Leveraging Spatiotemporal Relationships of High-frequency Activation in Human Electrocorticographic Recordings for Speech Brain-Computer-Interface

by

Griffin Milsap

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

October, 2018

Abstract

Speech production is one of the most intricate yet natural human behaviors and is most keenly appreciated when it becomes difficult or impossible; as is the case for patients suffering from locked-in syndrome. Burgeoning understanding of the various cortical representations of language has brought into question the viability of a speech neuroprosthesis using implanted electrodes. The temporal resolution of intracranial electrophysiological recordings, frequently billed as a great asset of electrocorticography (ECoG), has actually been a hindrance as speech decoders have struggled to take advantage of this timing information. There have been few demonstrations of how well a speech neuroprosthesis will realistically generalize across contexts when constructed using causal feature extraction and language models that can be applied and adapted in real-time. The research detailed in this dissertation aims primarily to characterize the spatiotemporal relationships of high frequency activity across ECoG arrays during word production. Once identified, these relationships map to motor and semantic representations of speech through the use of algorithms and classifiers that rapidly quantify these relationships in single-trials. The primary hypothesis put forward by this dissertation is that the temporal profile of high frequency activity in ECoG recordings is

a useful feature for speech decoding. These features have rarely been used in state-of-the-art speech decoders, which tend to produce output from instantaneous high frequency power across cortical sites, or rely upon precise behavioral time-locking to take advantage of high frequency activity at several time-points relative to behavioral onset times. This hypothesis was examined in three separate studies. First, software was created that rapidly characterizes spatiotemporal relationships of neural features. Second, semantic representations of speech were examined using these spatiotemporal features. Finally, utterances were discriminated in single-trials with low latency and high accuracy using spatiotemporal matched filters in a neural keyword-spotting paradigm. Outcomes from this dissertation inform implant placement for a human speech prosthesis and provide the scientific and methodological basis to motivate further research of an implant specifically for speech-based brain-computer-interfaces.

Thesis Committee

Primary Reader

Nathan Crone (Primary Advisor)
Professor
Department of Neurology
Johns Hopkins School of Medicine

Alternate Reader

Nitish Thakor
Professor
Department of Biomedical Engineering
Johns Hopkins University Whiting School of Engineering

Acknowledgments

I must first thank Dr. Nathan Crone, Dr. Nitish Thakor, and Dr. Gerwin Schalk. The three of you helped me through the last few years of my education and provided significant help in authoring publications, preparing presentations, and making the transition from problem-solving like an engineer to thinking like a scientist.

My parents, Jeff and Randi Milsap provided the emotional and financial support to get me into a highly respected engineering program at Rensselaer Polytechnic Institute, then to my doctorate program at Johns Hopkins University. You were there to pick me up when my results were inconclusive, and always ready to listen to my victories and defeats while keeping me on-track with my goals.

Significant thanks go out to Phil Keck; although you came into my life near the end of my education, you have made the hardest parts of this process bearable and supported me throughout them.

Many thanks go out to the school district of Cambridge, WI. The world-class primary education I received from the dedicated staff at CHS, NMS, and CES put me on a track to success. Ed Grunden, Pete Degen, and the Wisconsin Center for Academically Talented Youth summer programs significantly

accelerated my trajectory in STEM and deserve significant acknowledgement for setting me up for success.

I also give a fond thanks to my fellow lab mates and colleagues who I shared lunch and banter with on a daily basis for the last six years. I want to thank, among others I'm sure I've missed, Max Collard, Kyle Rupp, Matt Fifer, Chris Coogan, Yujing Wang, Tessy Lal, Dan Candrea, Ryan Smith, Geoffrey Newman, and Qinwan Rabbani.

Finally, I want to give a big thank you to my close friends Michael Batista and Zinnia Xu. Thank you for all of the laughing, eating, hiking, adventuring, movies, nights out, and games. The same can be said for Joe and Alyssa Galaro, Thomas Karathanos, Jason Lee, Will Matern, Steve Tilley, and Heather Kristjanson.

Dedication

I dedicate this dissertation to my cat, Iris.

I'm coming home soon.

I miss you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

There are few behaviors more fundamental to our daily lives than speech; we use the spoken word to communicate ideas and needs, and speech production/perception is the communication channel with the highest bit rate for information transfer. Word production has been investigated extensively using non-invasive neural recording modalities, and speech decoding using intracranial electrophysiology has shown great promise. Our burgeoning understanding of the various cortical representations of language motivates an investigation into the viability of a speech neuroprosthesis using implanted electrodes.

Speech involves very temporally restricted processes that are difficult to characterize in non-invasive recording modalities. The temporal resolution of intracranial electrophysiological recordings, frequently billed as a great asset of the modality, has been difficult to take advantage of using traditional decoding approaches. Observation and quantification of language processes

in streaming recordings with low latency and high specificity is a requirement for cortical speech decoding. Prior studies that suggest development of a real-time speech neuroprosthesis have relied upon non-causal methodology and high trial counts to achieve *statistically significant* speech decoding, but there have been few demonstrations of how well a speech neuroprosthesis will realistically generalize across contexts when constructed using causal feature extraction and language models that can be applied and adapted in real-time.

To address this gap in knowledge and prepare for/inform an eventual neural implant targeting speech neuroprosthesis, this research aims primarily to characterize the spatiotemporal relationships of high frequency activity across electrocorticographic (ECoG) arrays. Once identified, these relationships will be leveraged to understand both motor and semantic representations of speech through the use of algorithms and classifiers that rapidly quantify these relationships in single-trials. This approach is motivated primarily by the dynamics of articulation, suggesting neural representations with intricate temporal evolution patterns across spatially distributed articulator representations. The onset, duration and temporal profile of high frequency activity in ECoG recordings is a useful feature that has seen little application in state-of-the-art decoders, which tend to produce output from instantaneous high frequency power across cortical sites, or rely upon precise temporal locking to take advantage of high frequency activity at several post-stimulus time-points. This research demonstrates that simply characterizing spatiotemporal relationships of high frequency activity allows for the creation of clinically-relevant

maps of language cortex, to the benefit of neurosurgical patient populations. The research described in this dissertation further reveals that taking advantage of these spatiotemporal relationships can yield a better understanding of visual object semantics, and methodology that quantifies these relationships within streaming ECoG recordings can be used to deploy a high sensitivity/specificity, low-latency neural keyword-spotting system.

1.2 Specific Aims

Aim 1: To determine how behavioral events are encoded by spatial and temporal patterns of high frequency neural activity. This was accomplished via the development of a signal processing system capable of performing spatiotemporal functional mapping *using causal methods amenable to real-time computation*. By modernizing aspects of BCI2000, a research and development platform for brain-computer interfaces, experimental data collection was standardized and functional maps were generated for clinical and research purposes. This engineering effort resulted in clinically useful cortical mapping software that was deployed across two medical institutions that was used to create over 200 clinically useful cortical maps. The underlying framework that supports this functional mapping software has been downloaded over 1300 times to date, and enables rapid development and iteration of brain-computer-interfaces that are capable of taking advantage of state-of-the-art visualization technologies.

Aim 2: To map visual semantic attributes of object to spatiotemporal features of high frequency neural activity. Non-invasive neuroimaging studies

have shown that semantic category and attribute information are encoded in neural population activity. Electrocorticography (ECoG) offers several advantages over non-invasive approaches, but the degree to which semantic attribute information is encoded in ECoG responses is not known. Using semantic attribute encoding models, untrained objects were decoded with accuracies comparable to whole-brain functional Magnetic Resonance Imaging (fMRI), and we observed that high frequency activity at basal occipitotemporal electrodes was associated with specific semantic dimensions (manmade-animate, canonically large-small, and places-tools). Individual patient results were in close agreement with reports from other imaging modalities on the time course and functional organization of semantic processing along the ventral visual pathway during object recognition. Temporal envelopes of high frequency activity in ventral visual pathway were assessed for stability across object representations and a neurosemantic decoder facilitating a speech neuroprosthesis was prototyped; demonstrating that while real-time decoding of semantic representations is possible, additional representations of language are necessary to provide a usable communication channel.

Aim 3: To discriminate speech using spatiotemporal matched filter templates in single-trials. As demonstration of a speech-based brain-computer-interface (BCI), a two-step approach to perform neural keyword spotting was evaluated. Neural spatiotemporal matched filters were created from monosyllabic (consonant-vowel, CV) keyword utterances: one keyword utterance, and eleven similar non-keyword utterances. These filters were used in an analog to the acoustic keyword spotting problem, applied for the first time to

neural data. The filter templates were cross-correlated with the neural signal, capturing temporal dynamics of neural activation across recorded cortical sites. Neural vocal activity detection (VAD) was used to identify utterance times and a discriminative classifier was used to determine if these utterances were the keyword or non-keyword speech. Model performance appeared to be highly related to electrode placement and spatial density. Vowel height (/a/ vs /i/) was poorly discriminated in recordings from sensorimotor cortex, but was highly discriminable using neural features from superior temporal gyrus during self-monitoring. The best performing neural keyword detection (5 keyword detections with two false-positives across 60 utterances) and neural VAD (100% sensitivity, 1 false detection per ten utterances) came from high-density (2 mm electrode diameter and 5 mm pitch) recordings from ventral sensorimotor cortex, suggesting the spatial resolution and extent of high-density ECoG arrays may be sufficient for the purpose of speech-based BCIs. The causal, low-complexity algorithms that perform this keyword-spotting utilize spatiotemporal relationships to extract discriminative temporal information in single trials.

1.3 Organization

Chapter 2 of this thesis covers basic background information relevant to discussion of speech decoding from electrocorticographic recordings. Chapter 3 explains relevant methodology and algorithms, as well as details regarding data collection. Chapter 4 explores the development of BCI2000Web and

WebFM as a means of performing real-time spatiotemporal functional mapping. Chapter 5 details a study of semantic representations of visual object naming in addition to a follow-up study of the stability of spatiotemporal encoding between line-drawings and full-color image stimuli. Chapter 6 demonstrates keyword-spotting from neural recordings and how spatiotemporal relationships can be used to perform utterance spotting and keyword discrimination in single trials. Chapter 7 discusses general conclusions and the impact, novelty, and innovation of the work described herein. Appendix A contains software descriptions and algorithm listings from Chapter 3 and 4, and Appendix B contains supplemental figures for Chapter 5 and 6.

Chapter 2

Background

Generation of speech from concept to motor coordination of the articulators is thought to be primarily represented in the cortex of the brain (Indefrey and Levelt, 2004). Semantic, lexical, pre-motor, and auditory representations of speech are activated before the onset of acoustic speech, followed by the generation of motor output for actuating the muscles of speech in the lips, jaw, larynx, and tongue (Indefrey, 2011; Bouchard et al., 2013). The act of recording biosignals from these cortical representations of speech for the purpose of controlling a computer/voice synthesizer, or text transcription for speech synthesis is referred to as a speech-based brain-computer-interface (BCI), referred to as a "speech BCI" throughout this dissertation.

2.1 Motivation for Speech BCI

There are many applications for streaming decode of cortical speech representations. Degeneration of the neural connections between the brain and the muscles of articulation, particularly when caused by amyotrophic lateral

sclerosis (ALS), can result in the complete inability to communicate, let alone produce speech. This disease eventually leads to a "locked-in" state where an alive and very conscious person is trapped in their own body; one that no longer responds to volitional commands to move. Restoration of autonomous communication for these individuals would lead to a dramatic increase in their quality of life. Patient populations with an inability to speak due to other causes may also be able to take advantage of a speech BCI to re-enable speech-based communication.

In order for a speech BCI to have utility for these patient populations, it must be more usable than existing alternatives. Eye-tracking technology can enable a useful communication channel through the use of on-screen keyboards or communication-panels, provided the user has volitional control of their gaze direction. With coarse control of gaze direction, locked-in patients can communicate through their caretakers using communication boards; the caretaker sequentially points to responses on a grid and the patient can look away from the board when the caretaker is pointing at the intended response. Existing BCI technology such as the P300 speller, the mu-rhythm cursor control paradigm, and SSVEP interfaces enable a low-bandwidth autonomous communication channel, with a record information transfer rate (ITR) of 5.3 bps (Chen et al., 2015), as opposed to the 50-60 bps ITR of natural speech (Reed and Durlach, 1998).

There are also prospects for speech BCI beyond disabled patient populations. Automatic speech recognition (ASR) that functions with high accuracy in the presence of acoustic masking would be useful for applications with loud

background noise, particularly in space and military contexts. The presence of subvocal EMG in the muscles of speech suggests that covert speech activates these cortical motor representations similarly during overt and covert speech (Mendes et al., 2008). As such, the ability to communicate using a brain-to-text system could allow for covert conversation.

2.2 Functional Magnetic Resonance Imaging and Optical Techniques

Speech and language neuroscience must be performed in humans, as there are few comparative animal models of speech. As such, non-invasive neural recording modalities are typically employed as they allow a researcher to recruit enough subjects to achieve statistical power. Functional Magnetic Resonance Imaging, (fMRI) permits spatial sampling of the cortical surface at millimeter accuracy, and shows neural activation as a function of blood-flow; a feature known as Blood-oxygen-level dependent imaging (BOLD). As neurons fire action potentials, their metabolic needs are facilitated through changes in blood flow and blood oxygenation – of which BOLD is a direct measure. BOLD tends to modulate slowly; most recordings are done with temporal sampling around once per second. Optical techniques like functional near-infrared spectroscopy (fNIRS) similarly measure this secondary metabolic signal using photon absorption and reflectance without the need for an expensive MRI setup. While these metabolic features can measure neural modulation correlated with particular language processes or even articulation, and they were used to identify areas and representations of interest to the studies in this

dissertation, they do not modulate at rates compatible with real-time speech decoding, limiting their applicability as a target neural recording modality for this research.

2.3 Electrophysiology

Electrophysiology is the act of recording bio-electrical potentials that reflect some underlying physiological process. Electroencephalography (EEG), therefore is the act of recording electrical potentials from the cephalon (the head). The vertebrate nervous system uses electrical potentials to trigger cellular mechanisms that perform computations, relay information, or regulate bodily functions. The central nervous system is responsible for the majority of this computation and coordination. Most importantly, the brain is the origin of neural processing related to speech and word production. The history of electrophysiological approaches to the study of the brain is far more exhaustive than can be addressed in this dissertation, but a few electrophysiological recording techniques are relevant for the development of speech BCI. These techniques differ in the number of neurons they record from and the relevant features that can be extracted.

In brief, when a neuron fires an action potential, the flow of positive ions into the cell causes the extracellular space to be relatively negatively charged. When a population of neurons is firing, these charge gradients are additive and cause larger localized fluctuations in electrical potential. These electrical potentials propagate with a finite velocity through an electrolytic medium consisting of brain tissue, bone, muscle and skin to an electrode where they can

be measured relative to a reference electrode. Electrophysiology is typically amplified using operational amplifiers and digitized using an analog-to-digital converter, also known as an ADC, which samples the amplified analog signal at a regular rate. The potential at each electrode is measured relative to the reference electrode and digitized into a frame; one sample for every electrode. A minimal representation of an EEG recording is an array of unsigned integers with the size $(\text{channels} * \text{time})$ containing a time-series of EEG frames at a specific sampling rate in ADC units. Useful metadata includes a vector of integer offset values and a vector of floating point gain values used to convert the ADC units into microvolts, a list of channel names corresponding to the channels axis, and a sampling rate.

2.3.1 Scalp EEG

Scalp EEG is a non-invasive method of collecting neural electrophysiology. Electrodes placed on the scalp in standard locations are used to measure electric potentials, typically with respect to a mastoid or earlobe reference. The international 10-20 system is used to reference the locations of these electrodes with a standardized name. The spatial filtering properties of the skull limits the channel counts of scalp EEG to 64 channels before approaching spatial supersampling (Lantz et al., 2003). High density scalp EEG has shown promise for source localization (Lantz et al., 2003), but the act of applying conduction gel to more than 64 channels limits the utility and deployability of such systems. These recordings are typically sampled at around 256 Hz due to the fact that the skull and scalp filters the biopotential with a low

pass characteristic, typically putting frequency features above 60 Hz below recording threshold. Evoked responses can be localized roughly to specific lobes of the brain using scalp EEG, and can be recorded with a sufficient signal-to-noise ratio so as to permit ERP-based BCI like the P300 speller and SSVEP-based interfaces (Lotte, Bougrain, and Clerc, 2015). Low-frequency rhythms like the alpha or mu rhythm can be observed in scalp recordings from occipital and motor areas, providing some spectral-based BCI functionality as well (Pfurtscheller and Silva, 1999).

2.3.2 Spike and Multi-unit Recording Devices

Implanted and invasive electroencephalography can take many forms. Historically intracranial recordings had an emphasis on recording spikes from single-unit or multi-unit-activity; extracellular recordings in close proximity to a few neurons ("units"). A recent device that has been used to record multi-unit activity is the cortex-penetrating micro-electrode array (MEA) known as the "Utah" array (Maynard, Nordhausen, and Normann, 1997); a device that stabilizes and facilitates acquisition of spike-recordings for BCI purposes as a replacement for traditional microwire recordings. Spike detection/sorting can extract neural firing rates from these time-series recordings; a useful feature with high time precision and behavioral sensitivity. The spatial extent of these recordings are typically limited, which is not a problem with highly localized neural activity relating to hand and arm movements (Hochberg et al., 2006). Speech and language representations tend to have a much larger spatial extent, covering several lobes, limiting the application of these techniques for speech

decoding.

2.3.3 Electrocorticography (ECoG)

ECoG is realized by placing an array of non-penetrating electrodes on the surface of the brain, typically subdurally using a craniotomy. There are two primary factors that are varied in electrocorticography – the exposed electrode area is directly related to the size of the neural population recorded, and the electrode pitch is related to the spatial sampling frequency. Standard ECoG array implants consist of a 64 channel grid of electrodes with 2 mm exposed diameter and 1 cm pitch. Recently, increasing channel counts and an interest in higher spatial sampling has led to the development of a 128 channel grid of electrodes with 1 mm exposed electrode diameter and 5 or 3 mm pitch (also known as high-density "HD" ECoG). These high density arrays sample with much higher spatial resolution and capture neural activity that would exist between recording sites on a standard ECoG array. Micro-ECoG arrays consisting of 16 surface microwires with 75 um diameter and 1mm pitch have also been investigated throughout this research – a recording modality with limited spatial extent, but spatial resolution that approaches spatial supersampling at the cortical level (Slutzky et al., 2010).

ECoG – even at the micro-ECoG scale – does not capture spiking activity; the signals are more closely related to "local field potentials" as recorded by microwire recordings (Kellis et al., 2010). The population activity in ECoG recordings contains the frequency-specific features of scalp EEG with additional frequency resolution to observe high frequency (70+ Hz) activity.

Modulation in high frequency activity, also called "high gamma activity" is correlated with neural firing rates in the underlying cortex (Ray et al., 2008), and modulation of high gamma power (70-110 Hz), z-scored to a resting baseline period tends to be used as the neural feature of choice in these recordings.

2.3.4 Neural Features

Traditionally, time-series electrophysiology was used as the input feature to models; repeated averaging across trials of the raw EEG yields a temporal waveform referred to as an evoked response potential, or ERP. Components of these evoked responses including their amplitude, and timing of their peaks has been used to evaluate models and test hypotheses about speech and language, with a well constructed paradigm. These ERPs are easy to measure using scalp recordings because the dominant features of an ERP are primarily low-frequency in nature. Frequency features are most-commonly used in ECoG recordings. Isolation of high gamma activity in particular has repeatedly appeared as the most useful feature, even in studies where other frequencies have been investigated.

2.4 Cortical Representations of Speech/Language

Indefrey, 2011 performed a meta-study on a large number of studies pertaining to picture-naming tasks. They characterized the cortical process of speech output from picture input as several processes that occur within specific time ranges relative to picture onset, in particular cortical locations (Indefrey, 2011). Semantic, auditory and motor representations are of great interest to the

research discussed in this dissertation.

2.4.1 Semantic

Within the context of Indefrey, 2011, a semantic representation is present shortly after picture presentation. The cortical representation of this semantic access has been well investigated in FMRI by two high-impact studies. Mitchell et al., 2008 presented line drawings from twelve categories to subjects during a fMRI imaging sequence during which they were asked to rehearse mental imagery for several seconds. Nouns were transformed into their vector representation within a 218 dimensional semantic space, and a mapping between BOLD values in voxels and semantic dimensions was regressed. The model showed statistically significant generalization to held out objects in rank-accuracy statistics, indicating that the semantic space captures some aspect of neural semantic dimensions, and that the regressed model can be used to transform neural representations into a semantic representation. This particular work was foundational to the second aim of this thesis; which aimed to perform the same study using the temporal resolution of ECoG as a trade-off for the spatial resolution of fMRI.

Huth et al., 2016 investigated neural activation with significant correlation to semantically loaded stories. They did this by first replaying two hours of stories to subjects in an fMRI scanner. They calculated co-occurrences between semantically loaded words within the story and 1000 manually selected words. BOLD was regressed onto these time-varying word-loadings, forming an encoding model capable of predicting BOLD activity given a semantic vector.

Time-varying semantic loadings were generated from a held-out test story and used to predict bold activity. The correlation between these sequences was used to map areas of the brain where semantic activity can be used to statistically significantly predict neural features. This influential research indicated wide-spread semantic representations, and decomposed these semantic loadings into interpretable principle components of neural semantic activations.

