the use of both approaches to complement each other.

For actually generating joint trajectories during a reach task, the analytically determined Jacobian from the rigid-body model produced similar behavior to that approximated by the trained neural network in three of the four reaching segments. The rigid-body model, based in Cartesian coordinates, produced straighter trajectories between targets but had significant disagreement between its visual and proprioceptive locations as exhibited by the trajectory shifts when switching between the modalities occurred. This most impacted the model during the downward trajectory from target three to four, where it got stuck and convulsed for several seconds before finally achieving a correct configuration. This was due to the first target, placed just above and in front of the fourth, occluding the marker on the hand toward the end of the trajectory resulting in the model flipping between visual and proprioceptive locations. The disagreement between the two was large enough that, when using visual input, the hand was perceived where it actually was, above the target, but when using proprioception, the hand was perceived to be below the target. This conflict produced the observed spasms. While all targets were eventually reached here, in a separate instance the arm became locked into a never-ending cycle of jittering up and down and the trial had to be terminated. The neural network model, by contrast, was based in spherical coordinates, produced slightly arced trajectories, and had much greater agreement between proprioception and vision. It experienced no difficulties in any of the reaching segments. Even when losing sight of

the hand, there was enough agreement in the two modalities to allow for consistent smooth behavior during the trials.

### 3.4.4. Egocentric Navigation

In determining wheel distance, the real and virtual robots produced very similar final estimates, with the main difference being the noisiness of the Calliope's samples. Both robots were over the actual distance by 6cm and 5cm, respectively. This error could be related to the relative distances between wheels, camera, and reference target, as extending the wheel distance out further in the virtual environment produced very accurate estimates. This error did not appear to have an impact on the actual navigation tasks, as both the stationary and pursuit tasks produced comparable results. In the stationary task, the difference in average stopping distance was only 2.5cm, while in the pursuit task, the Calliope performed well despite slightly overshooting the turns then having to correct.

This method for egocentric navigation employs an aiming strategy (Franz and Mallot, 2000) for local navigation, where the goal of path planning is to keep a desired target position directly in front of the robot while moving towards it. Other aiming approaches include Concentric Spatial Maps (CSM) (Chao and Dyer, 1999), which uses a neural network to store goal positions and obstacles in discrete locations arranged in concentric circles around the agent. A similar, though non-neural, approach to CSM is used to produce multi-agent pedestrian navigation through crowds (Kapadia et al., 2012). Both of these methods account for obstacles whereas we assumed a clear path. CSM,

however, requires the environment map be loaded a priori, while the pedestrian model does not use sensory information from the agents themselves and instead determines them from the global simulation state.

An alternative and complementary strategy to aiming is guidance (Franz and Mallot, 2000), where the relative positions of environmental cues are used to determine desired trajectories. Examples of guidance-based approaches include ENav and variants (Altun and Koku, 2005; Fleming, 2005). They are based on the sensory egosphere (SES) (Albus, 1991), a 2D spherical projection of incoming sensory data to a spatial representation of the agent's environment, where the goal is to match the angular displacements of visually identified landmarks in the current SES with those provided in the desired SES. ENav is the only other method we are aware of to have reported implementation attempts outside of simulation (Fleming, 2005), though with limited results.

These navigation methods provide path planning abstracted from a specific kinematic model of locomotion. While ostensibly more general, they may produce trajectories that are not possible by an actual mobile robot, so an appreciation for the inverse kinematics of locomotion for target robot platforms is critical to produce a model that can work in real environments.

For differential-drive navigation, the inverse kinematics problem can be solved by breaking down the desired trajectories into pairs of distinct motions: first rotate in place to face the target, then drive straight forward toward it (Dudek and Jenkin, 2010). This, however, produces jerky motion, requiring the robot to stop forward progress every time

it needs to rotate. For a clear path in an ideal environment, the expectation would be only one rotation and one direct forward trajectory. However, in a real world environment, wheel slippage, dynamic target location, and perturbations in the floor can result in deviations from the ideal trajectory, requiring compensatory corrections, each resulting in the robot having to stop, rotate, and begin forward again. This would be especially inefficient in the egocentric model, where the relative positions of objects are always changing as the robot moves.

Similar arc-based solutions to the one above have been proposed in both Cartesian (Bethencourt et al., 2011) and polar (Maulana et al., 2014) forms, though the former relies upon accurate accumulation of encoder data to reconstruct allocentric position while the latter is geared toward following a fixed track. Instead of learning just the body size as demonstrated here, the NETMORC model (Zalama et al., 1995) attempts to learn the inverse kinematic solution itself through a neural network trained via a similar motor babbling phase. However, only simulated results with perfect positional information used in training the network were reported.

This is the first work we are aware of that combines the use of egocentric navigation with a specific model of inverse kinematics. Not only does this approach succeed with a high accuracy in simulation, it works very well in a real world robot despite the increased noise from and limitations of actual hardware and environments.

### 3.4.5. Grasping Distant Objects

The task of grasping and lifting distant objects combines the previously described subtasks into a unified whole, requiring an additional layer of coordination on top of the individual motor plans. The motor planning coordination strategy used in this work was to take a largely lock-step approach, where the individual subtasks were disinhibited only when their role was called upon. The only exception to this was head movement, which operated in parallel to the progression of navigation, reaching, grasping, and lifting. This coordination was implemented by having each cognitive process in the chain alter or check the working memory system and inhibiting or disinhibiting itself based on its state.

Two main factors can be attributed to the cases where the real robot failed to complete the grasping task by knocking the target over. First is the simplistic object identification method, which is highly susceptible to noise and treats objects as points. This results in generally poor performance when precision adjustments were needed, which were typically required due to the second factor, the segregated process of only reaching once navigation stopped. In this arrangement, the arm is held out and to the side until the reaching subtask begins. It makes a downward arcing trajectory to reach the target, which can result in the hand clipping the side of the object if the robot is even a centimeter too close. If the hand began its reach earlier while the robot was still driving forward, the hand could be brought into position before there was a risk of inadvertent contact.