Targeting a semantic representation of language for real-time decoding is difficult for several important reasons. It may not be possible to perform a linear transformation of a neural feature vector into a semantic manifold unless the semantic space used for the regression captures what is neurally relevant for a semantic representation. There have only been attempts to map neural activation to semantic spaces based on intuition or text corpora; a neural semantic space has not been investigated, and is unlikely to generalize across subjects. Furthermore, imaging studies indicate the representation of semantics is broadly spread throughout the cortex bilaterally (Huth et al., 2016), and with ECoG we tend to only sample a small portion of one hemisphere. The temporal encoding of semantics in ECoG recordings is yet unknown, which is explicitly addressed in a study described in Chapter 5.

Assuming a decoder can actually produce meaningful semantic information from single trials of ECoG data, it stands to reason that this content is only encoding the semantic loading of a thought, or a portion thereof, and cannot be sequenced into a meaningful sentence without syntax and noun-verb relationships; thought to be calculated temporally later and in a different

cortical location. Furthermore, it has not yet been shown that these semantic representations exist outside the context of extrinsic stimulation - all studies of semantic representations to date have addressed the concept of receptive semantics, and have not probed semantic representations in produced speech. Semantic representations have not even been shown to be modality independent; that is to say that a picture of a dog may evoke an orthogonal semantic representation to the word dog, either visually or aurally presented.

Semantic decoding using electrocorticography has been approached in a few existing studies. Liu et al., 2009 investigated visual object recognition, showing that object category information could be decoded from single trials of ECoG recordings from temporal and occipital lobe, suggesting the presence of visual-semantic integration in these areas is tied to semantic-category recognition. Wang et al., 2011 confirmed this result by predicting object category from single trials using response-locked high gamma activity in left inferior frontal gyrus and posterior STG. Chen et al., 2015 observed that stimulus-locked activity in ventral temporal lobe encodes semantic representations that are *uncorrelated* to the visual similarity of the images presented, while simultaneously controlling for the phonological similarity of the names of the objects shown. These studies were motivation for an investigation for a speech-BCI based on semantic representations, detailed further in Chapter 5.

2.4.2 Motor

Motor representations of speech are a bit more concrete. Sensorimotor cortex, consisting of M1 located in precentral gyrus and S1 located in postcentral

gyrus consist of neural populations whose firing rates correspond to activity in individual muscles, and sensory afferents, respectively. The most ventral aspect of sensorimotor cortex, vSMC, encodes sensory and motor representations of speech articulators and the vocal tract. Activity in this part of the brain directly corresponds to movement and sensation within the vocal tract and speech decoding from neural features in area alone has been shown to be possible.

Bouchard et al., 2013 investigated this area in a 2013 study using high density ECoG arrays, showing that the neural representation of articulators is present in motor cortex at a fine-scale, and principle components of this activity can be decoded into a neural state-space that directly encodes consonant-vowel syllable gestures. Mugler et al., 2014 first classified all english phonemes from vSMC, then in a follow-up study, suggested that although neural activity in this area of the brain is correlated with discrete aspects of articulation such as phonemes, the actual encoding is more related to the trajectory of articulators – such as the aperture of the vocal tract and the tongue height, etc (Mugler et al., 2015). This prior work strongly suggests that if vSMC is spatially sampled at a sufficient resolution, it may be possible to decode speech. This said, there have been very few studies that explore speech decoding in single-trials - instead showing that neural features averaged across repeated words can be used to discriminate articulations with statistical significance.

2.4.3 Auditory

Auditory research has a richer tradition in neuroscience with more animal models and basic science that has informed and characterized the low level aspects of audition through primary auditory cortex. It is non-intuitive that auditory representations are even relevant for decoding speech production – a significant part of producing speech involves self-monitoring and feedback-based control. It has repeatedly been shown that distorting or changing auditory feedback during speech can involuntarily alter speech production (Houde and Jordan, 1998). In fact, the dynamics of speech production are so rapid that the sensory processing streams responsible for monitoring these dynamics are simply not fast enough to form a stable control system without some form of forward modeling (Houde and Jordan, 1998). The existence of an auditory efference copy is actually the topic of active research and may in-fact form the basis of internal monologue or covert speech.

Hickok and Poeppel, 2007 have thoroughly reviewed the auditory literature for speech processing in humans. The dual-stream model they propose and characterize is the result of a comprehensive meta-analysis of decades of research in auditory neuroscience. Mesgarani et al., 2014 was the first to explore the phonemic representation of perceived speech along superior temporal gyrus using ECoG. These representations are thought to be intact for self-audition, though they may be repressed slightly in a phenomenon known as speech-induced suppression, or SIS. It may be necessary to decode auditory representations of speech during self-audition in order to produce a usable speech BCI. Unfortunately, they are not typically selective for self-monitoring

of speech, and will also be active for perception of extrinsic speech.

Several ECoG studies have detailed auditory representations of speech are captured by the population-level activity resolved by ECoG. These studies observed that the STG exhibits a distributed population response in the high gamma band (70 – 150 Hz) to continuously perceived speech with a robust anterior-posterior spatial distribution of spectrotemporal tuning encoding acoustic cues in a nonlinear fashion and a combination of cues (Mesgarani et al., 2014; Hullett et al., 2016). In particular, the posterior STG is more highly tuned for temporally-fast-varying speech sounds with relatively constant energy across the frequency axis (i.e., low spectral modulation), whereas the anterior STG is more highly tuned for temporally-slow-varying speech sounds with a high degree of spectral variation across the frequency axis (i.e., high spectral modulation)(Hullett et al., 2016). Together, these studies suggest that ECoG adequately captures spectrotemporal tuning of the STG to speech organized by acoustic features rather than by discrete phoneme categories.

2.5 Speech Decoding

Attempts to decode speech from streaming neural recordings are few and far inbetween. Guenther et al., 2009 was one of the first to control a real-time speech synthesizer to select vowels using an invasive glass neurotrophic electrode. They were able to increase the subject's accuracy by 25% (from 45% to 70%) and decrease his average endpoint error in a block paradigm by 46% using auditory feedback from the decoded sound, suggesting that feedback may be key to successful speech BCI training. Brumberg et al.,

2011 demonstrated this interface works similarly with "intended speech" in a follow-up study by providing basic speech control to a subject with locked-in-syndrome.

The Herff et al., 2015 "Brain-to-Text" system was the first demonstration of statistically significant ECoG-based speech decoding, by transcribing overt speech production from widely distributed brain areas directly into text with word and phone error rates ranging 25 to 60% and 50 to 80% for vocabularies ranging 10 to 100 words respectively (all significantly above chance). Though the performance was modest, this study served as a useful benchmark to understand how performance might scale with increasing vocabularies. Additionally, follow-up analysis demonstrates that these above-chance results extend to decoding using only pre-phonatory activity, a rough analog for decoding covert speech intention (Herff et al., 2017). The distinction between covert and overt representations of speech are still debated with many ongoing studies attempting to address these differences.

2.6 Covert and Overt Speech

Several ECoG studies explicitly study the relationship between overt and covert speech. Magrassi et al., 2015 observed highly correlated high frequency neural activity present prior to articulation in both overt and covert reading, while Brumberg et al., 2016 observed that this activity progresses from the speech motor areas in ventral precentral gyrus and Broca's area to auditory speech areas in the middle temporal gyrus (MTG) and middle and posterior

STG (including the A1 and Wernicke's area) during both conditions. Additionally, Martin et al., 2014 observed that a model built to reconstruct auditory speech features in overt speech could reconstruct these same features in covert speech. These results suggest a shared auditory and articulatory substrate between overt and covert speech with a spatiotemporal progression of activity aligned to speech production.

However, further studies suggest a differential contribution of auditory and articulatory areas to overt and covert speech production over time. In an MEG study by Tian and Poeppel, 2010, it was observed that overt and covert speech articulation are fundamentally different but that overt and covert perception are highly similar. Specifically, they observed that activation during covert articulation extends to posterior parietal areas rather than the motor cortex, as in overt articulation, but that auditory activation in bilateral temporal areas is present during both overt and covert speech perception. Interestingly, covert speech articulation also displays an "auditory-like" response following articulation highly similar to that observed during hearing, suggesting the presence of an "efference copy," a feed-forward prediction of the perceptual outcome of an action.

These findings were supported in ECoG studies by Pei et al., 2011b and Leuthardt et al., 2012, in which both overt and covert repetition tasks using both auditory and visual cues (4 combinations) were performed. In these studies, areas within the STG, including the planum temporale within the A1 (BA41/42) and Wernicke's area (BA22), showed more pronounced activation during covert speech than in the M1, with some disagreement about the

premotor cortex. However, the disagreement about the premotor cortex was elucidated in ECoG studies by Pei et al., 2011a and Ikeda et al., 2014 wherein phonemes were studied in isolation. High gamma activity (70 – 110 Hz) in the premotor cortex and the STG contributed most to decoding performance for covert speech.

References

Indefrey, P and W. J. M Levelt (2004). "The spatial and temporal signatures of word production components". In: *Cognition*. Towards a New Functional Anatomy of Language 92.1, pp. 101–144. ISSN: 0010-0277. DOI: 10.1016/j.cognition.2002.06.001.

Indefrey, Peter (2011). "The Spatial and Temporal Signatures of Word Production Components: A Critical Update". In: *Frontiers in Psychology* 2. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2011.00255.

Bouchard, Kristofer E., Nima Mesgarani, Keith Johnson, and Edward F. Chang (2013). "Functional organization of human sensorimotor cortex for speech articulation". In: *Nature* 495.7441, pp. 327–332. ISSN: 1476-4687. DOI: 10.1038/nature11911.

Chen, Xiaogang, Yijun Wang, Masaki Nakanishi, Xiaorong Gao, Tzyy-Ping Jung, and Shangkai Gao (2015). "High-speed spelling with a noninvasive brain–computer interface". In: *Proceedings of the National Academy of Sciences of the United States of America* 112.44, E6058–E6067. ISSN: 0027-8424. DOI: 10.1073/pnas.1508080112.

Reed, Charlotte M. and Nathaniel I. Durlach (1998). "Note on Information Transfer Rates in Human Communication". In: *Presence: Teleoperators and Virtual Environments* 7.5, pp. 509–518. ISSN: 1054-7460. DOI: 10.1162/105474698565893.

Mendes, José AG, Ricardo R. Robson, Sofiane Labidi, and Allan Kardec Barros (2008). "Subvocal Speech Recognition Based on EMG Signal Using Independent Component Analysis and Neural Network MLP". In: *2008 Congress on Image and Signal Processing*. Sanya, China: IEEE, pp. 221–224. ISBN: 978-0-7695-3119-9. DOI: 10.1109/CISP.2008.741.

Lantz, G, R Grave de Peralta, L Spinelli, M Seeck, and C. M Michel (2003). "Epileptic source localization with high density EEG: how many electrodes are needed?" In: *Clinical Neurophysiology* 114.1, pp. 63–69. ISSN: 1388-2457. DOI: 10.1016/S1388-2457(02)00337-1.

Lotte, Fabien, Laurent Bougrain, and Maureen Clerc (2015). "Electroencephalography (EEG)-Based Brain-Computer Interfaces". In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. American Cancer Society, pp. 1–20. ISBN: 978-0-471-34608-1. DOI: 10.1002/047134608X.W8278.

Pfurtscheller, G. and F. H. Lopes da Silva (1999). "Event-related EEG/MEG synchronization and desynchronization: basic principles". In: *Clinical Neurophysiology: Official Journal of the International Federation of Clinical Neurophysiology* 110.11, pp. 1842–1857. ISSN: 1388-2457.

Maynard, Edwin M., Craig T. Nordhausen, and Richard A. Normann (1997). "The Utah Intracortical Electrode Array: A recording structure for potential brain-computer interfaces". In: *Electroencephalography and Clinical Neurophysiology* 102.3, pp. 228–239. ISSN: 0013-4694. DOI: 10.1016/S0013-4694(96)95176-0.

Hochberg, Leigh R., Mijail D. Serruya, Gerhard M. Friehs, Jon A. Mukand, Maryam Saleh, Abraham H. Caplan, Almut Branner, David Chen, Richard D. Penn, and John P. Donoghue (2006). "Neuronal ensemble control of prosthetic devices by a human with tetraplegia". In: *Nature* 442.7099, pp. 164–171. ISSN: 1476-4687. DOI: 10.1038/nature04970.

Slutzky, Marc W., Luke R. Jordan, Todd Krieg, Ming Chen, David J. Mogul, and Lee E. Miller (2010). "Optimal Spacing of Surface Electrode Arrays for Brain Machine Interface Applications". In: *Journal of neural engineering* 7.2, p. 26004. ISSN: 1741-2560. DOI: 10.1088/1741-2560/7/2/026004.

Kellis, Spencer, Kai Miller, Kyle Thomson, Richard Brown, Paul House, and Bradley Greger (2010). "Decoding spoken words using local field potentials recorded from the cortical surface". In: *Journal of Neural Engineering* 7.5, p. 056007. ISSN: 1741-2552. DOI: 10.1088/1741-2560/7/5/056007.

Ray, Supratim, Nathan E. Crone, Ernst Niebur, Piotr J. Franaszczuk, and Steven S. Hsiao (2008). "Neural Correlates of High-Gamma Oscillations (60–200 Hz) in Macaque Local Field Potentials and Their Potential Implications in Electrocorticography". In: *Journal of Neuroscience* 28.45, pp. 11526–11536. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.2848-08.2008.

Mitchell, Tom M., Svetlana V. Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L. Malave, Robert A. Mason, and Marcel Adam Just (2008). "Predicting Human Brain Activity Associated with the Meanings of Nouns". In: *Science* 320.5880, pp. 1191–1195. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1152876.

Huth, Alexander G., Wendy A. de Heer, Thomas L. Griffiths, Frédéric E. Theunissen, and Jack L. Gallant (2016). "Natural speech reveals the semantic

maps that tile human cerebral cortex". In: *Nature* 532.7600, pp. 453–458. ISSN: 1476-4687. DOI: 10.1038/nature17637.

Liu, Hesheng, Yigal Agam, Joseph R. Madsen, and Gabriel Kreiman (2009). "Timing, timing, timing: fast decoding of object information from intracranial field potentials in human visual cortex". In: *Neuron* 62.2, pp. 281–290. ISSN: 1097-4199. DOI: 10.1016/j.neuron.2009.02.025.

Wang, Wei, Alan D. Degenhart, Gustavo P. Sudre, Dean A. Pomerleau, and Elizabeth C. Tyler-Kabara (2011). "Decoding semantic information from human electrocorticographic (ECoG) signals". In: *Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference* 2011, pp. 6294–6298. ISSN: 1557-170X. DOI: 10.1109/IEMBS.2011.6091553.

Mugler, Emily M., James L. Patton, Robert D. Flint, Zachary A. Wright, Stephan U. Schuele, Joshua Rosenow, Jerry J. Shih, Dean J. Krusienski, and Marc W. Slutzky (2014). "Direct classification of all American English phonemes using signals from functional speech motor cortex". In: *Journal of Neural Engineering* 11.3, p. 035015. ISSN: 1741-2552. DOI: 10.1088/1741-2560/11/3/035015.

Mugler, E. M., M. Goldrick, J. M. Rosenow, M. C. Tate, and M. W. Slutzky (2015). "Decoding of articulatory gestures during word production using speech motor and premotor cortical activity". In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5339–5342. DOI: 10.1109/EMBC.2015.7319597.

Houde, John F. and Michael I. Jordan (1998). "Sensorimotor Adaptation in Speech Production". In: *Science* 279.5354, pp. 1213–1216. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.279.5354.1213.

Hickok, Gregory and David Poeppel (2007). "The cortical organization of speech processing". In: *Nature Reviews Neuroscience* 8.5, pp. 393–402. ISSN: 1471-0048. DOI: 10.1038/nrn2113.

Mesgarani, Nima, Connie Cheung, Keith Johnson, and Edward F. Chang (2014). "Phonetic Feature Encoding in Human Superior Temporal Gyrus". In: *Science* 343.6174, pp. 1006–1010. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1245994.

Hullett, Patrick W., Liberty S. Hamilton, Nima Mesgarani, Christoph E. Schreiner, and Edward F. Chang (2016). "Human Superior Temporal Gyrus Organization of Spectrotemporal Modulation Tuning Derived from Speech Stimuli".

In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 36.6, pp. 2014–2026. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.1779-15.2016.

Guenther, Frank H., Jonathan S. Brumberg, E. Joseph Wright, Alfonso Nieto-Castanon, Jason A. Tourville, Mikhail Panko, Robert Law, Steven A. Siebert, Jess L. Bartels, Dinal S. Andreasen, Princewill Ehirim, Hui Mao, and Philip R. Kennedy (2009). "A Wireless Brain-Machine Interface for Real-Time Speech Synthesis". In: *PLOS ONE* 4.12, e8218. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0008218.

Brumberg, Jonathan S., E. Joe Wright, Dinal S. Andreasen, Frank H. Guenther, and Philip R. Kennedy (2011). "Classification of intended phoneme production from chronic intracortical microelectrode recordings in speech motor cortex". In: *Frontiers in Neuroscience* 5. ISSN: 1662-453X. DOI: 10.3389/fnins.2011.00065.

Herff, Christian, Dominic Heger, Adriana de Pesters, Dominic Telaar, Peter Brunner, Gerwin Schalk, and Tanja Schultz (2015). "Brain-to-text: decoding spoken phrases from phone representations in the brain". In: *Frontiers in Neuroscience* 9. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00217.

Herff, Christian, Adriana de Pesters, Dominic Heger, Peter Brunner, Gerwin Schalk, and Tanja Schultz (2017). "Towards Continuous Speech Recognition for BCI". In: *Brain-Computer Interface Research: A State-of-the-Art Summary 5*. Ed. by Christoph Guger, Brendan Allison, and Junichi Ushiba. SpringerBriefs in Electrical and Computer Engineering. Cham: Springer International Publishing, pp. 21–29. ISBN: 978-3-319-57132-4. DOI: 10.1007/978-3-319-57132-4_3.

Magrassi, Lorenzo, Giuseppe Aromataris, Alessandro Cabrini, Valerio Annovazzi-Lodi, and Andrea Moro (2015). "Sound representation in higher language areas during language generation". In: *Proceedings of the National Academy of Sciences* 112.6, pp. 1868–1873. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1418162112.

Brumberg, Jonathan S., Dean J. Krusienski, Shreya Chakrabarti, Aysegul Gunduz, Peter Brunner, Anthony L. Ritaccio, and Gerwin Schalk (2016). "Spatio-Temporal Progression of Cortical Activity Related to Continuous Overt and Covert Speech Production in a Reading Task". In: *PLOS ONE* 11.11, e0166872. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0166872.

Martin, Stéphanie, Peter Brunner, Chris Holdgraf, Hans-Jochen Heinze, Nathan E. Crone, Jochem Rieger, Gerwin Schalk, Robert T. Knight, and Brian N. Pasley (2014). "Decoding spectrotemporal features of overt and covert

speech from the human cortex". In: *Frontiers in Neuroengineering* 7. ISSN: 1662-6443. DOI: 10.3389/fneng.2014.00014.

Tian, Xing and David Poeppel (2010). "Mental imagery of speech and movement implicates the dynamics of internal forward models". In: *Frontiers in Psychology* 1. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2010.00166.

Pei, Xiaomei, Eric C. Leuthardt, Charles M. Gaona, Peter Brunner, Jonathan R. Wolpaw, and Gerwin Schalk (2011b). "Spatiotemporal dynamics of electrocorticographic high gamma activity during overt and covert word repetition". In: *NeuroImage* 54.4, pp. 2960–2972. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2010.10.029.

Leuthardt, Eric C., Xiao-Mei Pei, Jonathan Breshears, Charles Gaona, Mohit Sharma, Zac Freudenberg, Dennis Barbour, and Gerwin Schalk (2012). "Temporal evolution of gamma activity in human cortex during an overt and covert word repetition task". In: *Frontiers in Human Neuroscience* 6. ISSN: 1662-5161. DOI: 10.3389/fnhum.2012.00099.

Pei, Xiaomei, Dennis L. Barbour, Eric C. Leuthardt, and Gerwin Schalk (2011a). "Decoding vowels and consonants in spoken and imagined words using electrocorticographic signals in humans". In: *Journal of Neural Engineering* 8.4, p. 046028. ISSN: 1741-2552. DOI: 10.1088/1741-2560/8/4/046028.

Ikeda, Shigeyuki, Tomohiro Shibata, Naoki Nakano, Rieko Okada, Naohiro Tsuyuguchi, Kazushi Ikeda, and Amami Kato (2014). "Neural decoding of single vowels during covert articulation using electrocorticography". In: *Frontiers in Human Neuroscience* 8. ISSN: 1662-5161. DOI: 10.3389/fnhum.2014.00125.

Chapter 3

Common Methodology and Data Recording

This chapter details some of the methodology and aspects of signal processing that are common to ECoG recordings throughout this dissertation; some of these methods are referenced in later chapters. This chapter first describes technical details of electrophysiological data recording and the hardware/-software used to acquire usable data. Methodology for localizing cortical electrode locations is discussed, as well as some of its limitations. A breakdown of state-of-the-art ECoG pre-processing and feature extraction concludes this section.