Other approaches to visually guided mobile manipulators employ more fluid motor control and coordination (Andaluz et al., 2012a; Kazemi et al., 2012; Andaluz et

al., 2012b). Related to the co-robot goal of working in unstructured environments, (Xie et al., 2014) presents a model for visual-guided control for grasping household items. All of these systems use a camera mounted on the end-effector instead of elsewhere on the body. These eye-in-hand visual servoing systems can achieve greater grasping and manipulation accuracy at the expense of having to manage a potentially highly articulated neck, i.e. the arm itself, when not engaged in an actual reach action. They also lack the flexibility of the alternate hand-to-eye approach used by the Calliope.

The simplistic method for visual object detection worked well enough for both reaching and navigation in the virtual environment where color detection is much easier. It was less effective in the real world as it was highly susceptible to noise. For navigation, which operated in 2D, this proved less of an issue, but it did impact the success of reaching and grasping, which required accurate 3D locations. The grasping method used was also the simplest available. Real world use would require more intelligent grasping algorithms for shaping the hand to accommodate a variety of object shapes. As CoCoRo supports drop in replacement of components, upgrading to more robust computer vision and grasping processes would be possible.

The egocentric model worked well for traversing the immediate vicinity of the robot assuming a clear path to the target destination. If any obstacles were in its path that did not occlude the target object, however, the robot would attempt to drive through them. Likewise, if the robot failed to detect the desired target in its sensory field, it would either have to revert to an allocentric representation to derive new egocentric coordinates from memory or engage in some form of directed search.

### 3.5. Conclusion

We presented a control system with an eye toward co-robots that used motor babbling to enable a robot to learn about aspects of its own configuration in regards to hand-eye-body coordination. This system was built on a software platform designed to enable modular evaluation of the learning, sensory processing, and decision-making motor components across both virtual and physical versions of the Calliope robot. The capabilities embodied in the robot enabled it to autonomously follow a person around a room and retrieve distant objects specified by a remote operator. In order to achieve this we demonstrated a variant of the DIRECT neural model for reaching in a hardware robot and complemented it with novel methods for determining if the intended reach target is actually within the robot's grasp and a means for egocentric-based navigation to drive it toward the target if it is not.

There is still significant work to be done in order to extend this initial system to more practical real-world co-robot use. Adapting to cluttered and dynamic environments would require a much more robust and powerful form of visual object detection and identification that the simplistic model currently used. The navigational system would also be extended to handle obstacle avoidance and combine allocentric and egocentric path planning strategies. Smooth concurrent motor control coordination would also be a desirable improvement over the current lock-step approach.

# 4. BCI CONTROL OF A SEMI-AUTONOMOUS ROBOT

## 4.1. Introduction

The real value of a BCI is how useful it is in a practical setting, therefor it is important to evaluate BCI performance not just in an offline state or in an online exogenously directed capacity, but in an online endogenously directed scenario as well. In other words, the BCI performance needs to be evaluated when the user is trying to perform actions on their own initiative rather than being tasked to do certain fixed steps or cues. Furthermore, the BCI method should be applied to user-centric applications that perform actual desirable functions, such as communication, entertainment, or control of robotic agents.

Controlling robots via EEG-based BCI in both manual control (user has total control) and shared control (some amount of autonomy is available to the robot) settings is an active area of study (Bi et al., 2013). Many of these studies involve the user driving themselves around in a mechanized wheelchair, e.g. (Leeb et al., 2007), while others focus on a stationary user controlling a remote robot, e.g. (Dasgupta et al., 2010).

In order to support the creation of these user-centric BCI applications, such as an autonomous robot interface, we developed the Unlock framework, a Python-based software system geared toward the development of BCI apps. This chapter describes the Unlock framework in detail and the BCI apps that were created and tested to demonstrate the practical applicability of the c-VEP methods described in Chapter 2.

## 4.2 The Unlock Framework

The Unlock framework is a software system written in the Python programming language that supports the creation of BCI applications (available: https://github.com/NeuralProsthesisLab/unlock). These applications can be both purely user-centric, such as a game or communication interface, as well as research-driven experiments. Both the applications evaluated in this chapter and the experimental setup described in Chapter 2 were developed using Unlock.

The primary goal of Unlock is to provide a free, open source, BCI platform that can separate the application side of the interface from the BCI underpinnings in order to allow both motivated developers and BCI researchers the ability to contribute. It accomplishes this by decoupling and compositing various commonly used components together in order to create BCI applications. For instance, data acquisition and signal processing components, such as task-based decoders, can be designed by experts while adhering to a common application programming interface (API). This API ensures that a developer creating an entertainment app would not necessarily need to know which particular BCI paradigm ends up being used, e.g. four-choice SSVEP, 32-tile c-VEP speller, or even an eye tracker, as long as their application responded to specific defined event notifications and messages.

An early version of Unlock is described in (Brumberg et al., 2012), though most of the underlying architecture has changed since that work was published.

### 4.2.1. Architecture

Unlock follows a Model-View-Controller architecture. Models represent the state or behavior of the application, such as where a cursor is located or the current stage of an experimental task, and how to change those things in response to commands. Views are responsible for outputting information based on the models, e.g. render a cursor icon on the screen relative to its representation in the underlying model. Finally, controllers both handle the data flow between external sources and the models and ensure the views are updated accordingly.

An Unlock app consists of one or more controllers, each with associated models and views. For instance, a time scope app has a controller that collects raw data and passes it to a model. The scope model determines how many data samples have come in, appends them to a circular buffer, advances the cursor position the appropriate number of places, and recomputes any autoscaling parameters. The view redraws the screen by creating connected line charts generated from the model's buffer and scaling parameters. This process repeats on every draw cycle called by the OpenGL runtime.

In addition to the models, views, and controllers described above, there are signal objects, representing data acquisition sources, and decoder objects, representing online signal processors and classifiers. Signals can be direct Python interfaces to hardware devices, networking interfaces that retrieve hardware-derived data transmitted from other sources, or pure software sources, such as random signal generators. Decoders are a series of signal processing and classification stages used to determine actions from the raw data. These are typically tied to the the user-interface and determine which set of

commands are available. Examples include a template matching decoder for a four-choice c-VEP BCI and an eye tracker that detects eye blinks.