3.1 Electrodes and ECoG Arrays

A great deal of innovation in the configuration and density of cortical electrophysiology has occurred throughout this thesis work. Electrocorticographic recording devices consist of an arrangement of platinum-iridium electrodes

embedded in a flexible medical-grade silicone rubber, or "silastic". Platinum-iridium is chosen as the electrode metal of choice due to its ability to facilitate electrical stimulation without deposition of biologically toxic ions into the underlying neural tissue. The exposed electrode area and inter-electrode spacing are under a great deal of variability in current devices. Exposed electrode area is directly related to the size of the neural population sampled by the electrode, with larger electrodes recruiting responses from larger amounts of neural units. Decreasing the inter-electrode spacing increases the spatial density of sampling in the array, but decreases the extent of coverage. Striking an optimal balance of spatial density and extent is key to observing neural signals of interest, especially when it comes to the widespread activation during speech and language.

3.2 Data Recording

Acquisition of electrophysiological data was performed by the signal-source module of BCI2000. The two amplifiers used throughout this work were not natively supported by the BCI2000 software at the time, so custom acquisition code had to be written to interact with these amplifiers. The relevant code is provided in Appendix A.

3.2.1 Blackrock Neuroport

The Blackrock Neuroport System (Blackrock Microsystems, Salt Lake City, UT) is a 128 channel amplifier with 16 bit ADCs and a native 30 kHz sampling rate. The system consists of a front-end amplifier which is powered separately from

a neural-signal-processor (NSP) with which it communicates via a full-duplex fiber optic link. The front-end amplifier digitizes the analog recordings and transmits raw data to the NSP for downsampling and filtering prior to UDP multicast broadcast to an acquisition PC. The acquisition PC runs a software called Central that is capable of communicating to the NSPs over an ethernet-based network interface for configuration, acquisition, spike processing, and data recording. Blackrock provides a C/C++ API called "cbsdk" that was used to communicate to the Central software using a shared-memory interface for acquisition of the raw electrophysiology. An acquisition module was written that uses this API to acquire data in a real-time compatible way from the Blackrock hardware/software; reproduced in its entirety in Appendix A, but also released as part of the BCI2000 project as a signal source contribution. This amplifier supports a number of sampling rates, but most recordings from this amplifier utilized the 1000 Hz sampling group.

Recent EMU patients have had more than 128 electrodes implanted, so a second neuroport was used to acquire additional synchronized data, for a total of 256 channels. A buffered signal splitter was used to split the signal from the clinical acquisition system and route it into the Blackrock system without affecting the clinical data. ECoG micro-arrays required high input impedance to minimize signal distortion; the Blackrock system has a 10 GΩ input impedance, as opposed to the 200 MΩ input impedance of the clinical system, so these electrodes were not split, and were instead directly acquired by the Blackrock system. This system was used solely for research purposes, and did not affect clinical data collection.

3.2.2 Nihon Kohden Neurofax EEG-1200

The EEG-1200 (Nihon Kohden, Tomioka, Japan), is an electrophysiological recording system; the included JE-120-256 amplifier has 16 bit ADCs and a native 10 kHz sampling rate. The amplifier streams digitized samples using a proprietary network protocol over ethernet to a recording computer running the Neuroworkbench software. Although this system was installed as the clinical recording system, a read-only data stream was also made available at a 2 kHz sampling rate. This "research stream" can be acquired in real-time using a shared-memory interface through the use of a proprietary dynamic-link library. A source module for acquisition from this interface was written for BCI2000 and has been reproduced in its entirety in Appendix A, but also has been released as part of the BCI2000 project as a signal source contribution. As mentioned previously, this amplifier was deployed as the clinical recording system. A buffered splitter box connection terminal was used to electrically split the analog signal into the Blackrock neuroport system for research purposes.

3.2.3 Practical Considerations and Limitations

Both of these amplifiers were recording AC-coupled signals with a high pass characteristic around 0.5 Hz. Although DC coupled recordings could be useful for characterizing absolute voltage differences, most clinical amplifiers do not perform DC recordings to avoid clipping signals with large DC offsets. Another important consideration is that these amplifiers tend to reserve around six bits of recording headroom to avoid saturating the ADCs and

clipping during large spiking events. As such, these recordings tend to be captured with about 10 bits of resolution. The "pink noise" nature of biological recordings leads to a falloff of signal amplitude with respect to the inverse of its frequency; also known as a "$1/f$" characteristic (Bédard, Kröger, and Destexhe, 2006). The amplitude of "high gamma" signals between 70 and 110 Hz within these recordings tends fluctuate with around 5 bits of resolution in these recordings, which is sufficient for observing modulation of high frequency amplitude changes, but our ability to resolve high frequency signals reaches a so-called "noise-floor" around 250 Hz in recordings from these amplifiers. Higher frequency signals could exist in these recordings, but our ability to resolve these phenomena is limited given the circumstances. In an effort to conserve space and decrease processing time in accordance with the aforementioned considerations, recordings were temporally downsampled to 1 kHz prior to prior to logging in a data file.

3.3 Electrode Localization

Electrodes were localized to cortical locations by co-registration of a pre-operative nuclear magnetic resonance image (MRI) with a post-operative computed tomography (CT) scan. The pre-operative MRI captures the undisturbed cortical anatomy with great detail and high resolution, free of imaging artifact from implanted metal electrodes. The postoperative CT is a high resolution scan with the same anatomical landmarks as the preoperative MRI, but with high resolution detail of the electrode locations. A linear transformation from the post-operative space to the preoperative space can map the

electrode positions to the preoperative MRI space for visualization on a skull-stripped MRI. This process is mostly automated and was carried out using two purpose-built software packages; Bioimage Suite (Papademetris et al., 2006) and Freesurfer (Fischl, 2012). Bioimage Suite works primarily with volumetric images whereas Freesurfer performs a meshing operation on the cortical surface, creating a triangularized mesh for efficient rendering. This method of capturing brain anatomy is less optimal for visualizing electrodes implanted within the brain, but is sufficient for localizing which region of interest an electrode may be recording from. Reconstructions in this dissertation were performed using volumetric imaging methodology from Bioimage Suite, primarily for clinical reasons.

3.4 Signal Processing

Common signal processing and filtering of intracranial EEG is detailed below. These methods are used throughout the studies detailed in this dissertation, but should not be considered as the standard methodologies for feature extraction in ECoG.

3.4.1 Preprocessing

After electrodes are implanted, there are typically a number of electrodes that contain significant noise or do not produce usable electrophysiology. There are a variety of physiological and engineering reasons why this can happen, including: placement of an electrode over a blood vessel; distance between the electrode and the cortex occupied by cerebral-spinal fluid, blood,

or air; mechanical failure of the wire connecting the electrode to the amplifier; placement over ictal cortex with repeated spiking behavior; and many other less common occurrences. The recorded signals are typically inspected by a neurologist and channels containing artifact or noisy signals are identified as "bad" channels, and excluded from further analysis.

3.4.1.1 Common Average Reference

Voltage potentials are recorded as a differential measurement between two electrodes. Traditionally a DC recording measures the potential difference between the electrode of interest and a ground electrode which is defined to be 0 volts. AC coupled recordings tend to make measurements relative to a reference electrode. In ECoG recordings, this reference electrode can be chosen as any other ECoG electrode or an external reference. In many of the recordings used in this dissertation, a skull-facing strip of four electrodes was implanted, and one of the electrodes on that strip was used as a reference. Another common choice for a reference electrode is a "silent, distant" electrode, typically located on a strip in non-primary cortex. Any artifact or neurophysiological change that causes a disturbance at the reference electrode will be observed in all channels that are referenced to it. A "common average reference" or CAR spatial filter is typically applied to remove common-mode noise across a collection of channels. This simple spatial filter is simply a subtraction of the mean signal, or the "common signal" from a group of channels. Typically, channels are split into a number of subgroups or "CAR blocks" relating to the underlying reference configuration or signal quality, and bad channels are excluded from these blocks. Design of a CAR filter is not always

straightforward and draws upon experience and a good understanding of electrophysiology.

3.4.2 Feature Extraction

Although evoked response potentials and other time-domain features can be extracted from ECoG signals, spectral features tend to be used frequently. Feature extraction in ECoG recordings typically involves short-time extraction of 70+ Hz bandpower. Modulation of this feature during behavioral periods is typically z-scored with respect to a baseline period. The most popular methods for extraction of spectral content have different parameters, even though they are mathematically equivalent (Bruns, 2004).

3.4.2.1 Discrete Fourier Transform

The discrete fourier transform is a simple and effective method for decomposing a "stationary" signal into its spectrum. The fast fourier transform is an efficient implementation of the discrete fourier transform. While the nature of electrophysiological recordings is anything but stationary, the method can be applied over short windows wherein stationarity is assumed. Shifting this window in time and recalculating a new spectrum results in a "spectrogram" which is a measure of the frequency content in the signal over time. The length of the window input to the FFT is directly related to the frequency resolution and the temporal shift of said window controls the temporal resolution.

3.4.2.2 Hilbert Spectral Analysis

Another way to determine the amplitude of high frequency power is to estimate a projection of the real-valued signal into its complex-valued analytic representation (which cannot be observed) and then calculate the instantaneous magnitude of that signal. The hilbert transform is *a* way to create an analytic signal that can be decomposed into instantaneous magnitude and phase for the purpose of characterizing band power in the native time-base of the input signal. This is accomplished in practice by applying the fast fourier transform to a filtered time-series with limited bandwidth, setting all negative frequency coefficients to zero, then performing an inverse fast-fourier transform. This method is used frequently to extract high gamma activity from ECoG recordings because it generates neural features with the native time-base of the input ECoG data, with associated phase values from which an instantaneous frequency can be calculated, with seemingly no parameters to tune.

Methodologically, there are a few considerations when applying the hilbert transform to ECoG data. Firstly, the hilbert transform is typically implemented with an FFT over the entire time-series, typically spanning several high frequency events, which violates the stationarity assumptions FFT makes about its input. Second, the there are infinite analytic representations of any particular real-valued signal. The hilbert transform generates a non-unique analytic signal which may not accurately portray the underlying phenomena that generated the observed signal. Finally, and most importantly, the commonly held belief that this method has no parameters is strictly untrue, as

the temporal evolution of the resulting amplitude is directly related to the bandwidth of the input; the higher the bandwidth of the input signal, the faster the amplitude modulates. This result is particularly problematic for phase-amplitude-coupling research that uses this method for extracting low frequency phase and high frequency amplitude.

3.4.3 Segmentation and Baseline

Synchronized behavioral data are recorded with the electrophysiology and are typically used to align behavioral events with neural features. "Baseline" modulation distributions can be calculated from periods where the subject is awake and not moving, speaking, or thinking about anything. Spectral log-power features are typically z-scored to this baseline distribution per-frequency and electrode. This log-transform changes the distribution of power values into a more gaussian distribution which better matches the assumptions of statistical tests. Frequency features within the high gamma band are then averaged together to produce one time-varying neural activation vector for each channel across time. Microphone input, typically sampled synchronously with the electrophysiology, can be thresholded for voice-onset timings. BCI2000 injects digital time-markers into the recorded files for when stimuli appear on-screen. This information can be used to segment trial and response periods for the creation of spatiotemporal maps; a process which is described in detail in Chapter 4.

References

Bédard, C., H. Kröger, and A. Destexhe (2006). "Does the 1/f Frequency Scaling of Brain Signals Reflect Self-Organized Critical States?" In: *Physical Review Letters* 97.11, p. 118102. DOI: 10.1103/PhysRevLett.97.118102.

Papademetris, Xenophon, Marcel P. Jackowski, Nallakkandi Rajeevan, Marcello DiStasio, Hirohito Okuda, R. Todd Constable, and Lawrence H. Staib (2006). "BioImage Suite: An integrated medical image analysis suite: An update". In: *The insight journal* 2006, p. 209. ISSN: 2327-770X.

Fischl, Bruce (2012). "FreeSurfer". In: *NeuroImage* 62.2, pp. 774–781. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2012.01.021.

Bruns, Andreas (2004). "Fourier-, Hilbert- and wavelet-based signal analysis: are they really different approaches?" In: *Journal of Neuroscience Methods* 137.2, pp. 321–332. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2004.03.002.

Chapter 4

Characterization of Spatiotemporal Relationships in Real-Time

This chapter was submitted as a technical communication to a special issue of Frontiers in Neuroscience on electrocorticographic brain-computer-interfaces, and as of the time of writing this dissertation, is still under review.

4.1 Introduction

A brain-computer interface (BCI) is a system that translates brain activity into control signals for a computer. Modern incarnations of BCIs rely on rapid and low-latency brain signal acquisition, preprocessing, feature extraction, classification and/or regression, and frequently, postprocessing of the resultant control signal. In the case of closed-loop BCI, some form of visual or auditory feedback is given to the user to inform them of their control performance, typically requiring a low round-trip latency from signal acquisition to output. BCI development typically requires performant implementations of data acquisition and signal processing algorithms, high precision synchronization of

external device telemetry, and typically, control of external software, requiring inter-process control or device input emulation.

These technical requirements make the development of software for this purpose extremely challenging; however, there are a number of existing software platforms that bootstrap this development endeavor. BCI2000 (Schalk et al., 2004) has been a standardized research platform for BCI development for the last 15 years; it has been used by over 400 labs, and has been cited in nearly 2,000 publications as of the time of writing this article. OpenViBE is another platform that has been developed to support real-time BCI research, offering a graphical programming language for signal processing and visualization (Renard et al., 2010). Additionally, a low-level communication protocol supporting signal acquisition and synchronization, called LabStreamingLayer, allows for TCP network streaming and synchronization of multi-modal data streams (Kothe, 2016) and could form the foundation of a BCI platform.

Simultaneous to the development of the aforementioned software, web-based applications have become increasingly prevalent, with browser software serving as a universal platform for execution. Web browsers—which have seen remarkable technological advancements in the last decade—are gaining adoption across all device classes because they enable interaction with content that has been authored using uniform standards for delivery and functionality. The demand for web apps that are capable of advanced rendering and computational feats is growing, and browsers are becoming increasingly powerful to meet this demand. Recent advancements in browser technology and standards have even opened up direct access to low-level system resources

such as graphics hardware and accelerometry/system sensors. Application programming interfaces (APIs) have exposed this hardware and software functionality via easy-to-use yet powerful and performant Javascript packages. Network-enabled services also implement publicly available APIs that allow developers to call upon remote computational resources, such as Amazon web services (AWS), or to query information from vast databases of indexed knowledge, such as Wikipedia and Google Image Search. Moreover, many libraries supporting visual presentation of user interfaces and data visualizations have been developed. For example, d3.js (Bostock, 2012) has been used to power interactive data visualizations with impressive performance and an expressive yet functional API.

Many of the technologies readily available in the modern web browser would be useful to have available for the development of a modern BCI—for example, the ability to tag data in real-time with a speech transcription, or the ability to present stimuli in 3D using a virtual reality headset. Existing BCI software suites generally provide some amount of interprocess communication, typically exposed via user datagram protocol (UDP) or shared memory. However, browsers do not typically allow web apps to access UDP natively due to security concerns; further, existing communication schemes like BCI2000's AppConnector interface do not scale well to high data volumes, like those required to transmit human electrocorticography (ECoG) signals. Modern browsers implement a protocol built on top of TCP called WebSocket (Fette, 2011) that allows an HTTP client to escalate an existing connection to a general purpose real-time bidirectional binary/ASCII communication interface.

WebSockets are perfectly situated to facilitate the transfer of raw brain signals, extracted neural features, and processed control signals from a BCI software suite to a web app on a browser-enabled device, as well as the transfer of auxiliary sensor information from the web app back to the native software suite, all in real time.

In this article, we present an implementation of the aforementioned interface on top of the BCI2000 ecosystem. We demonstrate the utility of this new interface with an example application that shares many technical requirements with a BCI: a cloud-based ECoG functional mapping tool capable of visualizing cortical activation in real-time at the bedside or in the operating room, and of storing the results from multiple centers in a centrally-hosted repository for review.

4.2 Design and Implementation

We chose to build our BCI WebSocket interface on top of BCI2000 as opposed to the other aforementioned technologies for many reasons, including support for acquisition devices in common use within epilepsy monitoring units and EEG research lab settings, high performance spectral extraction implementations, pedigree within the research community, highly accurate stimulus presentation capabilities, comprehensive documentation, and its ability to replay experimental sessions post hoc easily and accurately.

The BCI2000 environment is a general-purpose computational framework, typically used to construct BCIs, built upon four binary executables: the signal source module, which acquires electrophysiology from a supported amplifier;

the signal processing module, which extracts neural features and transforms those features into control signals; an application module, which reacts to those control signals and provides feedback to the subject; and an Operator module, which orchestrates the behavior of all three functional submodules of the system (see Figure 4.1). Signals propagate from the source module to the processing module to the application module, with interconnections facilitated by a network-based protocol (in older versions of BCI2000) or a shared memory interface (in more recent iterations). Each of the modules consists of a series of signal "filters", which accept an incoming signal (as a channels-by-elements array) and output a derived signal, potentially of different dimensionality. A built-in Operator scripting language allows for setup and configuration of filters within an experimental session to occur automatically, and a Telnet interface exists in the Operator module, capable of accepting textual commands in the Operator scripting language from outside BCI2000.

BCI2000Web is a module that accepts Operator scripting language commands via WebSocket and transmits them to the Operator executable via Telnet, returning system output back to the client. It is primarily used to control data acquisition and signal processing parameters remotely via a connected WebSocket-enabled client, typically a browser. BCI2000Web has been developed as a service that runs within the Node.js runtime. Upon starting, it opens a Telnet connection to the Operator module and also functions as a basic HTTP server. While BCI2000's Telnet implementation only supports one client sending one set of instructions that are executed serially, BCI2000Web

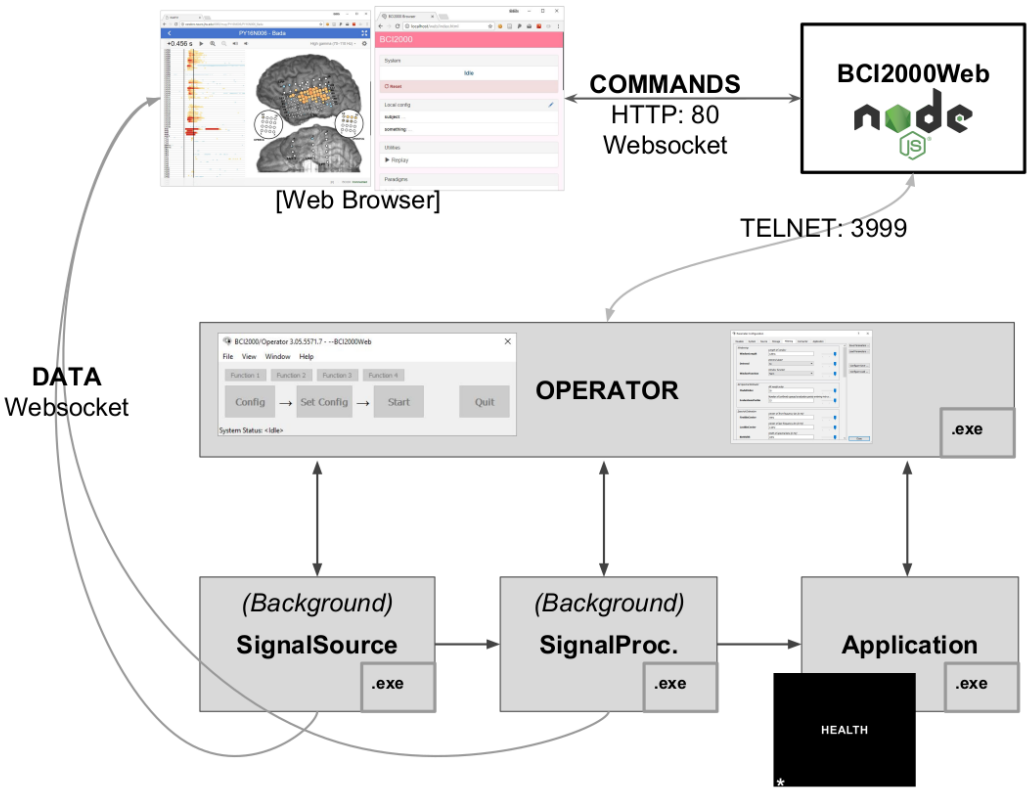

Figure 4.1: BCI2000Web System Diagram. A full BCI2000 stack including a Signal Source, Signal Processing, Application, and Operator module communicates with BCI2000Web, implemented as a Node.js module, via Telnet. Browser-based remote control software and visualization tools interact with BCI2000Web, and receive raw and processed neural signals directly from the BCI2000 system modules, via WebSockets, while the application module presents stimuli to the patient, in this case, the word stimulus "HEALTH" for a word reading paradigm.

provides an interface that allows multiple clients to send commands to the Operator module; these commands are queued and executed sequentially, with responses sent back to the appropriate client asynchronously. BCI2000Web is capable of interfacing with an unmodified BCI2000 distribution and automates system configuration without any further software or modifications to BCI2000 modules.

The raw and processed signal is never sent directly to the Operator module, so the signal can only be transmitted to a browser by compiling secondary WebSocket servers into the existing modules at desired locations within the filter chain. This modification has been realized in our implementation as a generic "WSIOFilter" (**W**eb**S**ocket **I**nput/**O**utput Generic**Filter**) that can be instantiated multiple times into the BCI2000 filter chain. Each WSIOFilter defines a parameter specifying the address and port its WebSocket server is hosted on. Once an incoming connection is escalated to a WebSocket, this filter sends packets to the client in the BCI2000 binary format, first describing the dimensionality of the signal and the system state vector via a "`SignalProperties`" and "`StateList`" packet, then a "`GenericSignal`" and "`StateVector`" packet for the current system signal and state vector once per sample block. These filters can be instantiated several times in the signal processing chain for any particular signal processing module. This filter has also been included as a source module extension that enables transmission of the raw signal in all signal source modules, and an application module extension that enables transmission of the application module input—identical to the signal processing output—in all application modules.

A WebSocket-enabled client is unlikely to natively understand the format of the incoming/outgoing messages on any of the aforementioned connections: our implementation of BCI2000Web adds some decorators to Operator scripting commands and Operator outputs to handle multiple clients, and the WSIOFilter output is implemented in the BCI2000 binary protocol. A JavaScript library, bci2k.js—available as a package on the Node package manager (NPM) registry—contains functions that manage the BCI2000 WebSocket connections and translate the binary BCI2000 format into readily usable data structures within a JavaScript context. Non-browser WebSocket-enabled clients will need to implement this functionality in order to communicate using these interfaces.

4.3 Application: ECoG Functional Mapping with WebFM

About a third of patients with epilepsy have seizures that are resistant to medication therapy. In many of these patients, seizures arise from a focal brain area, and if this area can be safely removed, seizure control can be achieved. When non-invasive testing cannot reliably identify the seizure onset zone as distinct from brain regions needed for normal neurological function, clinicians may choose to surgically implant electrodes in the depths of the brain (stereo-EEG) or on its surface (electrocorticography, or ECoG). These intracranial electrodes may be implanted for a week or more in order to reliably localize the onset of seizures. These electrodes also facilitate the identification of eloquent cortex—i.e., regions that are implicated in speech

and language, as well as perception, movement, and other important brain functions. A technique called electrocortical stimulation mapping (ESM) is typically used to map these regions. During ESM, pulse-trains of electrical current are passed between pairs of the implanted electrodes to temporarily disable a small patch of cortex while the patient performs a simple language or motor task. A behavioral change elicited by this temporary lesion indicates that the stimulated area of the brain is necessary for task completion (Ojemann et al., 1989). This testing procedure is time-consuming and uncomfortable for the patient, sometimes eliciting after-discharges (Lesser et al., 1984); these after-discharges can also evolve into seizures, which can be of questionable utility for diagnosing ictal cortex (Hamberger, 2007).

The limitations of ESM have motivated a complementary mapping technique based upon estimates of task-related changes in the power spectra, especially in high frequencies, of passive recordings of ECoG or stereo-EEG during behavioral tasks. This mapping technique, hereafter referred to as ECoG functional mapping, produces maps of task-related cortical activation, which may include cortex that is recruited by a task but not critical to task performance. In contrast, ESM uses a temporary electrophysiological disruption of cortical function to simulate the acute behavioral effects of tissue resection, and is presumed to be specific to areas critical to task performance. Nevertheless, a number of clinical studies have demonstrated good correspondence between ECoG functional mapping and ESM (Brunner et al., 2009; Wang et al., 2016). Moreover, several studies have shown that ECoG functional mapping can be used to predict post-resection neurological impairments, and in some

cases it has predicted impairments that were not predicted by ESM (Wang et al., 2016). For these reasons, some epilepsy surgery centers have begun to use ECoG functional mapping as a complement to ESM, sometimes providing a preliminary map of cortical function that guides the use of ESM. However, most epilepsy centers have not yet adopted ECoG functional mapping because of the lack of technical resources, especially software that can be used with their clinical EEG monitoring systems.

Several ECoG functional mapping packages have been developed in recent years. For example, SIGFRIED acquires a large baseline distribution of neural activity in a calibration block, then rapidly accumulates estimates of cortical activation by averaging neural activity evoked by behavior in blocks of time (Brunner et al., 2009). A commercial product called cortiQ (Prueckl et al., 2013) is capable of performing this block-based mapping paradigm, which makes it possible for minimally trained clinical professionals to perform passive ECoG mapping. Both SIGFRIED and cortiQ are built using the BCI2000 framework and take advantage of the extensive optimizations and development legacy of the platform. A more nuanced mapping technique, termed spatial-temporal functional mapping (STFM), provides time-resolved, trial-locked results during a specific task by collecting a pooled baseline activity from a pre-defined $\sim$1 second period before the onset of a trial, then performs a statistical test on each time/channel bin in a window of interest relative to trial onset (Wang et al., 2016). Though the results of STFM are more complicated and require more expertise to interpret than the block-based mapping used by SIGFRIED or cortIQ, they provide a more detailed map

of the spatial-temporal evolution of task-related activation, which can help clarify the role of different areas activated by a given task, of clear utility in cognitive neuroscience research and of potential clinical utility in planning surgical resections.

ECoG functional mapping relies on high performance signal processing and sophisticated real-time visualization, making it a suitable application example for BCI2000 and BCI2000Web. We saw an opportunity to build an easy-to-deploy-and-use tool for both researchers and clinicians that delivers the time-resolved, trial-locked results of STFM at the bedside in a web application, using BCI2000Web as the underlying communication technology to drive a browser-based interactive visualization. Below we present WebFM, a software suite built on top of Node.js and BCI2000Web for performing real-time functional mapping in a web browser.

4.3.1 Methods

An electrophysiological amplifier is typically used to collect data from implanted electrodes. In ECoG recordings, spectral power in the frequency band between 70 and 110 Hz (high gamma) is highly correlated with firing rates in the population of neurons directly under the recording electrode (Ray et al., 2008); hence, an increase in high gamma power is typically interpreted as an increase in neural activation in that area. The electrophysiological system theoretically needs to be sampling at a minimum of of 220 Hz to capture this activity, but in practice, most acquisition systems apply anti-aliasing low pass filters with cutoff frequencies at a quarter of the sampling rate. Because of this,

ECoG is typically sampled at 1000-2000 Hz across most systems, to ensure that the full high gamma band is captured after hardware filtering.

The signal processing module used in the system in the Johns Hopkins Epilepsy Monitoring Unit is a modification of the BCI2000 spectral signal processing module. This signal processing module consists of a chain of filters, the first of which is a spatial filter capable of applying a common average reference, a frequently used spatial filter for ECoG recordings (Liu et al., 2015). This is followed by a series of IIR Butterworth filters, including a fourth order low pass at 110 Hz, followed by a second order high pass at 70 Hz and a 4th order notch filter at 60 Hz. After the signal is downsampled to 500 Hz from the native sampling rate, it is passed through a spectral estimator filter, which generates an autoregressive model on a window of filtered data; the coefficients of this model are used to form an estimate of the signal's power spectrum. A WSIOFilter is instantiated at this point in the filter chain, capable of streaming this estimated spectral content of the neural signals in real-time. A browser is used to communicate to the bedside data-collection and stimulus-presentation machine, and to set up the system parameterization. A monitor and speaker connected to the bedside computer is set up in front of the patient, and a microphone is connected to the auxiliary analog inputs provided by the acquisition system, to be digitized synchronously with the electrophysiology. A system diagram and description of the full system topology is detailed in Figure 4.1.

A language or motor task is parameterized as a BCI2000 `.prm` file and a collection of audio-visual stimuli in a git repository hosted on GitHub.

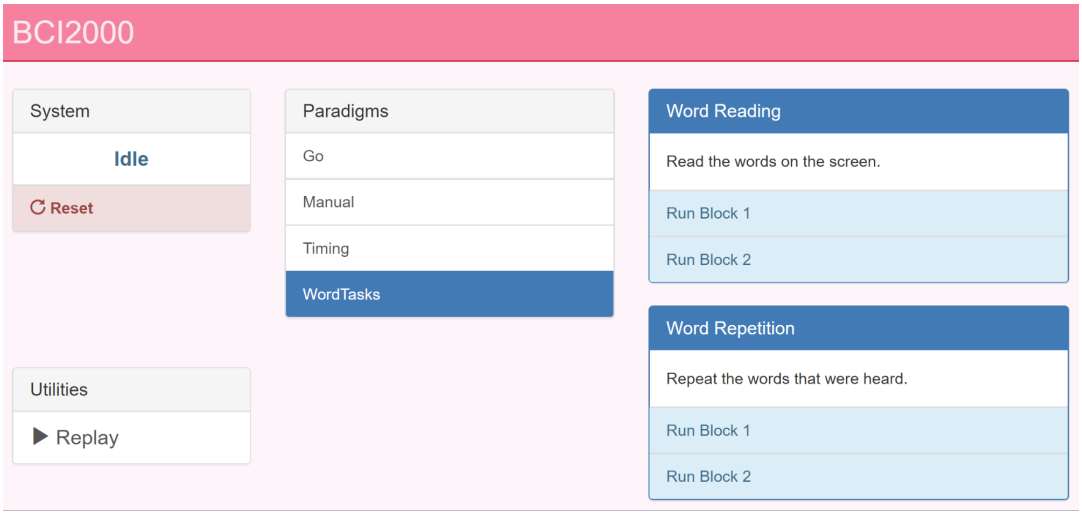

Figure 4.2: The BCI2000 Remote Control Interface. A screenshot of the BCI2000 remote control interface. The paradigm index is hosted by BCI2000Web over HTTP. This page is populated by the experimental paradigms present on the host machine (center) with buttons to start sub-tasks and specific blocks (right). A pane in the top left reads out the current BCI2000 system state, in addition to a system reset button. In the bottom left, a link to the system replay menu allows for recorded BCI2000 `.dat` file playback for system testing and offline mapping.

Any number of these tasks can be checked out into the BCI2000Web distribution, and the server will automatically present them as startup options within the built-in BCI2000Web browser interface, shown and described in Figure 4.2. These paradigms typically specify a parameterization for `StimulusPresentation.exe`, a BCI2000 application module capable of presenting audio-visual stimuli to the patient with high-precision timing and sequence control.

The BCI2000Web system currently supports more than 20 possible experimental paradigms, including a task battery used for clinical assessment of functional localization.

4.3.1.1 Patients and Electrode Localization

Before mapping, a post-operative computed tomography scan containing electrode locations is co-registered to a pre-operative magnetic resonance imaging scan of sufficient resolution (typically with voxel dimensions of 1 mm or less) to render the patient's cortical surface anatomy in high detail, using Freesurfer (Fischl, 2012) or Bioimage Suite (Papademetris et al., 2006). These electrode locations are overlaid on a 2D rendering of the cortical surface. An image file depicting this cortical anatomy and electrode layout, as well as a comma-separated value (`.csv`) file containing the normalized image coordinates of each electrode, is uploaded to the WebFM server via controls within the WebFM browser interface. This layout doesn't typically change during a patient's EMU stay, and it is referenced and retrieved through APIs in the WebFM sfotware using a subject identification code, effectively de-identifying the reconstruction for research purposes.

4.3.1.2 Software

During an ECoG functional mapping session, a browser running on the visualization device contacts the WebFM server and queries the bedside machine for the subject's identification code and what task is currently running. The WebFM server then serves the corresponding cortical reconstruction image and sensor location file in addition to a bolus of JavaScript code that is capable of opening WebSockets to the BCI2000Web server and WSIOFilters running on the bedside machine. The browser then opens these data streaming WebSockets and performs the mapping without further contacting the WebFM

server. After each trial of the task, the visualization is updated and once a full task run has been collected, the resulting map can be saved back to the WebFM server for indexing and post-hoc inspection; these maps are made available on the WebFM Landing page, as shown in Figure 4.3.

The statistics and visualization for WebFM are based on the techniques and methods described in Wang et. al (Wang et al., 2016). The baseline window for the tasks is defined as a configurable period from 1000 to 200 ms before the trial onset and a baseline distribution is formed per channel from the pooled high gamma power values during this period. A two-way t-test is performed between the distribution for each time-channel bin and that channel's baseline distribution. The resulting p-values are corrected for multiple comparisons using the Benjamini-Hochberg (BH) procedure, controlling the false discovery rate at 0.05 (Benjamini and Hochberg, 1995). This correction is used to threshold the results displayed in the WebFM raster and spatial plots: time-channel bins that did not survive the BH correction are hidden from view. Any individual time point in this raster can be dynamically selected and visualized by "scrubbing" the mouse cursor over the raster display; this yields circles drawn on a two dimensional representation of the electrode montage, highlighting which cortical locations were active during that particular time-point across trials. An options dialog allows users to change baseline periods, modify visualization timing parameters and amplitudes, as well as make comparisons across task conditions and contrasts. The visualization is available as a live demo at `www.webfm.io` and is shown and further described in Figure 4.4.

The visualization APIs exposed by WebFM can be used to implement a

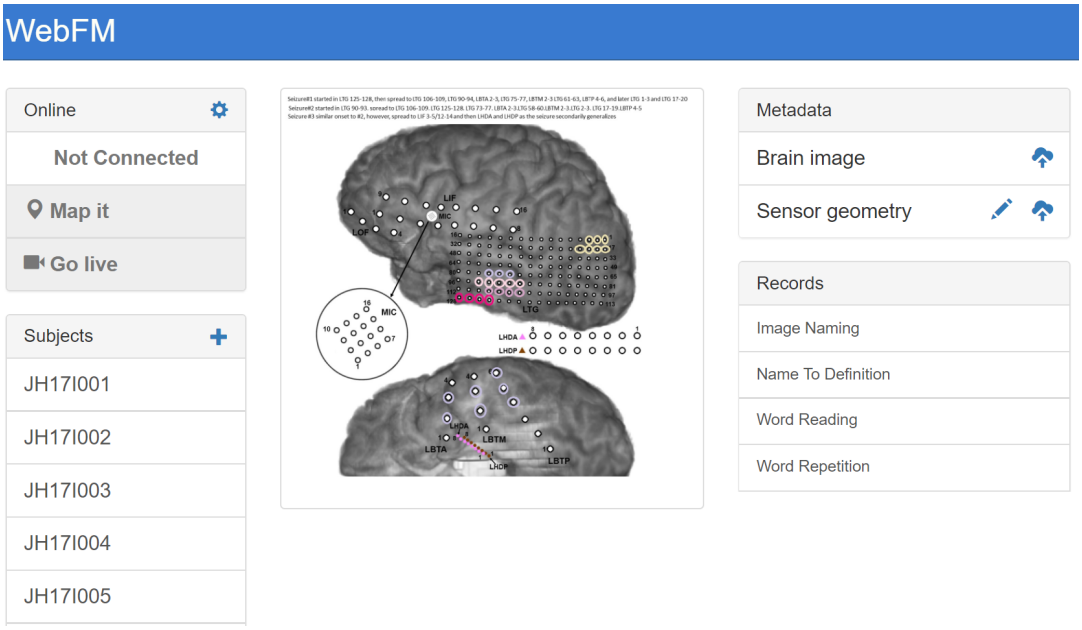

Figure 4.3: The landing page for WebFM. A pane in the top left shows system state and houses buttons that start trial-based functional mapping paradigms and a "live" mode that visualizes neural activity on the brain in real-time, as visualized in prior studies (Lachaux et al., 2007). A list of subject identifiers on the bottom left pane enables users to pull up previous/current subjects; a list of saved maps for the selected subject appears in the "Records" pane on the bottom right. The "+" in the top left of the "Subjects" pane allows operators to add new subjects to the database, and the "Metadata" pane at the top right allows operators to upload brain reconstruction images and normalized electrode locations for displaying functional mapping results. The brain images used for mapping are often overlaid with information about seizures and/or ESM results, so that functional activation can be easily visually compared with these data; the image shown in the center includes colored circles depicting the hypothesized spread of ictal activity during the subject's seizures.

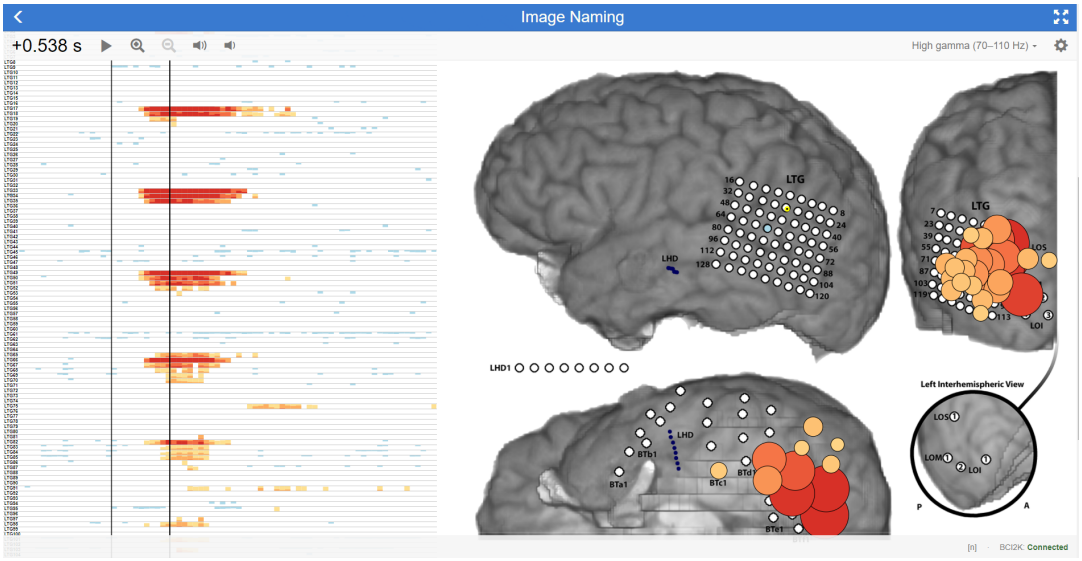

Figure 4.4: WebFM visualization description. An example of WebFM results for an image naming task in a subject with high density (5-mm spacing) temporal-parietal-occipital electrode coverage. A horizon raster Heer, Kong, and Agrawala, 2009 to the left shows a time (x-axis) by channels (y-axis) plot of trial-averaged task-modulated high gamma power, thresholded for statistical significance with BH correction for a FDR of less than 0.05. Warm colors represent a statistically significant increase in task-modulated high gamma power, while cool colors indicate a statistically significant decrease in task-modulated high gamma power. The left black vertical bar within the raster indicates the trial-start (t = 0s) where StimulusCode transitioned from zero to a non-zero value, indicating that a stimulus was being displayed.

number of other visualizations as well. One mode of WebFM provides a visualization of raw high gamma activation in real time, as in Lachaux et. al (Lachaux et al., 2007); other modifications have also been used to visualize the propagation of interictal spiking and seizure propagation across cortex.

4.3.2 Deployment and Results

As of the time of writing, the WebFM system has been deployed at two sites: the Johns Hopkins Hospital and the University of Pittsburgh Medical Center. Across these sites, WebFM has been used with three acquisition devices: the NeuroPort system (Blackrock Microsystems, Salt Lake City, UT), a Grapevine system (Ripple, Salt Lake City, UT), and the EEG1200 system (Nihon Kohden, Tomioka, Japan). Between these sites and amplifiers, WebFM has been used to create over 200 functional maps across 33 subjects. The majority of these subjects (19) were hospital inpatients undergoing epilepsy monitoring prior to resective surgery. Clinical staff in the Johns Hopkins Epilepsy Monitoring Unit have a link to the WebFM portal on their desktop machines and frequently reference the passive ECoG mapping results when discussing surgical plans. The remaining 14 subjects were temporarily implanted with a 64-channel high density ECoG strip during lead implantation for deep brain stimulation; for these subjects, WebFM was used to map sensorimotor cortex in the operating room.

4.4 Discussion

BCI2000Web and WebFM take advantage of several recent technological developments. First and foremost, these packages capitalize on advancements in the modern web browser, which is quickly becoming a platform capable of general purpose computing. With a focus on frontend user interaction, many packages have been written in JavaScript that support the rapid implementation of interactive applications and visualizations. WebFM makes use of d3.js (Bostock, 2012) to provide a high-quality interactive visualization of trial-averaged high gamma modulation directly on the brain.

The key to taking advantage of these web-based technologies is the implementation of BCI2000Web, which utilizes the WebSocket API to transmit binary-formatted brain data directly to the browser over TCP/HTTP, allowing direct communication to and from BCI2000. While the experimental paradigms presented in conjunction with WebFM utilized the native BCI2000 stimulus presentation module to interact with the subject, the general-purpose access to Operator scripting over WebSockets provided by BCI2000Web easily lends itself to a different system architecture, in which a browser application itself is responsible for interacting with the subject and providing experimental markers sent via WebSocket; this topology is depicted in Figure 4.5. Several paradigm packages for BCI2000Web leveraging this architecture have been authored to date; one makes use of the WebSpeech API (Shires and Wennborg, 2012) to do real-time speech tagging and segmentation for tasks involving freely generated speech, and another uses the WebMIDI and WebAudio APIs (Wyse and Subramanian, 2013) to register subject input on

musical peripheral devices, and performs high-performance audio synthesis in response. Public JavaScript APIs allow for rich BCI interactions, and experimental paradigms can pull upon web resources such as Google Image search for providing varied and tailored stimuli at run-time. Extending this idea, it is easy to envision a system architecture in which users' neural data are sent to a browser application that communicates with a server backend in real-time, allowing cloud-based services to apply sophisticated machine learning techniques that wouldn't be computationally feasible on the client-side. Even further, one could develop a browser-based application that transmits multiple users' neural data to each other's clients, facilitating brain-based communication.

Cross-device compatibility is another advantage to using the browser as a visualization and stimulus presentation platform. Any browser-enabled device (smartphone, tablet, PC, or even game console) can be used to present stimuli or visualize output. Because of this "write-once, run-anywhere" development process, WebFM can be used by clinicians to view mapping results in real-time on their smartphones from outside the patient's room while ECoG functional mapping is running.