The complete breakdown of the actions during each draw cycle is as follows. First the main Unlock runtime calls the **poll_signal** method on the active controller with the time delta since the last draw cycle occurred. This method polls the data acquisition source, and any data, such as EEG, eye tracking, or event markers, that have come in since the last poll are stored in a command object along with the time delta. If any decoders have been attached to this controller, this new data is also run through each decoder in order, with the command object updated with additional properties as necessary. Once all data acquisition and decoder activity is complete, the controller calls the **process_command** method with the generated command object as a parameter on all its associated models in order. Each model then responds accordingly to the data and other commands contained in the object. Finally, after all models have processed the command object, the controller calls the **render** method on all its associated views, which prompts each view to perform any on-screen presentation updates.

### 4.3. Methods and Materials

The BCI decoder methods used in these online tasks were the same as those used during the online testing phase for the overt task described in Chapter 2 with three major differences. First, instead of the cue-prep-trial-feedback-rest pattern, only the trial and combined rest-feedback phases were present. These phases also had significantly shorter

durations of 1.2s and 0.15s, respectively. This pattern continued indefinitely until the application was closed. Second, templates did not adapt over time but instead remained fixed from those generated after a training session. Finally, a confidence threshold was applied to the output of the classifier. If the correlation score of the predicted target was less than 0.25, the decoder issued a "no decision" and no action was taken by the active BCI application.

### 4.3.1. Online BCI Tasks

Subjects were asked to control three different Unlock applications using the overt four-choice c-VEP paradigm: GridCursor, GridSpeak, and a mobile robot controller in manual and autonomous modes. All three apps used the same target stimulus configuration used by the overt task described in Chapter 2 – four 180x180 pixel targets centered 360 pixels up, down, left, and right of center. This allowed for a usable application workspace of 540x540 pixels in the center of the screen.

The first app, GridCursor, was a simple game that acted as an introduction to 2D cursor control via the BCI. The subject was presented with a 5-by-5 grid that filled the entire workspace between the four flickering stimuli. In the center cell was a blue square (cursor) and in a different, randomly selected cell was a green square (target). The user was tasked with moving the cursor to the target and then issuing a selection action. Once successful, the target would randomly appear at a different location in the grid and the process would repeat. Users moved the cursor up, down, left, or right by attending to the stimulus at the top, bottom, left, or right edge of the grid, respectively. Selection actions

were achieved via either a double eye-blink or space bar key press. While the users were given a target destination, they could choose which route they took to navigate there.

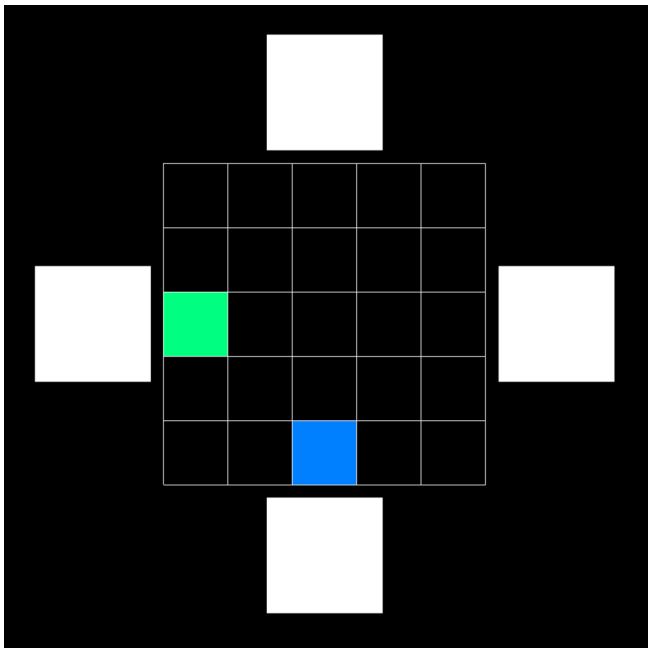

**Figure 4.1. The GridCursor app interface.**

The second app, GridSpeak, was an example of an alternative and augmentative communication (AAC) tool. The same 5-by-5 grid interface appeared, though now each grid cell contained a word or phrase. This time the cursor was represented by a red square outline. Subjects moved the cursor over the desired phrase and issued a selection action, upon which the computer would emit a pre-recorded voice speaking the phrase associated with the selected grid cell. Initially users were asked to select a couple phrases, but then were granted freedom to choose their own.

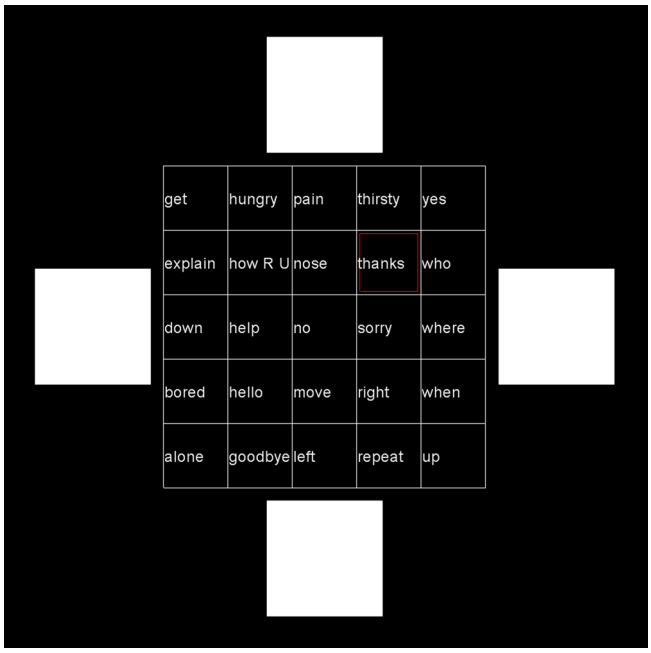


**Figure 4.2. The GridSpeak app interface.**

The third app provided a way to interact with a mobile robot. The four stimuli remained, but the grid in the center of the screen was replaced with a video feed coming from a camera mounted on the robot. It had two different modes: manual and auto. In the manual mode, users took direct control of the robot's motion. The up and down stimuli corresponded to commands to move forward and backward, respectively, while the left and right commands corresponded to turning in-place to the left and right, respectively. All motion was set at a fixed speed of 50 mm/s. A selection event sent a stop command to the robot. Users were asked to navigate toward either a green object or a red object placed in the environment within 2m of the front of the robot.