4.4.1 Drawbacks and Caveats

The rationale behind the division of processing using native binaries and visualization using browser-interpreted javascript is due to current limitations inherent to browsers. Browser-hosted JavaScript is rapidly advancing

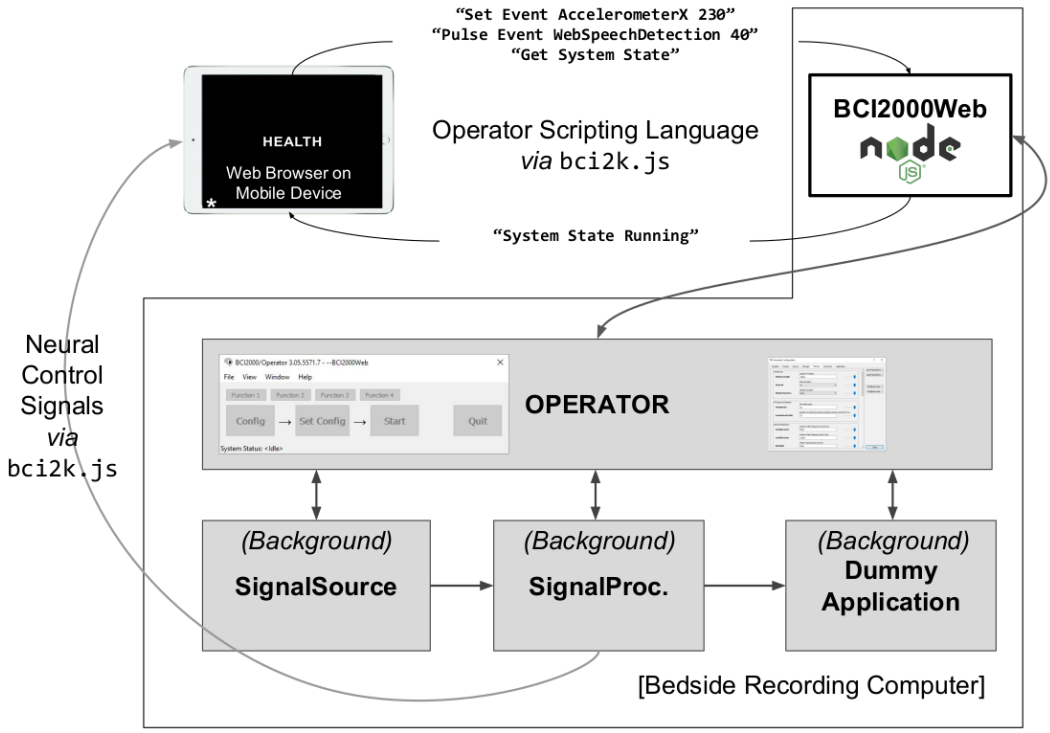

Figure 4.5: A BCI2000Web Browser-based Paradigm. A system diagram depicting an experiment implemented in browser JavaScript running on an independent mobile device. The mobile device is running an experimenter-implemented web-page in fullscreen mode which communicates directly with BCI2000Web for event logging as well as the Signal Processing module for receiving extracted neural control signals. A JavaScript package, bci2k.js, manages websocket connections that handle transmission of operator scripting language commands and decodes neural control signals from a binary format. The mobile device is running a word reading paradigm (with the stimulus "HEALTH" currently presented) that has defined asynchronous experimental states including markers for automated vocal transcription onsets using the WebSpeech API. A query for system state is also relayed by the BCI2000Web server. The benefit of such an architecture is that the patient interface is separated from the bedside clinical acquisition machine and can be left with the patient without concern of the patient manipulating the clinical datastream.

as a next-generation efficient computational platform with the advent of WebAssembly and ASM.js (Herman, Wagner, and Zakai, 2014), but at the time of writing it is still too computationally demanding to perform real-time feature extraction and signal processing in the browser. Furthermore, browser access to low level computer hardware and connected USB devices is only in the early development stages. Given these limitations, BCI2000Web was designed to take advantage of the device driver access and computational efficiency of the C++ code base that powers BCI2000 for acquisition device abstraction and signal processing/feature extraction. This architecture frees frontend developers from dealing with complicated signal processing code in JavaScript, and instead enables them to focus on user experience and design. In the future, a full-stack BCI2000 analog could be implemented directly within the browser, and BCI2000Web is a glimpse of what that software could empower for web developers with access to neural features.

A significant amount of the development effort for BCI2000 has been spent on implementing high-performance signal processing and stimulus presentation software. Delivering audio-visual stimuli to subjects with a consistent yet minimal latency is a nontrivial task that BCI2000 has accomplished by interfacing with low-level graphics drivers in a nuanced way. Operating system version, bit-width (32 vs 64), driver versions, compiler optimizations, and varying hardware capabilities collude to make this stimulus presentation problem a fragmented and moving target; one which BCI2000 has historically hit with surprising accuracy, achieving visual presentation latency on the order of one to two frames at a 60 Hz monitor refresh rate and audio latencies

on par with modern audio production software (Wilson et al., 2010). The BCI2000 core team encourage developers to implement custom signal processing and stimulus presentation paradigms within this BCI2000 environment using documented C++ code templates in order to benefit from these optimizations. That said, so long as tasks are designed properly and ground truth stimulus and response signals are collected (i.e. screen mounted photodiodes and patient facing microphones connected directly to auxiliary inputs on the amplifier), it is still possible to collect data of high scientific quality using the browser as the primary stimulus presentation software even if its stimulus display and communication latency are in question.

We benchmarked the visual timing performance of a system with and without BCI2000Web modifications using the procedure in (Wilson et al., 2010) on a platform comprising Windows 7 64 bit with BCI2000 r5688, Google Chrome 67.0.3396.99, and a 256 channel 1000 Hz recording from a Blackrock NeuroPort running with a 20 ms sample-block size. An unmodified BCI2000 distribution on this system exhibits a visual latency (t_{3v} as expressed in (Wilson et al., 2010)) of 52 ms with a standard deviation of 8.0 ms. With BCI2000Web sending neural signals to a browser via WebSocket on the same acquisition machine, a mean visual latency of 60 ms with a standard deviation 9.4 ms was observed. Using a hospital wireless network to send neural signals via WebSocket to a tablet PC running Windows 10 and the same version of Chrome results in a visual latency of 62 ms with a standard deviation of 13.4 ms. These latency metrics indicate a minimal impact to timing performance when using BCI2000Web. In many real-time BCI implementations, spectral feature

extraction occurs in windows of 128–256 ms with a slide of 16–32ms, and single-trial visual timing differences fall well within one windowing period. BCIs reliant upon time-domain features—in particular those that perform trial-averaging of evoked response potentials—will be more sensitive to these latency differences, and it is critically important to run timing benchmarks for specific hardware/software/network configurations in these circumstances.

4.5 Conclusions

The development of a communication protocol that connects one of the most widely adopted BCI research and development suites with the power of modern browser technologies is expected to accelerate the pace of development for BCI technologies. Newer software developers, primarily taught using these modern software development paradigms, can now develop new BCI applications and neural signal visualizations while leveraging the legacy and performance of native BCI2000 modules. We have developed and presented a web-based ECoG functional brain mapping tool using this technology, and we have successfully deployed it at two sites with a cohort of 33 patients over two years. BCI2000Web and WebFM together utilize the relative strengths of a highly optimized C++ code base in BCI2000 and the high level visualization libraries within modern browsers to demonstrate a clinically useful and modern functional mapping tool. We have also used BCI2000Web for ongoing, albeit unpublished, BCI research projects, and we describe herein the advantages and potential uses of BCI2000Web in future BCI applications. This software is documented and released under permissive free and open source

software licenses, and is put forward by the authors for use in the research and development of brain computer interfaces and multi-site studies on the clinical efficacy of ECoG functional mapping.

Data Availability Statement

A standalone distribution of BCI2000Web is available on GitHub (`github.com/cronelab/bci2000web`). The bci2k.js package can be installed with NPM (`node install bci2k`); its codebase is available on GitHub (`github.com/cronelab/bci2k.js`). WebFM can also be found on GitHub (`github.com/cronelab/webfm`).

The data used in the live demo at `www.webfm.io` is available via the WebFM API: the subject's brain image, base-64 encoded, is located at `www.webfm.io/api/brain/PY17N009`; the subject's sensor geometry is located, in JSON format, at `www.webfm.io/api/geometry/PY17N009`; the high gamma activation data for the presented task (syllable reading) is located at `www.webfm.io/api/data/PY17N009/SyllableReading`.

References

Schalk, G., D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw (2004). "BCI2000: a general-purpose brain-computer interface (BCI) system". In: *IEEE Transactions on Biomedical Engineering* 51.6, pp. 1034–1043. ISSN: 0018-9294. DOI: 10.1109/TBME.2004.827072.

Renard, Yann, Fabien Lotte, Guillaume Gibert, Marco Congedo, Emmanuel Maby, Vincent Delannoy, Olivier Bertrand, and Anatole Lécuyer (2010). "OpenViBE: An Open-Source Software Platform to Design, Test, and Use Brain–Computer Interfaces in Real and Virtual Environments". In: *Presence: Teleoperators and Virtual Environments* 19.1, pp. 35–53. ISSN: 1054-7460. DOI: 10.1162/pres.19.1.35. (Visited on 01/17/2018).

Kothe, C (2016). "Lab streaming layer". In:

Bostock, Michael (2012). "D3. js". In: *Data Driven Documents*.

Fette, Ian (2011). "The websocket protocol". In:

Ojemann, G., J. Ojemann, E. Lettich, and M. Berger (1989). "Cortical language localization in left, dominant hemisphere. An electrical stimulation mapping investigation in 117 patients". In: *J Neurosurg* 71.3, pp. 316–26.

Lesser, R. P., H. Lüders, G. Klem, D. S. Dinner, H. H. Morris, and J. Hahn (1984). "Cortical afterdischarge and functional response thresholds: results of extraoperative testing". In: *Epilepsia* 25.5, pp. 615–621. ISSN: 0013-9580.

Hamberger, M. J. (2007). "Cortical language mapping in epilepsy: a critical review". In: *Neuropsychology Review* 17.4, pp. 477–489. ISSN: 1040-7308. DOI: 10.1007/s11065-007-9046-6.

Brunner, P., A. L. Ritaccio, T. M. Lynch, J. F. Emrich, J. A. Wilson, J. C. Williams, E. J. Aarnoutse, N. F. Ramsey, E. C. Leuthardt, H. Bischoi, and G. Schalk (2009). "A practical procedure for real-time functional mapping of eloquent cortex using electrocorticographic signals in humans". In: *Epilepsy Behav*. ISSN: 1525-5069 (Electronic). DOI: 10.1016/j.yebeh.2009.04.001.

Wang, Yujing, Matthew S. Fifer, Adeen Flinker, Anna Korzeniewska, Mackenzie C. Cervenka, William S. Anderson, Dana F. Boatman-Reich, and Nathan

E. Crone (2016). “Spatial-temporal functional mapping of language at the bedside with electrocorticography”. In: *Neurology*. ISSN: 0028-3878, 1526-632X. (Visited on 03/03/2016).

Prueckl, R., C. Kapeller, C. Potes, M. Korostenskaja, G. Schalk, K. H. Lee, and C. Guger (2013). “cortiQ - Clinical software for electrocorticographic real-time functional mapping of the eloquent cortex”. In: *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 6365–6368. DOI: 10.1109/EMBC.2013.6611010.

Ray, S., S. S. Hsiao, N. E. Crone, P. J. Franaszczuk, and E. Niebur (2008). “Effect of stimulus intensity on the spike-local field potential relationship in the secondary somatosensory cortex”. In: *Journal of Neuroscience* 28.29, pp. 7334–43. DOI: 10.1523/JNEUROSCI.1588-08.2008.

Liu, Y., W. G. Coon, A. de Pesters, P. Brunner, and G. Schalk (2015). “The effects of spatial filtering and artifacts on electrocorticographic signals”. In: *Journal of Neural Engineering* 12.5, p. 056008. ISSN: 1741-2552. DOI: 10.1088/1741-2560/12/5/056008. (Visited on 01/17/2018).

Fischl, Bruce (2012). “FreeSurfer”. In: *Neuroimage* 62.2, pp. 774–781.

Papademetris, Xenophon, Marcel P Jackowski, Nallakkandi Rajeevan, Marcello DiStasio, Hirohito Okuda, R Todd Constable, and Lawrence H Staib (2006). “BioImage Suite: An integrated medical image analysis suite: An update”. In: *The insight journal*, p. 209.

Lachaux, J. P., K. Jerbi, O. Bertrand, L. Minotti, D. Hoffmann, B. Schoendorff, and P. Kahane (2007). “BrainTV: a novel approach for online mapping of human brain functions”. In: *Biol Res* 40.4, pp. 401–13. ISSN: 0716-9760 (Print) 0716-9760 (Linking).

Benjamini, Yoav and Yosef Hochberg (1995). “Controlling the false discovery rate: a practical and powerful approach to multiple testing”. In: *Journal of the royal statistical society. Series B (Methodological)*, pp. 289–300.

Heer, Jeffrey, Nicholas Kong, and Maneesh Agrawala (2009). “Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. New York, NY, USA: ACM, pp. 1303–1312. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518897. URL: http://doi.acm.org/10.1145/1518701.1518897 (visited on 01/17/2018).

Shires, Glen and Hans Wennborg (2012). “Web speech api specification”. In: *Final Report, W3C*.

Wyse, Lonce and Srikumar Subramanian (2013). "The Viability of the Web Browser as a Computer Music Platform". In: *Computer Music Journal* 37.4, pp. 10–23. ISSN: 0148-9267. DOI: 10.1162/COMJ_a_00213. (Visited on 01/17/2018).

Herman, Alon Zakai David, Luke Wagner, and A Zakai (2014). *asm. js—Working Draft—18 August 2014*.

Wilson, J. A., J. Mellinger, G. Schalk, and J. Williams (2010). "A Procedure for Measuring Latencies in Brain-Computer Interfaces". In: *IEEE Transactions on Biomedical Engineering* 57.7, pp. 1785–1797. ISSN: 0018-9294. DOI: 10.1109/TBME.2010.2047259.

Chapter 5

Mapping of Visual Semantic Attributes to Spatiotemporal Features of Neural Recordings

This chapter was published as an article in NeuroImage (Rupp et al., 2017)

5.1 Background and Motivation

The view that objects are encoded according to their semantic attributes or features, while not new, has become quite practical. Under an attribute-based view, a concept can be encoded over a large set of meaningful attributes, with each attribute assigned a value or set of values related to its probability, weight, or importance (Rosch, 1978). For example, the encoding of the concept "bird" assigns high probabilities to attributes typical of birds (has beak, flies, etc.) and low or zero probabilities to attributes atypical of birds (has four legs, manmade, etc). Substantial work has been done to catalogue the attributes and weights associated with different concepts, and attribute ratings can account for a host of human judgments about the relationships between concepts

and the organization of categories (Binder et al., 2016; Cree and McRae, 2003; Garrard et al., 2001; Ruts et al., 2004). In related work on vector space models of semantics, automated methods can be used in place of human annotators to learn latent semantic features from the statistical properties of words and phrases in large text corpora (Deerwester et al., 1990; Mikolov et al., 2013; Pennington, Socher, and Manning, 2014), and these latent features are similarly useful in accounting for human judgments (Pereira et al., 2016).

Efforts to decompose concepts into their constituent attributes or features have been used to great effect in the study of knowledge representation in the human brain. Following methods pioneered by Mitchell et al., 2008 to learn relationships between individual semantic features and the neural activity patterns they evoke, subjects perform tasks that require semantic processing – viewing or naming objects (Clarke et al., 2015), reading words or sentences (Wehbe et al., 2014), considering semantic attributes (Sudre et al., 2012), generating category exemplars (Simanova et al., 2014b), watching movies (Huth et al., 2012), or listening to stories (Huth et al., 2016) – while neural responses are recorded with functional magnetic resonance imaging (fMRI) or magnetoencephalography (MEG). Because stimuli can be represented in terms of their constituent semantic attributes or features, a mapping can be learned between each semantic feature and its associated neural responses (i.e. voxel intensities, MEG sensor amplitudes), typically through linear regression. These encoding models project semantic features into a neural feature space, and similarly, decoding models can be used to project recorded neural activity patterns into a semantic feature space.

The resulting neurosemantic models have provided new insights into conceptual knowledge representation in the mind and brain. The fact that neurosemantic models can be used to successfully learn mappings between semantic and neural features suggests that the brain's representation of objects involves decomposition into semantic features. This paradigm also provides a framework for testing theories about what specific semantic features are represented in the human brain (Just et al., 2010), how they are encoded in neural activity (Huth et al., 2016), and how cognitive processes modulate neurosemantic representations (Çukur et al., 2013). Likewise, from a decoding perspective, decompositional neurosemantic models are very powerful in that they can interpret neural activity from concept classes they have not been trained on in a process termed zero-shot learning (Palatucci et al., 2009).

The impact of this approach, though, is limited by the quality and quantity of available neural data. Non-invasive neuroimaging methods are subject to lower signal-to-noise ratios, trade-offs between temporal and spatial resolution, and indirect estimates of neural activity. Invasive alternatives like electrocorticography (ECoG) can only be used in the relatively rare clinical setting when implanting electrodes on the surface of the cortex is a clinical necessity. As a result, spatial coverage is determined solely by clinical considerations, which leads to varied anatomical sampling across patients. At the same time, ECoG offers high temporal resolution, a high signal-to-noise ratio due to direct contact between electrodes and the cortical surface, and more direct observations of neural processing. Evidence of this can be found in studies showing that ECoG responses correlate well with spiking activity (Manning et

al., 2009; Ray et al., 2008), and hemodynamic responses (Logothetis et al., 2001; Niessing et al., 2005), with activity in high-gamma frequencies (e.g. 70–110 Hz) serving as a particularly good index of underlying neural processing.

Despite the potential advantages, there have been relatively few studies of semantic attribute representation using ECoG. The few attempts to use ECoG for semantic decoding have relied on discriminative approaches over a small number of trained classes or categories (Liu et al., 2009; Wang et al., 2011). In one of the only published examples of semantic decoding from ECoG, Wang et al., 2011 asked patients to perform several different tasks that activated representations of semantic properties (e.g. visual object naming), and then trained Support Vector Machine (SVM) and Gaussian Naïve Bayes (GNB) classifiers to decode the evoked responses to one of the three possible target categories (i.e. foods, tools, and body parts). Performance varied across subjects, tasks, and classifier types, with mean classification rates of approximately 56% correct and a range from approximately 40% to 74% (where 33% is chance), indicating that substantial category information can be extracted from ECoG. While encouraging, conclusions drawn from a very restricted number of classes (e.g. foods, tools, and body parts) or dimensions of variation (e.g. living vs. non-living, large vs. small) may be partially confounded by expectation and perceptual set effects that cause subjects to artificially attend to and process these dimensions.

Chen et al., 2016 recently extended the ECoG-based study of semantic representations to 100 objects across a range of semantic categories and attributes. They adapted the searchlight-based Representational Similarity

Analysis (RSA) typically used in fMRI to assess where and when semantic information was encoded in ECoG responses during a picture naming task. The technique assumes that semantic information is represented as complex spatiotemporal patterns detectable by ECoG, and looks for spatiotemporal structure in the neural responses that correlate with the structure inherent in semantic representations. Using RSA, the authors found evidence of semantic encoding in the ventral pathway from basal occipital-temporal to anterior temporal lobe regions, but these results were primarily accounted for by a simple binary semantic model that coded items as either living or non-living (stimuli were evenly split between these two categories).

ECoG research in other language domains like speech perception (Pasley et al., 2012) and speech production (Mugler et al., 2014) have used generative encoding and decoding approaches to study language processing as it naturally varies across a range of stimuli and dimensions, but these methods have yet to be applied to semantics. In the current report, we adapted the decompositional semantic encoding approach previously used with fMRI and MEG data for ECoG (as summarized in Figure 5.1) to assess the degree to which semantic attributes are encoded in the ECoG signal. To do this, we recorded ECoG while patients named objects from 12 different semantic categories. Using these responses, high-dimensional semantic attribute encoding models were trained to decode objects unseen during model training. The trained models were then analysed in terms of which semantic dimensions were reliably encoded across different electrodes, time points, and frequency bands.

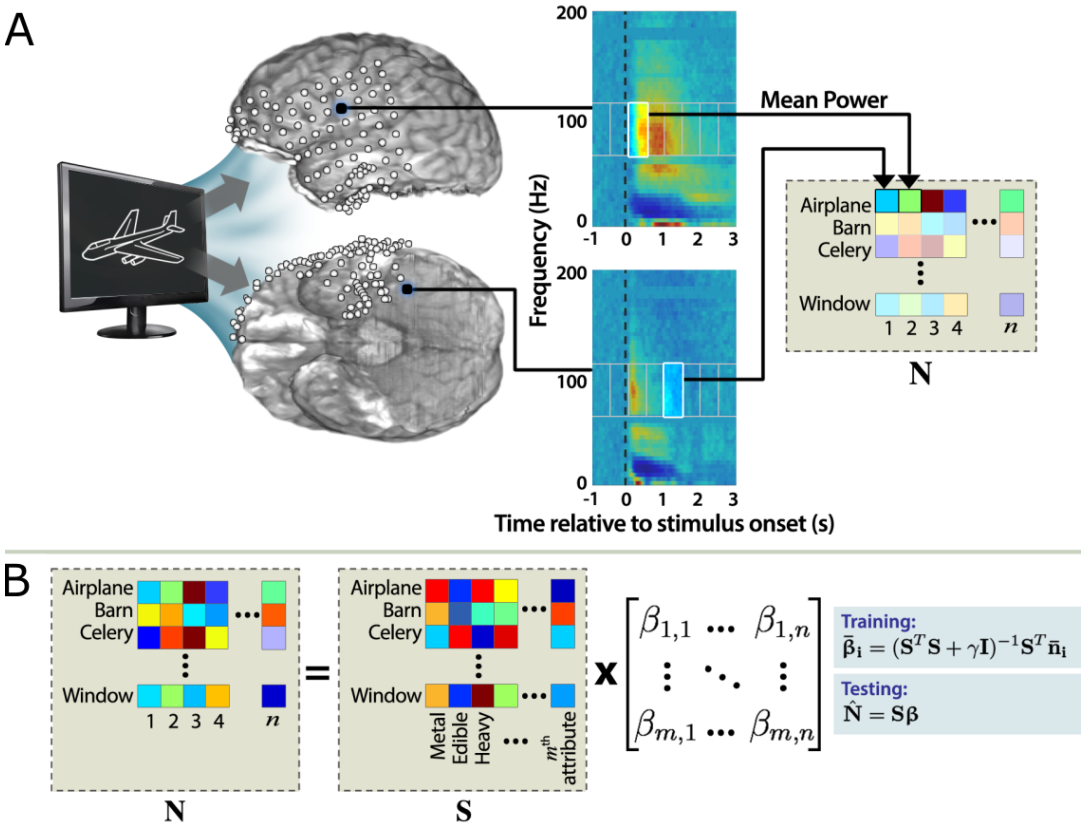

Figure 5.1: Training and testing encoding models from ECoG. (A) Patients named objects and spectral estimation was performed on their neural signals to produce mean power over a variety of frequency bands and temporal windows (only high-gamma shown here). A subset of neural features (particular frequency bands at particular time windows at particular electrodes) was selected for use in the encoding model. (B) Linear ridge regression was used to learn a neural encoding model β, which maps from semantic attribute ratings $\mathbf{S}$ to neural feature values $\mathbf{N}$. To decode a new neural activity pattern $\bar{\mathbf{n}}$ generated by an untrained object, $\bar{\mathbf{n}}$ is compared via cosine distance to a set of predicted neural activity patterns generated by applying β to a catalogue of possible objects and their semantic attributes.

5.2 Materials and Methods

5.2.1 Data

5.2.1.1 Subjects

Electrocorticography was recorded from 9 patients with intractable epilepsy (2 female, 31–44 years old) during in-patient monitoring for pre-surgical localization of their ictal onset zone and eloquent cortex. All patients provided informed consent according to a protocol approved by the Johns Hopkins Medicine Institutional Review Boards.

5.2.1.2 Paradigm

Patients performed a standard visual object naming task with the same 60 line drawings used by Mitchell et al., 2008. Briefly, white line drawings were presented on a black background, with a centered white fixation cross present during inter-stimulus intervals. Drawings were shown for 1 s, with an inter-stimulus interval varying randomly between 3.5 and 4.5 s. Patients were instructed to name the pictured object as soon as it came to mind, or to say "pass" when they could not immediately answer. Four patients were familiarized with the stimuli beforehand using one of two procedures: two patients were simply shown the stimuli with labels and instructed to learn them, and two patients were asked to provide a verbal description for each object. The remaining five patients were not exposed to the stimuli prior to the naming task.

The 60-item stimulus set consisted of 5 different objects from each of 12

different categories (animals, body parts, buildings, building parts, clothing, furniture, insects, kitchen utensils, man-made objects, tools, vegetables, and vehicles. For each patient, six blocks of data were collected. All 60 objects were shown in each block in a pseudorandom order. Both the behavioral paradigm and the ECoG data recording were implemented with BCI2000 (Schalk et al., 2004). Verbal responses and stimulus onset were both recorded through the analog input bank using a microphone and photodiode, respectively. Behavioral performance was overall very good for all patients (See Table 5.1). Occasional naming errors were not excluded from ECoG analysis.

5.2.1.3 ECoG Recordings

Data analyzed from patients P2 through P9 were collected with standard ECoG grids and strips, each of which contained electrodes with 4 mm diameter and 10 mm spacing. For one patient, P1, a subset of analyzed electrodes was part of a high-density grid with 2 mm diameter and 5 mm spacing. Additional depth and micro-ECoG electrode arrays were implanted in a subset of patients but were not analyzed. Electrode placements were determined by clinical criteria and varied widely across patients (See Table 5.1).

5.2.1.4 Signal Processing

ECoG signals were sampled at 1000 Hz, digitized, and recorded using the BlackRock Neuroport system. Recordings were made with a referential montage in which the reference was a contact on a 4-electrode strip that had been implanted for this purpose facing the dura mater, or in which the reference was a cortical contact chosen because of its greater distance from most other

	Sex	Age	LD	Electrode placement	(#)	Naming
P1	F	37	L	Left temporal HD grid, left fronto-parietal grid, basal strips	153	89%
P2	M	39	L	Left fronto-temporal grid, superior frontal grid, inferior frontal strip, basal strips	87	96%
P3	M	31	L	Left temporal grid, basal grid	36	89%
P4	M	37	L	Right fronto-temporal grid, basal strips	72	96%
P5	M	44	L	Bilateral strips	86	74%
P6	M	37	L	Right fronto-temporal grid, basal strips, frontal strops, occipito-parietal strip	82	97%
P7	M	37	L	Right parietal grid, frontal strip, posterior basal strips	80	93%
P8	M	33	R	Left occipital grid, temporal grid and strips, basal strips	95	89%
P9	F	35	L	Bilateral strips	106	98%

Table 5.1: Patient Demographics. Summary of patient demographics, electrode placement, and task performance. Hemispheric language dominance (abbreviated here as LD) was verified in all patients by intracarotid amobarbital testing, fMRI, and/or electrocortical stimulation mapping.

recording contacts and because of its low likelihood of functional responses. Channels were visually inspected and those identified as containing excessive noise were discarded. A common average reference, where each electrode was referenced to the grid or strip to which it belonged, was used to minimize spatial bias from the reference electrode. Signals were low-pass filtered to prevent aliasing, resampled to 256 Hz, and epoched by clipping from 250 ms before stimulus onset to 4000 ms post stimulus onset.

Spectral power was extracted using one of two different techniques, autoregressive estimation or the Hilbert transform, depending on the goal. Autoregressive estimation permitted the extraction of a broad range of frequencies at the cost of temporal resolution. Models that used all extracted frequency bands were trained and tested, as were models that used high-gamma features only. While models using all frequency bands achieved the best performance, models using high-gamma only performed nearly as well, and thus subsequent analyses explored models using only high-gamma features. Once we made this determination, Hilbert estimation was used to extract high-gamma features with higher time resolution, permitting analyses on the timing and cortical locations related to semantic processing (Buck, 1999).

Autoregressive spectral estimation was performed using the Burg method (Kay, 1999) with 500 ms windows and a 250 ms overlap. The log of the spectral power values were averaged over multiple frequency bins. Frequency bins consisted of delta (1–4 Hz), theta (4–8 Hz), alpha (8–13 Hz), beta (15–30 Hz), gamma (30–50 Hz), and high-gamma (70–110 Hz). The autoregressive filter order was set to 26, and spectral estimation was performed with a

frequency resolution of 2 Hz. To estimate high-gamma features using the Hilbert transform data was first forward and backward filtered to a passband of 70–110 Hz using a 3rd order Butterworth filter. The Hilbert transform was used to generate the analytic signal, and the magnitude of this signal was squared to calculate signal power. Features were then calculated by averaging over 250 ms windows, sliding every 31.25 ms.

5.2.2 Encoding Model

Linear ridge regression was used to learn the encoding model parameters (Chen et al., 2014; Hastie, Tibshirani, and Friedman, 2009) relating semantic attribute ratings (Section 5.2.2.2) to neural activity features (Section 5.2.2.1). Linear ridge regression is a least-squares technique that employs regularization via an l_2-norm penalty, where this penalty effectively biases coefficients toward zero in exchange for reducing the variance on their estimates. Regularization is usually necessary in regression problems involving high-dimensional data as a safeguard against over-fitting. Our encoding model predicted neural features from semantic features by linear weighting:

$$n_m = \mathbf{s}^T \mathbf{b}_m + \epsilon_m$$

Where n_m was the mth neural feature, $\mathbf{s}$ was a vector of semantic features associated with the stimulus, $\mathbf{b}_m$ was a vector of weights, and ϵ_m was an error term. The ridge regression solution for determining the weights was given by

$$\begin{aligned}\mathbf{b}_m &= \underset{\mathbf{b}_m}{\operatorname{argmin}} |\mathbf{S}\mathbf{b}_m - \mathbf{n}_m|_2^2 + \lambda \, |\mathbf{b}_m|_2^2 \\ &= \left(\mathbf{S}^T\mathbf{S} + \lambda \mathbf{I}\right)^{-1} \mathbf{S}^T \mathbf{n}_m\end{aligned}$$

where $\mathbf{n}_m$ was a vector of mth neural features for a set of trials, $\mathbf{S}$ was

a J×K matrix of K=218 semantic features for J=354 trials (J=354 rather than 360 because six trials for a held-out object are not used when training the model), and λ was the regularization parameter. The semantic feature matrix may contain row vectors for repeated trials. Neural feature vectors, $\mathbf{n}_m$, were normalized to zero mean and unit variance over all trials. Initial testing using a leave-one-out cross-validation to optimize λ from seven logarithmically-spaced values between 1 and 1000 often produced λ=3.16, and so this value was adopted for all models. This determination was made in initial testing of data from P2 using earlier versions of the encoding model and was not specifically optimized to any models or results reported here. Fitting patient-specific λ values might have improved overall performance of the encoding models slightly.

5.2.2.1 Neural Features

Each neural feature used by the encoding model can be described as the signal power from a specific electrode in a specific frequency band during a specific time window. To limit the inclusion of neural features to those primarily associated with semantic processing, neural features were restricted to times 0–750 ms post stimulus onset. To determine the degree to which neural features may have been associated with spoken responses, additional analyses compared the timing of spoken responses with the semantic decoding performance over time. We found that peak decoding tended to occur before the vast majority of spoken response onsets (see Section 5.3.3 in Results).

Neural features were selected for their stability over stimulus presentations, similar to correlation-based stability measures used with fMRI to select voxels (Mitchell et al., 2008; Shinkareva et al., 2008). This approach was chosen because it is computationally straightforward and is commonly used in similar studies. In this procedure, neural features are selected that have stable response profiles across repeated presentations of the same item. Correlation stability was calculated for a neural feature by averaging all pairwise Pearson correlations between responses in blocks of trials. For example, for a data set with 60 objects and six blocks of object presentations, the correlation stability of a neural feature is produced by calculating the correlation between the 60 responses in block i to the 60 responses in block j and averaging correlation coefficients for all possible i, j pairs (15 in total). The most stable neural features were selected for use in the encoding model, up to 200 features. Features were always selected on training data as part of a nested cross-validation process.

5.2.2.2 Semantic Features

Encoding models used semantic features from the human218 semantic knowledge base (Palatucci et al., 2009; Sudre et al., 2012) consisting of 218 interpretable semantic attributes. This semantic model was chosen because the 218 meaningful dimensions facilitated analysis of semantic dimension encoding at specific cortical sites (see Section 5.2.3.2). Attribute ratings were collected by Palatucci et al., 2009 by asking a series of 218 questions to a group of Amazon Mechanical Turk users about 1000 different nouns, including the 60 objects included in this study. Questions probed a variety of semantic properties, including size, usage, composition, and category, with answers on an ordinal

scale from 1 to 5 (definitely not to definitely yes), and then rescaled to a range of -1 to 1. Lastly, the 218-element vector for each object was scaled to unity length. Each object was represented in this 218-dimensional semantic space in subsequent analyses.

5.2.2.3 Model Validation

For each model, performance was assessed via rank accuracy for decoding held-out objects. To decode a held-out neural activity pattern $\bar{\mathbf{n}}$ generated by an untrained object, $\bar{\mathbf{n}}$ was compared via cosine distance to a set of predicted neural activity patterns generated by applying encoding model β to the semantic attribute vectors for the 60 objects. These distances determined the relative ranks of the 60 objects, and rank accuracy for the correct object ranked ith among 60 objects was computed as 100*(60-i)/59. Within-category rank accuracies were also computed by limiting the possible objects to the correct object and the remaining four objects falling in the same semantic category as the held-out object. Rank accuracy is one of several possible metrics for assessing encoding model performance; qualitatively similar results were achieved using alternative metrics (e.g. leave-two-out PairedPerf, as in Mitchell et al., 2008).

Training and testing of encoding models consisted of three phases: feature selection, model training, and model testing. Model training, i.e. estimation of model weights, was always performed using individual trials, rather than aggregating the six presentations to produce a single neural activity pattern for an object. Feature selection and model testing were performed using one of

four different conditions: (1) responses averaged over all six presentations and from all frequency bands, (2) responses averaged over all six presentations and from high-gamma only, (3) single-trial responses from all frequency bands, and (4) single-trial responses from high-gamma only. The testing for the response-averaged condition involved calculating rank accuracies by comparing 60 predicted neural activity patterns to a neural activity pattern averaged over six recordings, while the testing for the single-trial condition compared predicted neural activity patterns to a single recording six times.

Features were selected via cross-validation to either optimize the single-trial rank or the response-averaged rank accuracy. A nested leave-one-object-out cross-validation procedure was used for all model validation. The 60-fold outer loop of the procedure was used to iterate over each of the 60 objects as the held-out object. For each of these folds, a 59-fold inner loop was used to select the optimal neural feature set for the held-out object. For a given fold of the outer loop, the correlation stability was calculated for each neural feature. Then, for each fold of the corresponding inner loop, mean rank accuracy (MRA) was calculated for encoding models using between 1 and 200 of the features with the highest correlation stability (logarithmically spaced). The MRA curves were averaged across folds of the inner loop, and the number of features that produced the maximum MRA was adopted for that outer fold.

Rank accuracy results were statistically thresholded using a Monte Carlo procedure. A null model was trained for each patient and for each testing condition by permuting the rows of the semantic attribute matrix, effectively shuffling the noun labels. All model estimation and validation procedures

were replicated, and in each case, the procedure was repeated 50,000 times to produce a null probability distribution. P-values were calculated for each patient and decoding condition by computing the fraction of null distribution values that were greater than the actual MRA (while adding 1 to both the numerator and denominator to prevent p=0). An analogous procedure was performed to determine within-category rank accuracy thresholds, but for this Monte Carlo procedure, shuffling was done within categories rather than across all noun labels.

5.2.3 Feature Analyses

5.2.3.1 Informative Time Points

To assess the time course of semantic attribute information in the ECoG signal relative to spoken responses, models were trained and tested for neural features restricted to individual temporal windows. Only high-gamma features calculated using the Hilbert transform were considered, because models using only high-gamma features performed substantially better across subjects than models restricted to any other single frequency band (see Supplementary Figure A.1B). Additionally, given the goal of identifying the temporal windows with the most semantic content, the Hilbert transform method allowed for higher temporal resolution in our neural features, and high-gamma features intrinsically have a higher temporal resolution than lower frequency features. A sliding window of 250 ms and a step size of 31.25 ms was adopted, and the timing of the onset of significant rank accuracy as well as the peak was

recorded for high-gamma that was time-locked to stimulus onsets. We performed a similar analysis on high-gamma locked to spoken response onsets. Response onsets and offsets were determined manually by listening to the spoken responses and using simultaneous visual inspection of the speech recording's spectrotemporal content. Time points were conservatively chosen to ensure articulation was entirely captured. Significance was determined by performing a single-tailed rank-sum test to compare rank accuracies at each time window to baseline, which was defined as time points between -1000 and -125 ms relative to stimulus onset (because of the 250 ms window, baseline actually contained information from -1125 to 0 ms). Bonferroni multiple comparisons corrections were applied individually for each patient across all time points.

For each window, the top 200 features were selected using correlation stability with proper cross-validation. Within temporal window analyses, the number of neural features was not selected within cross-validation, due to computational cost; rather, models were trained and tested with varying numbers of neural features, and the maximum MRA was chosen. Due to non-causality of spectral estimation (250/2=125 ms from windowing during Hilbert feature extraction), a given temporal window could have contained some information from future temporal windows. To account for this, results are also reported with respect to the leading edge of the extraction window (Table 5.3).

5.2.3.2 Informative Electrodes

To determine where evoked responses were well-predicted by semantic attributes, we estimated encoding models that mapped semantic attributes to high-gamma responses at individual electrodes. This analysis focused on the three patients with the highest performing encoding models and was restricted to the high-gamma features from the 250 ms time window with the highest MRA for each patient. Cross-validation was used to estimate separate encoding models for each individual electrode, and predicted high-gamma values were calculated for each object. For each channel, we calculated Pearson correlations between the set of 360 actual high-gamma values (six presentations of 60 objects) and the set of 60 predicted high-gamma values (replicated six times to match the set of actual neural features). For a given channel, the resulting correlation coefficient indexed the degree to which high-gamma variance at that channel was accounted for by semantic attributes. Finally, p-values for correlation coefficients were calculated through Monte Carlo simulations for each patient that replicated the sensitivity analysis procedure with 10,000 null semantic attribute matrices (formed through row shuffling). P-values were identified as significant using FDR correction ($\alpha = .05$) across all electrodes for all three patients.

To assess what semantic information was reliably encoded in the ECoG signal, we examined the correlations between a reduced set of semantic features and the high-gamma features recorded at the reliably informative electrodes. We used the high-gamma features directly (rather than the model weights) to eliminate the impact of specific choices for regularization. The human218

semantic attributes are highly redundant, so to reduce the number of informative attributes, we applied principal component analysis (PCA) on the full semantic matrix (218 semantic dimensions by 1000 nouns), and mapped the 60 nouns from the original human218 semantic attributes to PC semantic attributes. PCA was applied to the normalized human218 vectors, and the resulting PC semantic vectors were also normalized to unity length. Based on fraction of variance explained, 14 of the 218 PC attributes or dimensions were significant ($p<0.05$) when thresholding using 1000 iterations of a Monte Carlo simulation comparing the PC attributes to PCs found with null semantic attribute matrices (shuffled across nouns and attributes). Semantic attributes will be provided upon request.

Visual inspection of the significant PCs revealed that the first four components (fraction of variance explained: 16.5%, 12.5%, 6.3%, and 3.6%) were readily interpretable. For each of these four PCs, we listed the nouns with the largest and smallest (most negative) values for the PC, as well as the human218 attributes with the largest and smallest projections onto the PC (inner products). Based on this information, these PCs or semantic dimensions were subjectively labeled man-made, large, manipulable, and edible respectively (see Table 5.2). Note that these labels reflected the positive values of the PCs. In some cases, negative values of a PC indicated a semantic meaning that was intuitively opposite of the label (e.g., an intuitive opposite of man-made is alive), whereas for other PCs, the meaning of negative values was less clear (e.g., an intuitive opposite of edible might be inedible, but based on the attribute and object lists, this might be better labeled threatening).

	PC1: Manmade		PC2: Large	
	Nouns	**Attributes**	**Nouns**	**Attributes**
High	tray	manmade	train	bigger than microwave oven
	clipboard	manufactured	factory	bigger than loaf of bread
	frame	invented	hotel	bigger than bed
	box	can use it	museum	taller than person
	magnet	has corners	capitol	bigger than car
Low	antelope	has a face	grape	can be easily moved
	raccoon	conscious	cinnamon	lightweight
	deer	once alive	kernel	can pick it up
	ape	alive	raisins	can hold it
	hyena	grows	spice	can hold it in one hand

	PC3: Manipulable		PC4: Edible	
	Nouns	**Attributes**	**Nouns**	**Attributes**
High	ipod	has a front and back	fruit	used during meals
	toy	can be easily moved	watermelon	edible
	laptop	manufactured	pineapple	has internal structure
	guitar	can pick it up	tomato	goes in mouth
	phone	can hold it	grapefruit	tasty
Low	puddle	vegetable or plant	rust	silver
	fog	part of something larger	dandruff	hard to catch
	hill	bigger than house	measles	can cause you pain
	medow	bigger than bed	dent	dangerous
	valley	bigger than car	spark	avoid touching it

Table 5.2: Top four Principal Components (PC) from Human218. For each PC, the 5 objects with the highest values and the 5 objects with the lowest values for that component are listed, along with the 5 human218 attributes with the largest and smallest projections (i.e. inner products) on to that PC.

Correlation coefficients were computed between these four PCs and the responses from the informative high-gamma features using Pearson correlation. Coefficients were computed using all 60 objects and all six presentations for a total of 360 PC-feature pairs for each PC. Correlation p-values were calculated under a Gaussian distribution assumption.

5.3 Results

Zero-shot object decoding using a model trained to encode 218 semantic attributes as neural activity features was significantly better than chance for all nine patients, for nearly all decoding conditions (Figure 5.2). FDR correction ($\alpha = .05$) was applied across all 9 patients and 4 decoding conditions. The subjects (P1–P9) were arranged in decreasing order of MRA. Using recorded signals from all frequency bands aggregated across six presentations per object (to improve SNR), MRA across patients for held-out objects was 76% and ranged from 65% (P9) to 91% (P1), where chance rank accuracy is 50%.