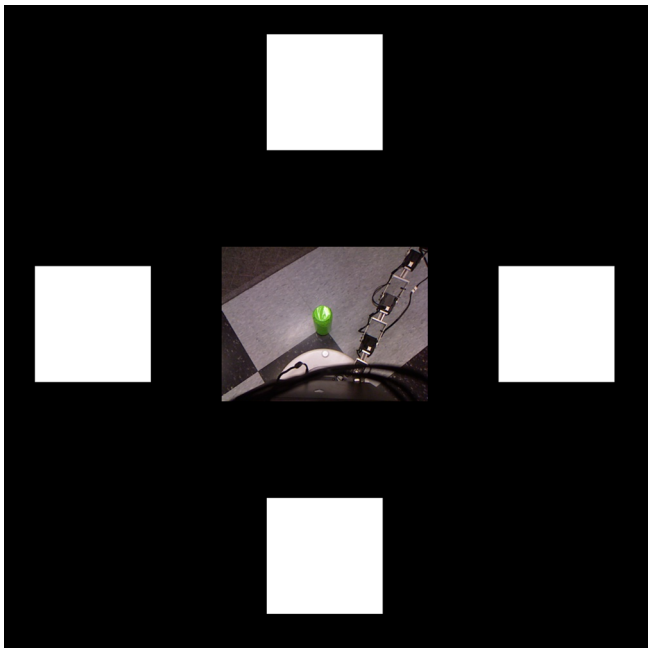

**Figure 4.3. The robot controller app in manual drive mode.**

In the auto mode, users did not have direct control over the robot's motion. Instead, they could select a pre-set target object that was mapped to one of the stimuli. The robot would then search for and move toward that target using the control system described in Chapter 3. The target options were green (left stimulus) and red (right stimulus) to coincide with the targets from the manual control task. Unlike in the manual task, the robot's speed was capped at 300 mm/s instead of 50 mm/s. The up and down stimuli did not map to any actions in this task. A selection action resulted in the robot having its goal cleared and all motion stopped. Users were asked to select one of the targets. After completing the task, the robot was reset and the user was asked to attend to the other target. Finally, the user was asked to select one target, then at some point stop the robot and select the other target.

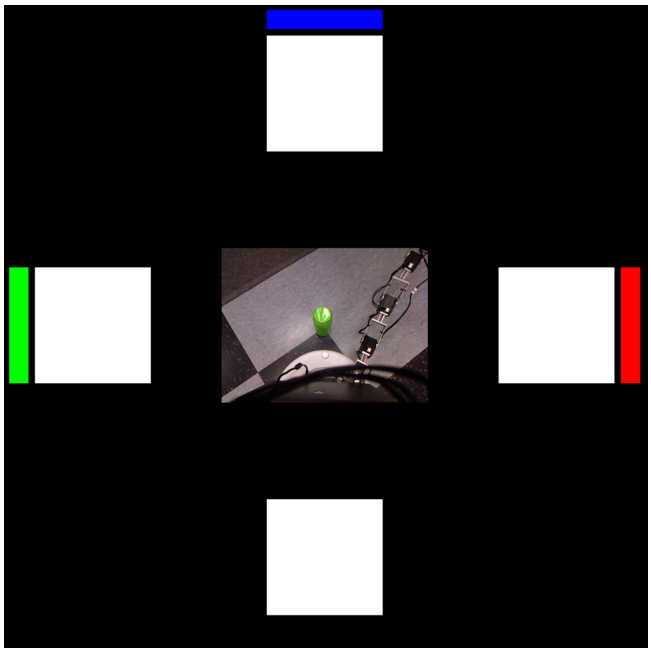

**Figure 4.4. The robot controller app in auto-drive mode**

### 4.3.2. Experimental Procedure

All eight subjects who participated in the experimental sessions described in Chapter 2 also performed these online control tasks. Before beginning, users underwent a training session to generate c-VEP templates for the overt paradigm.

Subjects were presented with the Grid Cursor application first and asked to navigate the blue cursor to the green target then attempt a double-blink selection action. In the event that double-blink detection was not performing well, subjects were allowed to press the space bar instead. If cursor control appeared reliable, subjects were given a few minutes to reach several targets or to explore movement on their own. When cursor control appeared unreliable, the subject was instructed to ignore the green target and look at a particular stimulus repeatedly instead. If control continued to be unreliable, the

session was terminated, otherwise the Grid Cursor app was closed and the Grid Speak app launched.

In the Grid Speak app, subjects were initially asked to go to the grid cell containing the phrase “hello” and select it, followed by “how R U.” Again, depending on control performance, users were either left to select their own choices or directed to additional targets. After a couple minutes with Grid Speak, the app was closed and the robot trials were initiated.

The Calliope robot was initialized and placed near the subject facing the opposite direction. Two target objects, red and green monochromatic cylinders, were placed between 1.5m and 2m in front of and on either side of the robot. First the subjects attempted manual control of the robot. They were initially instructed on what movement action each stimulus corresponded to, then, once the robot was ready, they were told to drive the robot toward either the green or red target and then stop the robot once it was close. The decision as to when the robot was close enough to the target was left up to the subject. After completing one run, they were then instructed to repeat the process but targeting the remaining object.

Following the manual trials, subjects switched to the high-level control version of the robot controller app. They were again instructed as to what actions corresponded to each stimulus and asked to attend to either the red or green target choice.

### 4.4. Results

As there was no single correct choice or path to achieve any of the directed online tasks, specific decoder accuracy numbers could not be computed. Instead, performance on the first two tasks was measured qualitatively through both observation of the system during user engagement and asking the user afterwards how well they felt the system was responding to their intent. Here, user performance was described as Poor (no apparent control), Mediocre (user was able to complete tasks but with multiple errors), Good (clear demonstration of user control but with some errors and several rejected trials), and Excellent (clear demonstration of user control with few errors or rejected trials). Table 4.1 lists the relative performance for each subject.

| Subject | Previous BCI Experience | Overall Peformance |
|---|---|---|
| S1 | N | Poor |
| S2 | N | Poor |
| S3 | N | Excellent |
| S4 | N | Mediocre |
| S5 | Y | Excellent |
| S6 | N | Good |
| S7 | Y | Mediocre |
| S8 | N | Excellent |

**Table 4.1. Relative performance of subjects on online task control.**

Three users achieved Excellent levels of control, one achieved Good, two achieved Mediocre levels, and two were unable to demonstrate any control. Of the two Poor users, neither moved beyond the initial GridCursor task, while all subjects exhibiting greater than Poor levels of control completed all four tasks. Prior BCI experience did not play a factor in predicting success at performing the tasks.