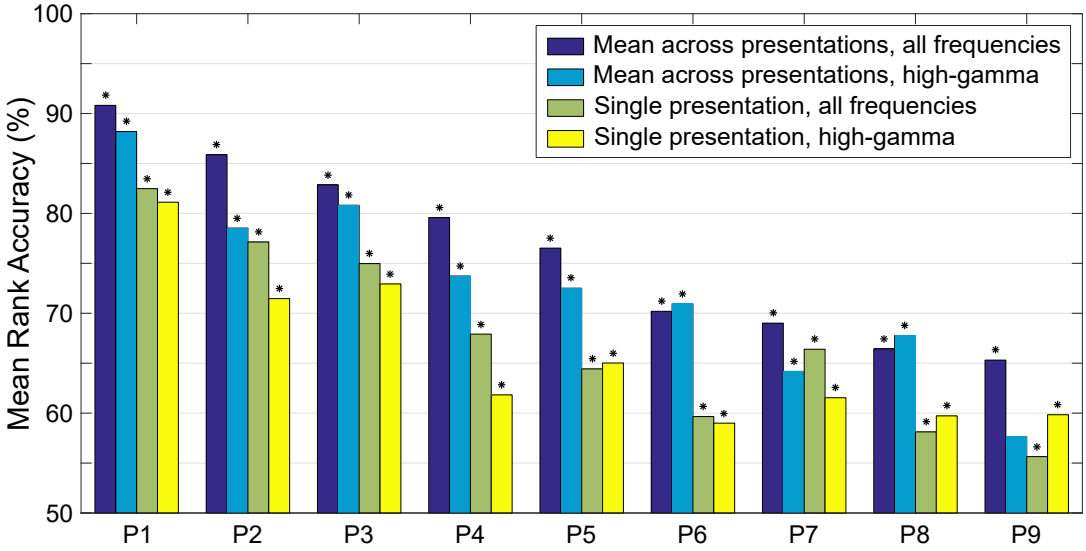

Figure 5.2: Zero-shot mean rank accuracy decoding performance by patient. Rank accuracies are reported for four encoding models: a full model that is trained and tested using all six presentations or trials of each object and all recorded frequency ranges, a high-gamma model that is trained and tested on all presentations of each object and only the high-gamma range, and restricted data versions of these models that are trained on all repetitions of objects in the training set, but tested using single presentations only. Accuracies were produced through leave-one-object-out cross-validation. Monte Carlo significance test procedures were used to calculate p-values for each condition, and FDR correction ($\alpha = .05$) was applied to correct for multiple comparisons. Asterisks (*) denote significant results.

Performance remained significantly better than chance for 8 of 9 patients when encoding models used high-gamma activity only (70–110 Hz) and trials were aggregated across presentations, and high-gamma models for all 9 patients were significant when decoding from individual trial responses. Compared to models using all frequency bands, high-gamma model mean rank accuracy across patients fell only slightly to 73% with a range of 58% (P9) to 88% (P1). Semantic information was still extractable under low SNR, high variability conditions of single trial testing, but mean rank accuracy fell substantially (mean=67%, range=56–82%; high-gamma mean 66%, range=59–81%). Still, rank accuracy from the patient with the best performing encoding models (P1) remained remarkably high for single-trial decoding (82%).

We also calculated winner-take-all accuracy where classifications are scored as correct only when the target object occupies the top overall rank. These results can be found in Supplementary Figure A.2.

5.3.1 Semantic Resolution of the Encoding Model

To assess whether the learned models encoded semantic detail beyond the basic semantic category associated with each object, we calculated within-category rank accuracies for all patients and for all objects. For example, when decoding the left-out neural activity pattern evoked by the item butterfly, we rank ordered only the five objects from the stimulus set from the insect category (ant, bee, beetle, butterfly, and fly), rather than the entire set of 60 objects. Within-category rank accuracies that are reliably higher than chance (i.e. 50%) would therefore indicate that the model was encoding semantic

detail of a finer grain than category identity.

We restricted our analysis to conditions where neural responses were aggregated across 6 presentations for both all-frequency and high-gamma models. Results were mixed (see Supplementary Figure A.3), with mean rank accuracy (MRA) ranging across patients from 38% to 66% using all frequencies and from 39% to 67% using high-gamma alone. Within-category rank accuracies using all frequencies were significantly better than chance in 4 patients: P1 (61%), P2 (63%), P3 (66%), and P4 (65%). Using high-gamma encoding models, within-category rank accuracies were significantly better than chance in two patients: P1 (67%) and P3 (65%). Significance was determined by applying FDR correction ($\alpha = .05$) across all 9 patients and both decoding conditions. These results suggest that under some circumstances, object-specific semantic information beyond category-level semantics is extractable from ECoG.

5.3.2 Informative Time Points

Semantic processing during visual object naming has been demonstrated as early as 110 ms after stimulus onset (Clarke et al., 2015), and has been shown to continue through the spoken response (Chen et al., 2016). To give encoding models access to the full time course of semantic processing, while at the same time limiting access to spoken output processing, decoding results reported thus far were limited to neural features starting at stimulus onset and ending at 750 ms. To better measure the time course of semantic activity and its relation to spoken responses, a sliding window decoding approach was also tested. Because the temporal resolution of power estimates in the

high-gamma frequency range is intrinsically better than that of estimates in lower frequencies, we focused our analysis on decoding accuracies using high-gamma frequencies only.

In all patients except P7, high-gamma encoding model performance from at least one individual window performed significantly better than chance (Figure 5.3). Significance was determined by comparing each individual time point to the baseline (pre-stimulus) distribution and applying a Bonferroni correction across all time points for a single subject. Onset of significant MRA ranged from 94 ms (P2 and P3) to 1156 ms (P8), with a median of 235 ms. Peak MRA ranged from 313 ms (P1 and P2) to 1156 ms (P8) with a median of 407 ms. Out of 8 subjects with significant MRAs, 5 patients (P1-4, P6) had MRA peak accuracies that occurred before any spoken responses. Even when accounting for the non-causality of spectral estimation (Table 5.3), 3 patients (P1-2, P6) had MRA peaks that preceded the earliest spoken response. We also explored response-locked MRA, and found that 6 out of 9 subjects had MRAs that were significantly better than chance (see Supplementary Figure A.4). The onset of significance for these results preceded speech onset for 5 of these 6 subjects.

5.3.3 Informative Electrodes

For the three patients with the highest performing encoding models (P1, P2, and P3), high-gamma activity from individual electrodes was analyzed for semantic attribute information. For each patient, the 250 ms window with the maximum overall MRA was selected for further analysis: 188–438 ms (centered at 313 ms) for P1 and P2, and 219–469 ms (centered at 344 ms)

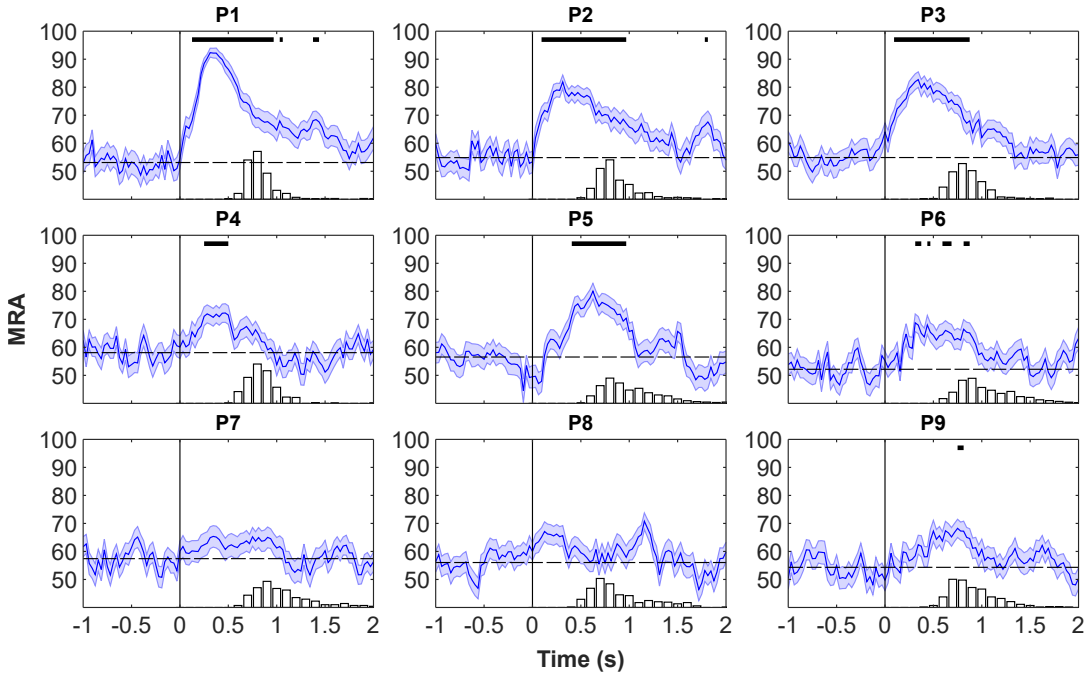

Figure 5.3: Decoding accuracies for sliding windows time-locked to stimulus onsets. Bar graphs are histograms of speech onset times. MRA traces were calculated using a sliding window of 250 ms and a step size of 31.25 ms. The plotted times correspond to the center of the extraction window, and thus may contain information spanning -125 to +125 ms about the center. Dashed lines represent patient-specific chance performance, calculated as the mean MRA during the baseline period. Black lines above each trace indicate windows where MRA was significantly greater than baseline.

for P3. For all three patients analyzed, high-gamma responses during the optimal decoding windows reliably encoded semantic attribute information along the left (dominant) ventral visual pathway (Figure 5.4). Responses from the left fusiform (P1, P2, P3), inferior temporal gyrus (P1, P2), and the parahippocampal gyrus (P1, P3) were significantly predicted by the semantic attribute encoding models. While there was less agreement across patients beyond the ventral visual pathway, high-gamma responses reliably encoded semantic attribute information at middle and superior temporal electrodes

	Significant windows (ms)		Speech onsets (ms)		Speech onsets before peak (%)	
Patient	**Onset**	**Peak**	**Earliest**	**Median**	**Uncorrected**	**Corrected**
P1	125	313	499	792	0.0%	0.0%
P2	94	313	486	816	0.0%	0.0%
P3	94	344	423	835	0.0%	0.0%
P4	250	469	494	833	0.0%	0.3%
P5	406	625	462	942	4.1%	18.4%
P6	219	313	573	991	0.0%	0.0%
P7	N/A	N/A	560	1005	N/A	N/A
P8	1156	1156	436	805	78.70%	84%
P9	500	500	392	855	29.30%	53.60%

Table 5.3: Peak decoding performance compared to speech onset times. The timing of significant decoding windows (both onset and peak) corresponds to the center of the 250 ms window used to estimate the high-gamma. The last column in this table accounts for this non-causality by adding 125 ms to the peak MRA time, and then calculating the fraction of speech onsets that occur before this adjusted time point.

as well as supramarginal electrodes in P1, and from several inferior frontal electrodes in P2.

Finally, we examined the semantic profiles of each of the significant electrodes in basal occipito-temporal cortex by calculating the correlation between the high-gamma responses during optimal decoding windows and each of the four top semantic PCs (in decreasing order: manmade, large, manipulable, and edible). For all three patients, high-gamma responses at multiple sites in basal occipitotemporal cortex were significantly correlated with the semantic dimensions associated with manmade/living and size distinctions (See Figure 5.3b). In all electrodes where the manmade dimension was significant, the size dimension was also significant. Furthermore, the signs of these two correlations were always matched, i.e. electrodes with positive manmade correlations also had positive correlations with the size dimension and vice

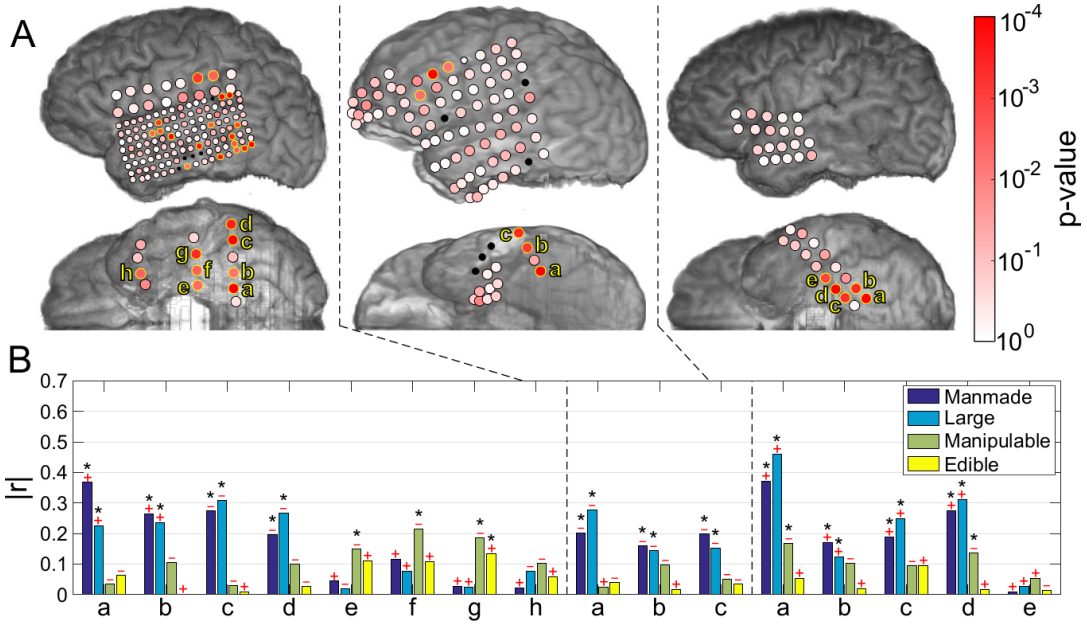

Figure 5.4: Significant electrode locations and semantic dimensions encoded in high-gamma activity for P1, P2, and P3. A) Encoding models were built for the top 3 patients that mapped from the semantic space to the high-gamma responses for each electrode. The red color scale represents the p-value of the correlation between predicted and observed high-gamma, with significant electrodes (FDR-corrected across all electrodes and subjects, $\alpha = 0.05$) indicated with a yellow ring. B) Bar plots report correlation coefficients (absolute value, with the sign of the correlation displayed above each bar) for each of the four identified semantic dimensions (i.e. PCs), for each of the significant electrodes along the basal occipitotemporal cortex. Asterisks (*) denote significant correlations (FDR-corrected across all significant electrodes and PCs, $\alpha = .05$).

versa. Note that nouns with negative loadings on the manmade dimension can readily be interpreted as living (see Table 5.2). All electrodes with positive correlations on the manmade and large dimensions are located medially on the basal occipito-temporal surface (P1: a and b; P3: a, b, c, and d). Conversely, electrodes with negative correlations on these two dimensions are located more laterally on the basal cortex (P1: c and d; P2: a, b, and c). Following these results, larger and manmade things evoke more high-gamma activity

medially and smaller and living things evoke more high-gamma activity laterally. Significant negative correlations with the manipulable dimension were observed in medial regions in or near the parahippocampal gyrus (P1: e, f, and g; P3: a and d). Nouns that loaded negatively on this dimension can be categorized as scenes and places (or in the specific case of our stimulus set, buildings, and to a lesser extent, building parts). A single basal electrode (g in P1) showed a positive correlation with the edible dimension. Lastly, we investigated the significant electrodes on the lateral surface of the brain in P1 and P2 (see Supplementary Figure A.5), though few patterns were discernible.

5.4 Discussion

Studies in other domains have shown that ECoG responses contain detailed information about a variety of representations, and that these responses can be used for decoding with varying degrees of performance (see Gunduz et al., 2012 for spatial attention; Hotson et al., 2016 for motor control; Martin et al., 2016 for speech production; Pasley et al., 2012 for speech perception). The series of results presented in the current study demonstrate that ECoG responses recorded during visual object naming are semantically rich, and untrained objects can be accurately decoded from these responses at rates equivalent to whole-brain fMRI in healthy subjects (despite variability in electrode placement and the presence of lesions in some patients). Beyond high rates of decoding accuracy, we observed that high-gamma activity recorded approximately 200–500 ms after stimulus onset was associated with specific semantic dimensions for a subset of patients with basal occipitotemporal

electrode coverage.

5.4.1 Frequency Encoding of Semantic Attributes

Semantic attribute information was consistently found in high-gamma band activity, and the addition of other frequency bands yielded only slight improvements to the trained encoding models. Oscillatory activity and synchronization in the gamma range (25–128 Hz) and especially the high-gamma range (defined as 70–110 Hz in the current report) appears to be critical for neural computation and communication (Fries, 2009). Activity in this frequency band has been strongly linked to different representations and processes, from low-level perceptual features to abstract conceptual features during visual processing (Jacobs and Kahana, 2009), voluntary motor commands (Cheyne et al., 2008), as well as language processing (Crone, Sinai, and Korzeniewska, 2006) where very accurate language mapping for neurosurgical patients has been demonstrated from high-gamma activity (Babajani-Feremi et al., 2016).

5.4.2 Timing of Semantic Attribute Information

The time course of semantic processing during object recognition can be estimated by tracking encoding model performance over time. The best semantic encoding models (i.e. for P1, P2, and P3) began performing significantly better than chance at a mean of 104 ms post stimulus onset, with accuracies peaking at a mean of 323 ms (using 250 ms windows centered at the reported times). Comparable results in visual object recognition have been reported with EEG (VanRullen and Thorpe, 2001; Simanova et al., 2010; Chan et al., 2011a), MEG

(Clarke et al., 2015; Cichy, Pantazis, and Oliva, 2016), and ECoG (Vidal et al., 2010; Chan et al., 2011b). Moreover, size and position-invariant visual object representations begin to emerge at 125 ms and 150 ms respectively (Isik et al., 2013), and more abstract semantic category or attribute information becomes available between 200 and 500 ms (Clarke et al., 2015).

5.4.3 Semantic Encoding in Basal Occipitotemporal Cortex

For the visual object naming task reported here, high-gamma responses from left basal occipitotemporal cortex were well predicted by our semantic attribute encoding models. Electrode placement and encoding model performance varied across patients, but the patients with the best performing encoding models all had electrode strips covering the left language-dominant fusiform gyrus. For all three of these patients, high-gamma activity recorded at fusiform electrodes 200–500 ms post-stimulus onset was significantly predicted by the semantic attributes of the named object. High-gamma activity from neighboring electrodes over inferior temporal and parahippocampal gyri were also significantly predicted in a subset of these patients.

These results appear to be in close agreement, both spatially and temporally, with other studies relating semantic attributes to neural responses (Chen et al., 2016; Sudre et al., 2012). Left fusiform involvement is commonly reported during visual semantic tasks like naming, reading, and categorization. While the region is perhaps most associated with visual word forms and orthographic processing (Tsapkini, Vindiola, and Rapp, 2011), findings that different semantic categories like animals and tools differentially activate the

fusiform gyrus are well-established (Simanova et al., 2014a; Ishibashi et al., 2016). It appears that this region links visual form to meaning in hierarchical processing stages from occipital cortex to the medial and anterior temporal lobe (Patterson, Nestor, and Rogers, 2007; Rogers et al., 2006; Starrfelt and Gerlach, 2007), and projects to distributed semantic representations throughout cortex (Binder and Desai, 2011).

5.4.4 Semantic Dimensions

Moving beyond strict categorical distinctions, our results show that information along particular semantic dimensions is encoded in basal occipitotemporal ECoG responses. While our encoding models used 218 attributes to predict ECoG responses, semantic dimensionality reduction was necessary to interpret the observed encoding patterns. PCA on the human218 database indicated that semantic variability could be mostly captured by four semantic dimensions: the degrees to which an object is manmade, large, manipulable, and edible. We used this semantic dimensionality reduction to interpret activity at the electrodes that were significantly predicted by the encoding model. Results showed that basal occipitotemporal responses in the high-gamma range were closely associated with the first three of these four dimensions, though responses from one anterior fusiform electrode for one patient was significantly correlated with values along the edible dimension.

For the dimensions labeled manmade and large, a medial to lateral functional organization was observed along basal occipitotemporal cortex. Values along these two dimensions positively correlated with high-gamma responses

from medial electrodes, and negatively correlated with responses from lateral electrodes. As an illustration of this organization, an airplane (with high values on the manmade and large dimensions) elicits more high-gamma activity in medial electrodes as compared to lateral electrodes, an ant (with low values on the manmade and large dimensions) elicits more high-gamma activity in lateral as compared to medial electrodes, and items that are split on these dimensions (e.g. a relatively small but manmade object like a spoon) elicit moderate high-gamma activity medially and laterally (on average). This pattern was fully observed in subject P1; subject P2 had only lateral coverage (with negative correlations for these dimensions); subject P3 had only medial coverage (with positive correlations for these dimensions).