An additional metric that was evaluated was the approximate total time spent engaged with a particular task (Figure 4.5). This is also largely qualitative, as times are determined by counting the number of samples recorded to disk during the time the app was active and dividing by the sample rate of 500 Hz. Some users ended up running an app multiple times for various reasons such as by accidentally exiting prematurely due to a triple eye-blink event, in order to attempt recalibration, or to perform multiple trials as in the case of the robot controller. The median app session time on the far right is most representative of the relative time spent on the apps across all subjects.

Subjects predominantly spent time in the GridCursor app getting used to the interface and finding out whether or not the system appeared to be working. Even though the control interface was identical to GridCursor, finding where phrases were located took time and required that gaze be diverted from the stimuli. Frequently if subjects were demonstrating Good or Excellent control, most of the errors experienced on this task were a result of the cursor moving while they were searching for a phrase to navigate toward.

Unsurprisingly, and with few notable exceptions, the manual drive robot task was much slower than the autonomous drive task. Subject S3's extended time with the auto

mode was not due to failure on their part but rather experimental apparatus problems that had to be rectified. Other subjects, like S8, were able to manually control the robot reasonably well but ran into WiFi connectivity problems where the video feed coming from the robot stopped updating for several seconds at a time. Some subjects who had Excellent control performance indicated that it was more fun to manually drive the robot around instead of simply selecting a target option and watch as the robot charge forward. For those with less accurate control, however, the autonomous version was preferred.

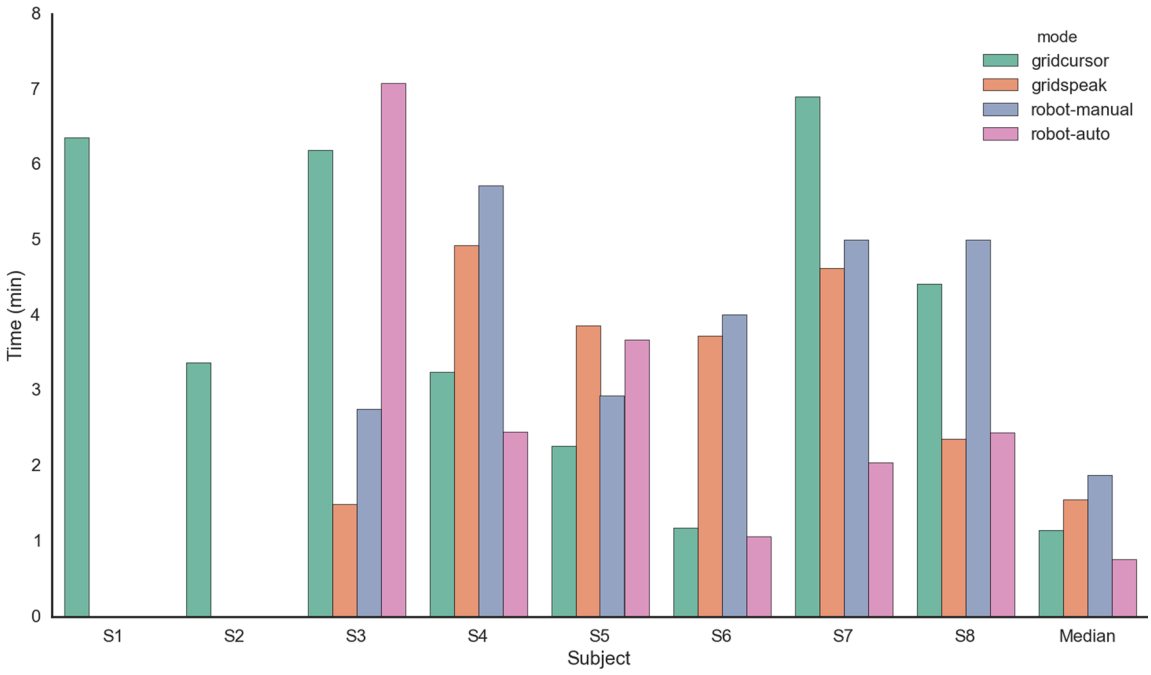


**Figure 4.5. Total time spent engaged with an online task.**
These times are a rough approximation of engagement time derived from the lengths of offline EEG data recorded while each task was activated. The median times on the right are for all sessions across all subjects.

Most subjects were unable to get reliable performance from the double eye-blink detection system and ended up using the space bar instead to generate selection events. Two primary factors contributed to this. First, the eye blink detection mechanism was

calibrated to a single pilot study user. Without adequate training, subjects were unable to consistently achieve the timing needed to trigger intentional double eye-blink detection. Second, the short rest period combined with the potential for an erroneous decision during the ensuing trial presentation meant that even if the double eye-blink was properly detected, the cursor might have moved before the event was registered. Finally, a triple eye-blink corresponded to a stop event, which was accidentally triggered by some subjects attempting a double blink. This caused the current app to halt, necessitating a restart. Improvements would offer an eye-blink calibration app that both tuned the blink detection process to what each user was comfortable with, and also allowed users time to get accustomed to the blink control interface. Extending the duration of the rest period or preventing any decision actions from taking place if a blink had been detected during a trial could also minimize undesirable app behavior during blink attempts.

### 4.5 Discussion

Unlock is a novel BCI application platform that was designed with practical applications and non-researcher developers in mind. Other general purpose BCI platforms, such as BCI2000 (Schalk et al., 2004) or Pyff (Venthur et al., 2010) are geared largely toward research or clinical use by experts. While Unlock has been demonstrated as a powerful tool for conducting both BCI research and application control demonstrations, it has some drawbacks. Notably, the high-precision timing code used for task presentation was written from scratch instead of relying upon existing work in this area such as VisionEgg (Straw 2008), though this was largely due to initial

incompatibilities with how that library functioned in relation to the design of Unlock. With high temporal precision demands, such as flickering visual stimuli on the order of milliseconds, slight deviations crept in due to the non-real time behavior from the host operating system. The timing code employed by Unlock was not smart enough to compensate for this effectively, resulting in the temporal drift that affected the experiments described in Chapter 2.

Controlling the autonomous robot via the BCI was the ultimate goal of this work, and it was successfully demonstrated by all subjects who were capable of using the BCI in this capacity. Only one other study demonstrated online control of a robot using c-VEP (Kapeller et al., 2013), though this was a manually controlled robot that was on a fixed track.

The decoding method used in the online tasks described in this work reflects initial designs that came from pilot study results from two non-naïve subjects and did not incorporate the lessons learned from the work presented in Chapter 2. Future work would update the decoder to use those design considerations. Likewise, the decoder was fixed after training instead of updating or continually adapting, which is an additional change that would be made in future experiments. Despite that, the six subjects who performed all online tasks were able to complete them using a very short decision time window of 1.1s and only after being exposed to the paradigm for the first time during the experimental session.

# 5. CONCLUSION

Three key elements were presented in this work: an adaptive c-VEP-based BCI intended for practical use, a control system for an autonomous assistive robot, and an implementation and demonstration of using the c-VEP BCI to interact with and control the robot amongst other tasks.

The adaptive BCI method found promising results in gaze-dependent tasks. The gaze-independent scenarios, however, were not as successful, though further areas of research were identified. These methods were also only tested on healthy subjects with normal or corrected to normal vision, and a major criticism of visual-based BCI is that motor-impaired subjects who have good vision and gaze control are better off using an eye tracker, while those who lack these abilities may be unable to use the visual-based BCI in the first place. Additional work is needed to evaluate these methods in an LIS population to see if they have real utility.

The robot system was able to perform a number of tasks successfully using motor babbling to improve its performance. These capabilities were still limited, largely due to the poor computer vision methods employed. Improving these and adding additional navigational and reaching sophistication are necessary for the system to be useful in a dynamic environment.

Finally, the Unlock framework was described and how it was used to create BCI user apps, notably the assistive robot interface. The apps as they exist were fairly rudimentary, only offering pre-defined options. A more robust set of choices could be added to enhance the features that are available to the user. Additionally, more automatic

calibration techniques could be employed in order to ensure further individual customization for things like eye-blink detection and confidence thresholding of BCI decisions.

# BIBLIOGRAPHY

Acqualagna, L., and Blankertz, B. (2013). Gaze-independent BCI-spelling using rapid serial visual presentation (RSVP). Clinical Neurophysiology 124, 901–908. doi:10.1016/j.clinph.2012.12.050.

Albus, J.S. (1991). Outline for a theory of intelligence. IEEE Transactions on Systems, Man, and Cybernetics. 21:3, 473–509. doi:10.1109/21.97471

Allison, B. Z., McFarland, D. J., Schalk, G., Zheng, S. D., Jackson, M. M., and Wolpaw, J. R. (2008). Towards an independent brain-computer interface using steady state visual evoked potentials. Clinical Neurophysiology 119, 399–408. doi:10.1016/j.clinph.2007.09.121

Altun, K., and Koku, A.B. (2005). "Evaluation of egocentric navigation methods," in IEEE International Workshop on Robot and Human Interactive Communication, 2005 (ROMAN 2005), 396–401. doi:10.1109/ROMAN.2005.1513811

Andaluz, V., Carelli, R., Salinas, L., Toibero, J.M., and Roberti, F. (2012). Visual control with adaptive dynamical compensation for 3D target tracking by mobile manipulators. Mechatronics 22:4, 491–502. doi:10.1016/j.mechatronics.2011.09.013

Andaluz, V., Roberti, F., Toibero, J.M., and Carelli, R. (2012). Adaptive unified motion control of mobile manipulators. Control Engineering Practice 20:12, 1337–1352. doi:10.1016/j.conengprac.2012.07.008

Bethencourt, J.V.M., Ling, Q., and Fernandez, A.V. (2011). "Controller design and implementation for a differential drive wheeled mobile robot," in Control and Decision Conference (CCDC), 2011 Chinese, 4038–4043. doi:10.1109/CCDC.2011.5968930

Bi, L., Fan, X.-A., and Liu, Y. (2013). EEG-Based Brain-Controlled Mobile Robots: A Survey. IEEE Transactions on Human-Machine Systems 43, 161–176. doi:10.1109/TSMCC.2012.2219046.

Bin, G., Gao, X., Wang, Y., Li, Y., Hong, B., and Gao, S. (2011). A high-speed BCI based on code modulation VEP. Journal of Neural Engineering 8, 025015. doi:10.1088/1741-2560/8/2/025015.

Bouganis, A., and Shanahan, M. (2010). "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity;" in Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), 1–8. doi:10.1109/IJCNN.2010.5596525

Brumberg, J. S., Lorenz, S. D., Galbraith, B. V., and Guenther, F. H. (2012). “The Unlock Project: a Python-based framework for practical brain-computer interface communication “app” development,” in Proceedings of the IEEE Conference of the Engineering in Medicine and Biology Society 2012 (EMBC), 2505–2508. doi:10.1109/EMBC.2012.6346473.

Bullock, D., Grossberg, S. and Guenther, F.H. (1993). A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. Journal of Cognitive Neuroscience 5:4, 408–435. doi:10.1162/jocn.1993.5.4.408

Chao, G., and Dyer, M.G. (1999). "Concentric spatial maps for neural network based navigation," in Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN 99), 1, 144–149. doi:10.1049/cp:19991099

Dasgupta, S., Fanton, M., Pham, J., Willard, M., Nezamfar, H., Shafai, B., et al. (2010). “Brain controlled robotic platform using steady state visual evoked potentials acquired by EEG,” in Proceedings of the Forty Fourth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), 1371–1374. doi:10.1109/ACSSC.2010.5757758.

Du, K.-L., and Swamy, M.N.S. (2014). "Radial basis function networks," in Neural Networks and Statistical Learning (London: Springer), 299–335. doi:10.1007/978-1-4471-5571-3_10

Dudek, G., and Jenkin, M. (2010). Computational principles of mobile robotics, 2nd edition. New York, NY: Cambridge University Press.

Faller, J., Vidaurre, C., Solis-Escalante, T., Neuper, C., and Scherer, R. (2012). Autocalibration and recurrent adaptation: towards a plug and play online ERD-BCI. IEEE Transactions on Neural Systems and Rehabilitation Engineering 20, 313–319. doi:10.1109/TNSRE.2012.2189584

Fernandez-Vargas, J., Pfaff, H. U., Rodriguez, F. B., and Varona, P. (2013). Assisted closed-loop optimization of SSVEP-BCI efficiency. Frontiers in Neural Circuits 7, 27. doi:10.3389/fncir.2013.00027

Fleming, P. (2005). "Implementing a robust 3-dimensional egocentric navigation system" (MS thesis, Vanderbilt University).

Frank, M., Leitner, J., Stollenga, M., Förster, A., and Schmidhuber, J. (2014). Curiosity driven reinforcement learning for motion planning on humanoids. Frontiers in Neurorobotics. 7:25. doi:10.3389/fnbot.2013.00025

Frank, M., Leitner, J., Stollenga, M., Harding, S., Foerster, A., and Schmidhuber, J. (2012). "The modular behavioral environment for humanoids and other robots (mobee)," in 9th International Conference on Informatics in Control, Automation and Robotics (ICINCO).

Franz, M.O., and Mallot, H.A. (2000). Biomimetic robot navigation. Robotics and Autonomous Systems, 30:1, 133–153. doi:10.1016/S0921-8890(99)00069-X

Galbraith, B., Chandler, B., and Versace, M. (2011). "Asimov: middleware for modeling the brain on the irobot create," in PyCon 2011.

Grosse-Wentrup, M., and Contreras-Vidal, J.L. (2007). The role of the striatum in adaptation learning: a computational model. Biological Cybernetics 96:4, 377–388. doi:10.1007/s00422-007-0142-8

Guenther, F.H., and Micci Barreca, D. (1997). "Neural models for flexible control of redundant systems," in Self-organization, Computational Maps, and Motor Control, ed. P.G. Morasso and V. Sanguineti (Amsterdam: North Holland), 383-421.

Hart, S. G., and Staveland, L. E. (1988). “Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research,” in Human Mental Workload Advances in Psychology. eds.P. A. Hancock and N. Meshkati (North-Holland), 139–183. doi:http://dx.doi.org/10.1016/S0166-4115(08)62386-9

Hayes, B., and Scassellati, B. (2013). "Challenges in shared-environment human-robot collaboration," in Proceedings of the Collaborative Manipulation Workshop at the ACM/IEEE International Conference on Human-Robot Interaction (HRI 2013).

He, B., Gao, S., Yuan, H., and Wolpaw, J. R. (2012). “Brain–Computer Interfaces,” in Neural Engineering, ed. B. He (Boston, MA: Springer US), 87–151. doi:10.1007/978-1-4614-5227-0_2

Hild, K., Orhan, U., Erdogmus, D., Roark, B., Oken, B., Purwar, S., et al. (2011). “An ERP-based Brain-Computer Interface for text entry using Rapid Serial Visual Presentation and Language Modeling,” in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations (HLT ’11), 38–43.

Jia, C., Gao, X., Hong, B., and Gao, S. (2011). Frequency and phase mixed coding in SSVEP-based brain--computer interface. IEEE Transactions on Biomedical Engineering 58, 200–206. doi:10.1109/TBME.2010.2068571

Kapadia, M., Singh, S., Hewlett, W., Reinman, G., and Faloutsos, P. (2012). Parallelized egocentric fields for autonomous navigation. The Visual Computer, 28:12, 1209–1227. doi:10.1007/s00371-011-0669-5

Kapeller, C., Hintermuller, C., Abu-Alqumsan, M., Pruckl, R., Peer, A., and Guger, C. (2013). "A BCI using VEP for continuous control of a mobile robot," in Proceedings of the IEEE Conference of the Engineering in Medicine and Biology Society 2013 (EMBC), 5254–5257. doi:10.1109/EMBC.2013.6610734.

Kazemi, M., Gupta, K., and Mehrandezh, M. (2012). "Path planning for image-based control of wheeled mobile manipulators," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 5306–5312. doi:10.1109/IROS.2012.6385898

Kelly, S. P., Lalor, E. C., Finucane, C., McDarby, G., and Reilly, R. B. (2005). Visual Spatial Attention Control in an Independent Brain-Computer Interface. IEEE Transactions on Biomedical Engineering 52, 1588–1596. doi:10.1109/TBME.2005.851510

Klein, C.A., and Huang, C.-H. (1983). Review of pseudoinverse control for use with kinematically redundant manipulators. IEEE Transactions on Systems, Man, and Cybernetics. 13:2, 245–250. doi:10.1109/TSMC.1983.6313123

Kristoffersson, A., Coradeschi, S., and Loutfi, A. (2013). A review of mobile robotic telepresence. Advances in Human-Computer Interaction 2013:3, 1-17. doi:10.1155/2013/902316

Kübler, A., Neumann, N., Wilhelm, B., Hinterberger, T., and Birbaumer, N. (2004). Predictability of Brain-Computer Communication. Journal of Psychophysiology 18, 121–129. doi:10.1027/0269-8803.18.2–3.121

Leeb, R., Friedman, D., Müller-Putz, G. R., Scherer, R., Slater, M., and Pfurtscheller, G. (2007). Self-paced (asynchronous) BCI control of a wheelchair in virtual environments: a case study with a tetraplegic. Computational Intelligence and Neuroscience 2007, 79642–8. doi:10.1155/2007/79642.

Marchetti, M., Onorati, F., Matteucci, M., Mainardi, L., Piccione, F., Silvoni, S., and Priftis, K. (2013). Improving the Efficacy of ERP-Based BCIs Using Different Modalities of Covert Visuospatial Attention and a Genetic Algorithm-Based Classifier. PLoS ONE 8, e53946. doi:10.1371/journal.pone.0053946.t002

Martel, A., Dähne, S., and Blankertz, B. (2014). EEG predictors of covert vigilant attention. Journal of Neural Engineering 11, 035009–12. doi:10.1088/1741-2560/11/3/035009

Maulana, E., Muslim, M.A., and Zainuri, A. (2014). "Inverse kinematics of a two-wheeled differential drive an autonomous mobile robot," in 2014 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), 93–98. doi:10.1109/EECCIS.2014.7003726

McFarland, D. J., McCane, L. M., David, S. V., and Wolpaw, J. R. (1997). Spatial filter selection for EEG-based communication. Electroencephalography and Clinical Neurophysiology 103, 386–394.

Michel, O. (2004). WebotsTM: professional mobile robot simulation. International Journal of Advanced Robotics Systems 1:1, 39–42. doi:10.5772/5618

Middendorf, M., McMillan, G., Calhoun, G., and Jones, K. S. (2000). Brain-computer interfaces based on the steady-state visual-evoked response. IEEE Transactions on Rehabilitation Engineering, 8, 211–214.

Miller, A. (2010). "calibkinect.py." Accessed April 22, 2013. https://github.com/amiller/libfreenect-goodies/blob/master/calibkinect.py.

Nezamfar, H., Orhan, U., Purwar, S., Hild, K., Oken, B., and Erdogmus, D. (2011). Decoding of multichannel EEG activity from the visual cortex in response to pseudorandom binary sequences of visual stimuli. International Journal of Imaging Systems and Technology 21, 139–147. doi:10.1002/ima.20288

Pasqualotto, E., Matuz, T., Federici, S., Ruf, C. A., Bartl, M., Olivetti Belardinelli, M., Birbaumer, N., and Halder, S. (2015). Usability and workload of access technology for people with severe motor impairment: a comparison of brain-computer interfacing and eye tracking. Neurorehabilitation and Neural Repair, 1–8. doi:10.1177/1545968315575611

Saegusa, R., Metta, G., Sandini, G., and Sakka, S. (2009). "Active motor babbling for sensorimotor learning," in IEEE International Conference on Robotics and Biomimetics, 2008 (ROBIO 2008), 794–799. doi:10.1109/ROBIO.2009.4913101

Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., and Wolpaw, J. R. (2004). BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. IEEE Transactions on Biomedical Engineering 51, 1034–1043. doi:10.1109/TBME.2004.827072

Sobol, I.M. (1976). Uniformly distributed sequences with an additional uniform property. USSR Computational Mathematics and Mathematical Physics 16:5, 236–242. doi:10.1016/0041-5553(76)90154-3

Spüler, M., Rosenstiel, W., and Bogdan, M. (2012). Online adaptation of a c-VEP Brain-computer Interface(BCI) based on error-related potentials and unsupervised learning. PLoS ONE 7, e51077. doi:10.1371/journal.pone.0051077

Spüler, M., Walter, A., Rosenstiel, W., and Bogdan, M. (2013). Spatial Filtering Based on Canonical Correlation Analysis for Classification of Evoked or Event-Related Potentials in EEG Data. IEEE Transactions on Neural Systems and Rehabilitation Engineering. doi:10.1109/TNSRE.2013.2290870

Stokbro, K., Umberger, D.K., and Hertz, J.A. (1990). Exploiting neurons with localized receptive fields to learn chaos. Complex Systems 4:6, 603–622.

Straw, A. D. (2008). Vision Egg: An Open-Source Library for Realtime Visual Stimulus Generation. Frontiers in Neuroinformatics 2, 1–10. doi:10.3389/neuro.11.004.2008.

Sutter, E. E. (1992). The brain response interface: communication through visually-induced electrical brain responses. Journal of Microcomputer Applications 15, 31–45. doi:http://dx.doi.org/10.1016/0745-7138(92)90045-7.

Tira-Thompson, E., and Touretzky, D.S. (2011). "The tekkotsu robotics development environment," in IEEE International Conference on Robotics and Automation (ICRA), 2011, 6084–6089.

Treder, M. S., Schmidt, N. M., and Blankertz, B. (2011). Gaze-independent brain-computer interfaces based on covert attention and feature attention. Journal of Neural Engineering 8, 066003. doi:10.1088/1741-2560/8/6/066003

Venthur, B., Scholler, S., Williamson, J., Dähne, S., Treder, M. S., Kramarek, M. T., et al. (2010). Pyff – A Pythonic Framework for Feedback Applications and Stimulus Presentation in Neuroscience. Frontiers in Neuroscience 4. doi:10.3389/fnins.2010.00179.

Vidaurre, C., Sannelli, C., Müller, K.-R., and Blankertz, B. (2011). Co-adaptive calibration to improve BCI efficiency. Journal of Neural Engineering. 8, 025009. doi:10.1088/1741-2560/8/2/025009

Vilaplana, J.M., and Coronado, J.L. (2006). A neural network model for coordination of hand gesture during reach to grasp. Neural Networks 19:1, 12–30. doi:10.1016/j.neunet.2005.07.014

Walter, S., Quigley, C., Andersen, S. K., and Mueller, M. M. (2012). Effects of overt and covert attention on the steady-state visual evoked potential. Neuroscience Letters 519, 37–41. doi:10.1016/j.neulet.2012.05.011

Waytowich, N. R., and Krusienski, D. J. (2015). Spatial decoupling of targets and flashing stimuli for visual brain–computer interfaces. Journal of Neural Engineering 12, 1–10. doi:10.1088/1741-2560/12/3/036006

Xie, H., Li, G., Wang, Y., Fu, Z., and Zhou, F. (2014). Research on visual servo grasping of household objects for nonholonomic mobile manipulator. Journal of Control Science and Engineering. 2014:16, 1-13. doi:10.1155/2014/315396

Yuan, P., Gao, X., Allison, B., Wang, Y., Bin, G., and Gao, S. (2013). A study of the existing problems of estimating the information transfer rate in online brain-computer interfaces. Journal of Neural Engineering 10, 026014. doi:10.1088/1741-2560/10/2/026014

Zalama, E., Gaudiano, P., and Coronado, J.L. (1995). A real-time, unsupervised neural network for the low-level control of a mobile robot in a nonstationary environment. Neural Networks 8:1, 103–123. doi:10.1016/0893-6080(94)00063-R

Zhang, D., Maye, A., Gao, X., Hong, B., Engel, A. K., and Gao, S. (2010). An independent brain-computer interface using covert non-spatial visual selective attention. Journal of Neural Engineering 7, 16010. doi:10.1088/1741-2560/7/1/016010

# VITA