Interestingly, the parcellation of fusiform gyrus into ventromedial and ventrolateral regions as suggested by these results is supported by independent functional and anatomical connectivity analyses (Zhang et al., 2016), as well as task-based fMRI results. The medial-to-lateral organization of ventral temporal cortex for the manmade/living or animate/inanimate distinction has been well studied (Chao, Haxby, and Martin, 1999; Downing et al., 2006; Bell et al., 2009; Wiggett, Pritchard, and Downing, 2009), and in other work, the large-to-small organization has also been observed along this axis (Konkle and Oliva, 2012). Konkle and Caramazza, 2013 varied animacy and size simultaneously across a large set of animals and objects, and found that medial regions preferentially responded to large objects (congruent with our results), while lateral regions preferentially responded to animals regardless of size (partially congruent with our results). There are many factors to consider

when resolving these results with the current report: relative to Konkle and Caramazza, animate items may have been under-represented in our stimulus set; many continuous dimensions were simultaneously varied here while animacy and size were treated as binary variables by Konkle and Caramazza; there may be individual differences in this organization that are not captured by our small patient group.

The semantic dimension labeled manipulable also correlated with high-gamma activity from multiple electrodes. Significant negative correlations for this dimension were largely observed at medial basal temporal sites, including parahippocampal cortex. Importantly, negative loadings on this dimension corresponded to places and geographic features in the data set used to generated the semantic dimensions, and primarily buildings and building parts within the 60 experimental stimuli. The parahippocampal place area is involved in processing place and scene information (Aguirre et al., 1996; Epstein and Kanwisher, 1998), consistent with our results.

5.4.5 Perceptual Features of Semantic Attributes?

Sensitivity to semantic attributes and categories in basal occipitotemporal cortex may be partially accounted for by differences or confounds in low-level visual features that exist between semantic categories. Indeed, the dimensions focused on here (e.g. animate-inanimate, large-small, and tools-places) can be associated with both visual and semantic concepts. For example, the animate-inanimate dimension may relate to visual features in that several of the semantic attributes that had the highest projections on to this dimension

had to do with visual structure and form (e.g. has corners, has flat or straight sides, has a face). Canonical size is another feature that has both semantic and perceptual interpretations: surprisingly, it was recently shown that canonical size information can be recovered from primary visual responses during a reading task (Borghesani et al., 2016), blurring the line between traditional distinctions of perceptual and semantic features.

While low-level features were not strictly controlled for or modeled in the current study, some studies of semantic decoding have attempted to account for low-level visual differences by including perceptual features in their models (Sudre et al., 2012; Clarke et al., 2015; Borghesani et al., 2016). These studies found posterior to anterior gradients from perceptual to conceptual representation such that posterior regions of basal occipitotemporal cortex contained information related to perceptual features, while regions just anterior contained information related to semantic features. Other studies have shown evidence for semantic representation in fusiform responses through semantic priming of words (Gold et al., 2006) and cross-modal generalization, where classifiers trained to discriminate animals from tools from left fusiform responses to one stimulus class (i.e. spoken names, written names, photographs, and natural sounds) can discriminate animals from tools using responses evoked by a different stimulus class (Simanova et al., 2014b). Most of these results come from fMRI; the only other study to date to analyze ECoG responses for semantic attribute information reported that semantic attribute models were much more predictive of neural responses than basic visual or phonological feature models, particularly in more anterior aspects of basal

temporal cortex (Chen et al., 2016).

5.4.6 Limitations of the Model

Very accurate decoding performance was achieved for untrained objects in a subset of patients, but our model was very modest in terms of the semantic embedding space, the neural features, and the statistical learning methods used to relate the two. Several different semantic embedding spaces have been used for predicting and interpreting neural data: those based on corpus statistics, those based on human judgments, and those that attempt to use neural responses themselves to define or optimize the embedding space for neural decoding (Fyshe et al., 2014). Different semantic embeddings are likely to be better matched for different recording modalities and paradigms, but whether there is substantial room for improvement beyond current results is unclear given the SNR and resolution achievable with today's neuroimaging tools (Bullinaria and Levy, 2013). In this work, we focused on a very limited set of concrete objects, but training encoding models for more complex concepts will require embeddings that can support more abstract concepts and concept compositionality.

5.5 Conclusion

Responses recorded with ECoG during visual object naming contain rich semantic attribute information that can be used to both decode untrained objects at very high levels of performance and study semantic encodings

within individual subjects. For a subset of patients with basal occipitotemporal electrode coverage, we observed that high-gamma activity recorded approximately 200–500 ms after stimulus onset was associated with specific semantic dimensions: manmade-animate, canonically large-small, and places-tools. Individual patient results were in surprisingly close agreement with reports from other modalities on the functional organization of semantic information in ventral temporal cortex during object recognition. Semantic attribute encoding models are powerful tools that are critical for both generalizing outside the training set as well as for allowing the study of semantic encodings among large sets of diverse categories.

References

Rupp, Kyle, Matthew Roos, Griffin Milsap, Carlos Caceres, Christopher Ratto, Mark Chevillet, Nathan E. Crone, and Michael Wolmetz (2017). "Semantic attributes are encoded in human electrocorticographic signals during visual object recognition". In: *NeuroImage* 148, pp. 318–329. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2016.12.074.

Rosch, Eleanor (1978). *Principles of categorization. Cognition and categorization, ed. by Eleanor Rosch & Barbara B. Lloyd, 27-48*. Hillsdale, NJ: Lawrence Erbaum Associates.

Binder, Jeffrey R., Lisa L. Conant, Colin J. Humphries, Leonardo Fernandino, Stephen B. Simons, Mario Aguilar, and Rutvik H. Desai (2016). "Toward a brain-based componential semantic representation". In: *Cognitive Neuropsychology* 33.3-4, pp. 130–174. ISSN: 0264-3294. DOI: 10.1080/02643294.2016.1147426.

Cree, George S. and Ken McRae (2003). "Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns)". In: *Journal of Experimental Psychology. General* 132.2, pp. 163–201. ISSN: 0096-3445.

Garrard, Peter, Matthew A. Lambon Ralph, John R. Hodges, and Karalyn Patterson (2001). "Prototypicality, distinctiveness, and intercorrelation: Analyses of the semantic attributes of living and nonliving concepts". In: *Cognitive Neuropsychology* 18.2, pp. 125–174. ISSN: 0264-3294. DOI: 10.1080/02643290125857.

Ruts, Wim, Simon De Deyne, Eef Ameel, Wolf Vanpaemel, Timothy Verbeemen, and Gert Storms (2004). "Dutch norm data for 13 semantic categories and 338 exemplars". In: *Behavior Research Methods, Instruments, & Computers* 36.3, pp. 506–515. ISSN: 1532-5970. DOI: 10.3758/BF03195597.

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman (1990). "Indexing by latent semantic analysis". In: *Journal of the American Society for Information Science* 41.6, pp. 391–407.

ISSN: 1097-4571. DOI: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and Their Compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. USA: Curran Associates Inc., pp. 3111–3119.

Pennington, Jeffrey, Richard Socher, and Christoper Manning (2014). "Glove: Global Vectors for Word Representation". In: *EMNLP*. Vol. 14, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

Pereira, Francisco, Samuel Gershman, Samuel Ritter, and Matthew Botvinick (2016). "A comparative evaluation of off-the-shelf distributed semantic representations for modelling behavioural data". In: *Cognitive Neuropsychology* 33.3-4, pp. 175–190. ISSN: 0264-3294. DOI: 10.1080/02643294.2016.1176907.

Mitchell, Tom M., Svetlana V. Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L. Malave, Robert A. Mason, and Marcel Adam Just (2008). "Predicting Human Brain Activity Associated with the Meanings of Nouns". In: *Science* 320.5880, pp. 1191–1195. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1152876.

Clarke, Alex, Barry J. Devereux, Billi Randall, and Lorraine K. Tyler (2015). "Predicting the Time Course of Individual Objects with MEG". In: *Cerebral Cortex* 25.10, pp. 3602–3612. ISSN: 1047-3211. DOI: 10.1093/cercor/bhu203.

Wehbe, Leila, Brian Murphy, Partha Talukdar, Alona Fyshe, Aaditya Ramdas, and Tom Mitchell (2014). "Simultaneously Uncovering the Patterns of Brain Regions Involved in Different Story Reading Subprocesses". In: *PLOS ONE* 9.11, e112575. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0112575.

Sudre, Gustavo, Dean Pomerleau, Mark Palatucci, Leila Wehbe, Alona Fyshe, Riitta Salmelin, and Tom Mitchell (2012). "Tracking neural coding of perceptual and semantic features of concrete nouns". In: *NeuroImage* 62.1, pp. 451–463. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2012.04.048.

Simanova, Irina, Marcel A. J. van Gerven, Robert Oostenveld, and Peter Hagoort (2014b). "Predicting the Semantic Category of Internally Generated Words from Neuromagnetic Recordings". In: *Journal of Cognitive Neuroscience* 27.1, pp. 35–45. ISSN: 0898-929X. DOI: 10.1162/jocn_a_00690.

Huth, Alexander G., Shinji Nishimoto, An T. Vu, and Jack L. Gallant (2012). "A Continuous Semantic Space Describes the Representation of Thousands

of Object and Action Categories across the Human Brain". In: *Neuron* 76.6, pp. 1210–1224. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2012.10.014.

Huth, Alexander G., Wendy A. de Heer, Thomas L. Griffiths, Frédéric E. Theunissen, and Jack L. Gallant (2016). "Natural speech reveals the semantic maps that tile human cerebral cortex". In: *Nature* 532.7600, pp. 453–458. ISSN: 1476-4687. DOI: 10.1038/nature17637.

Just, Marcel Adam, Vladimir L. Cherkassky, Sandesh Aryal, and Tom M. Mitchell (2010). "A Neurosemantic Theory of Concrete Noun Representation Based on the Underlying Brain Codes". In: *PLOS ONE* 5.1, e8622. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0008622.

Çukur, Tolga, Shinji Nishimoto, Alexander G. Huth, and Jack L. Gallant (2013). "Attention during natural vision warps semantic representation across the human brain". In: *Nature Neuroscience* 16.6, pp. 763–770. ISSN: 1546-1726. DOI: 10.1038/nn.3381.

Palatucci, Mark, Dean Pomerleau, Geoffrey Hinton, and Tom M. Mitchell (2009). "Zero-shot Learning with Semantic Output Codes". In: *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*. NIPS'09. USA: Curran Associates Inc., pp. 1410–1418. ISBN: 978-1-61567-911-9.

Manning, Jeremy R., Joshua Jacobs, Itzhak Fried, and Michael J. Kahana (2009). "Broadband Shifts in Local Field Potential Power Spectra Are Correlated with Single-Neuron Spiking in Humans". In: *Journal of Neuroscience* 29.43, pp. 13613–13620. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.2041-09.2009.

Ray, Supratim, Nathan E. Crone, Ernst Niebur, Piotr J. Franaszczuk, and Steven S. Hsiao (2008). "Neural Correlates of High-Gamma Oscillations (60–200 Hz) in Macaque Local Field Potentials and Their Potential Implications in Electrocorticography". In: *Journal of Neuroscience* 28.45, pp. 11526–11536. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.2848-08.2008.

Logothetis, Nikos K., Jon Pauls, Mark Augath, Torsten Trinath, and Axel Oeltermann (2001). "Neurophysiological investigation of the basis of the fMRI signal". In: *Nature* 412.6843, pp. 150–157. ISSN: 1476-4687. DOI: 10.1038/35084005.

Niessing, Jörn, Boris Ebisch, Kerstin E. Schmidt, Michael Niessing, Wolf Singer, and Ralf A. W. Galuske (2005). "Hemodynamic Signals Correlate Tightly with Synchronized Gamma Oscillations". In: *Science* 309.5736, pp. 948–951. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1110948.

Liu, Hesheng, Yigal Agam, Joseph R. Madsen, and Gabriel Kreiman (2009). "Timing, Timing, Timing: Fast Decoding of Object Information from Intracranial Field Potentials in Human Visual Cortex". In: *Neuron* 62.2, pp. 281–290. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2009.02.025.

Wang, Wei, Alan D. Degenhart, Gustavo P. Sudre, Dean A. Pomerleau, and Elizabeth C. Tyler-Kabara (2011). "Decoding semantic information from human electrocorticographic (ECoG) signals". In: *Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference* 2011, pp. 6294–6298. ISSN: 1557-170X. DOI: 10.1109/IEMBS.2011.6091553.

Chen, Y., A. Shimotake, R. Matsumoto, T. Kunieda, T. Kikuchi, S. Miyamoto, H. Fukuyama, R. Takahashi, A. Ikeda, and M. A. Lambon Ralph (2016). "The 'when' and 'where' of semantic coding in the anterior temporal lobe: Temporal representational similarity analysis of electrocorticogram data". In: *Cortex* 79, pp. 1–13. ISSN: 0010-9452. DOI: 10.1016/j.cortex.2016.02.015.

Pasley, Brian N., Stephen V. David, Nima Mesgarani, Adeen Flinker, Shihab A. Shamma, Nathan E. Crone, Robert T. Knight, and Edward F. Chang (2012). "Reconstructing Speech from Human Auditory Cortex". In: *PLOS Biology* 10.1, e1001251. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.1001251.

Mugler, Emily M., James L. Patton, Robert D. Flint, Zachary A. Wright, Stephan U. Schuele, Joshua Rosenow, Jerry J. Shih, Dean J. Krusienski, and Marc W. Slutzky (2014). "Direct classification of all American English phonemes using signals from functional speech motor cortex". In: *Journal of Neural Engineering* 11.3, p. 035015. ISSN: 1741-2552. DOI: 10.1088/1741-2560/11/3/035015.

Schalk, G., D.J. McFarland, T. Hinterberger, N. Birbaumer, and J.R. Wolpaw (2004). "BCI2000: A General-Purpose Brain-Computer Interface (BCI) System". In: *IEEE Transactions on Biomedical Engineering* 51.6, pp. 1034–1043. ISSN: 0018-9294. DOI: 10.1109/TBME.2004.827072.

Buck, Oppenheim / Schafer / (1999). *Discrete-Time Signal Processing*. TBS.

Kay, Steven M. (1999). *Modern Spectral Estimation: Theory and Application*. 1 edition. Upper Saddle River, N.J: Prentice Hall. ISBN: 978-0-13-015159-9.

Chen, Mo, Junwei Han, Xintao Hu, Xi Jiang, Lei Guo, and Tianming Liu (2014). "Survey of encoding and decoding of visual stimulus via FMRI: an image analysis perspective". In: *Brain Imaging and Behavior* 8.1, pp. 7–23. ISSN: 1931-7565. DOI: 10.1007/s11682-013-9238-z.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. Springer Series in Statistics. New York: Springer-Verlag. ISBN: 978-0-387-84857-0.

Shinkareva, Svetlana V., Robert A. Mason, Vicente L. Malave, Wei Wang, Tom M. Mitchell, and Marcel Adam Just (2008). "Using fMRI Brain Activation to Identify Cognitive States Associated with Perception of Tools and Dwellings". In: *PLOS ONE* 3.1, e1394. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0001394.

Gunduz, Aysegul, Peter Brunner, Amy Daitch, Eric C. Leuthardt, Anthony L. Ritaccio, Bijan Pesaran, and Gerwin Schalk (2012). "Decoding covert spatial attention using electrocorticographic (ECoG) signals in humans". In: *NeuroImage* 60.4, pp. 2285–2293. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2012.02.017.

Hotson, Guy, David P. McMullen, Matthew S. Fifer, Matthew S. Johannes, Kapil D. Katyal, Matthew P. Para, Robert Armiger, William S. Anderson, Nitish V. Thakor, Brock A. Wester, and Nathan E. Crone (2016). "Individual finger control of a modular prosthetic limb using high-density electrocorticography in a human subject". In: *Journal of Neural Engineering* 13.2, pp. 026017–026017. ISSN: 1741-2552. DOI: 10.1088/1741-2560/13/2/026017.

Martin, Stephanie, Peter Brunner, Iñaki Iturrate, José del R Millán, Gerwin Schalk, Robert T Knight, and Brian N Pasley (2016). "Word pair classification during imagined speech using direct brain recordings". In: *Scientific reports* 6, p. 25803.

Fries, Pascal (2009). "Neuronal Gamma-Band Synchronization as a Fundamental Process in Cortical Computation". In: *Annual Review of Neuroscience* 32.1, pp. 209–224. ISSN: 0147-006X. DOI: 10.1146/annurev.neuro.051508.135603.

Jacobs, Joshua and Michael J. Kahana (2009). "Neural Representations of Individual Stimuli in Humans Revealed by Gamma-Band Electrocorticographic Activity". In: *Journal of Neuroscience* 29.33, pp. 10203–10214. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.2187-09.2009.

Cheyne, Douglas, Sonya Bells, Paul Ferrari, William Gaetz, and Andreea C. Bostan (2008). "Self-paced movements induce high-frequency gamma oscillations in primary motor cortex". In: *NeuroImage* 42.1, pp. 332–342. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2008.04.178.

Crone, Nathan E., Alon Sinai, and Anna Korzeniewska (2006). "High-frequency gamma oscillations and human brain mapping with electrocorticography". In: *Progress in Brain Research*. Ed. by Christa Neuper and Wolfgang Klimesch. Vol. 159. Event-Related Dynamics of Brain Oscillations. Elsevier, pp. 275–295. DOI: 10.1016/S0079-6123(06)59019-3.

Babajani-Feremi, Abbas, Shalini Narayana, Roozbeh Rezaie, Asim F. Choudhri, Stephen P. Fulton, Frederick A. Boop, James W. Wheless, and Andrew C. Papanicolaou (2016). "Language mapping using high gamma electrocorticography, fMRI, and TMS versus electrocortical stimulation". In: *Clinical Neurophysiology* 127.3, pp. 1822–1836. ISSN: 1388-2457. DOI: 10.1016/j.clinph.2015.11.017.

VanRullen, Rufin and Simon J. Thorpe (2001). "The Time Course of Visual Processing: From Early Perception to Decision-Making". In: *Journal of Cognitive Neuroscience* 13.4, pp. 454–461. ISSN: 0898-929X. DOI: 10.1162/08989290152001880.

Simanova, Irina, Marcel van Gerven, Robert Oostenveld, and Peter Hagoort (2010). "Identifying Object Categories from Event-Related EEG: Toward Decoding of Conceptual Representations". In: *PLOS ONE* 5.12, e14465. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0014465.

Chan, Alexander M., Eric Halgren, Ksenija Marinkovic, and Sydney S. Cash (2011a). "Decoding word and category-specific spatiotemporal representations from MEG and EEG". In: *NeuroImage* 54.4, pp. 3028–3039. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2010.10.073.

Cichy, Radoslaw Martin, Dimitrios Pantazis, and Aude Oliva (2016). "Similarity-Based Fusion of MEG and fMRI Reveals Spatio-Temporal Dynamics in Human Cortex During Visual Object Recognition". In: *Cerebral Cortex* 26.8, pp. 3563–3579. ISSN: 1047-3211. DOI: 10.1093/cercor/bhw135.

Vidal, Juan R., Tomás Ossandón, Karim Jerbi, Sarang S. Dalal, Lorella Minotti, Philippe Ryvlin, Philippe Kahane, and Jean-Philippe Lachaux (2010). "Category-Specific Visual Responses: An Intracranial Study Comparing Gamma, Beta, Alpha, and ERP Response Selectivity". In: *Frontiers in Human Neuroscience* 4, p. 195. ISSN: 1662-5161. DOI: 10.3389/fnhum.2010.00195.

Chan, Alexander M., Janet M. Baker, Emad Eskandar, Donald Schomer, Istvan Ulbert, Ksenija Marinkovic, Sydney S. Cash, and Eric Halgren (2011b). "First-Pass Selectivity for Semantic Categories in Human Anteroventral Temporal Lobe". In: *Journal of Neuroscience* 31.49, pp. 18119–18129. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.3122-11.2011.

Isik, Leyla, Ethan M. Meyers, Joel Z. Leibo, and Tomaso Poggio (2013). "The dynamics of invariant object recognition in the human visual system". In: *Journal of Neurophysiology* 111.1, pp. 91–102. ISSN: 0022-3077. DOI: 10.1152/jn.00394.2013.

Tsapkini, Kyrana, Manuel Vindiola, and Brenda Rapp (2011). "Patterns of brain reorganization subsequent to left fusiform damage: fMRI evidence from visual processing of words and pseudowords, faces and objects". In: *NeuroImage* 55.3, pp. 1357–1372. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2010.12.024.

Simanova, Irina, Peter Hagoort, Robert Oostenveld, Van Gerven, and Marcel A. J (2014a). "Modality-Independent Decoding of Semantic Information from the Human Brain". In: *Cerebral Cortex* 24.2, pp. 426–434. ISSN: 1047-3211. DOI: 10.1093/cercor/bhs324.

Ishibashi, Ryo, Gorana Pobric, Satoru Saito, and Matthew A. Lambon Ralph (2016). "The neural network for tool-related cognition: An activation likelihood estimation meta-analysis of 70 neuroimaging contrasts". In: *Cognitive Neuropsychology* 33.3-4, pp. 241–256. ISSN: 0264-3294. DOI: 10.1080/02643294.2016.1188798.

Patterson, Karalyn, Peter J. Nestor, and Timothy T. Rogers (2007). "Where do you know what you know? The representation of semantic knowledge in the human brain". In: *Nature Reviews Neuroscience* 8.12, pp. 976–987. ISSN: 1471-0048. DOI: 10.1038/nrn2277.

Rogers, Timothy T., Julia Hocking, Uta Noppeney, Andrea Mechelli, Maria Luisa Gorno-Tempini, Karalyn Patterson, and Cathy J. Price (2006). "Anterior temporal cortex and semantic memory: Reconciling findings from neuropsychology and functional imaging". In: *Cognitive, Affective, & Behavioral Neuroscience* 6.3, pp. 201–213. ISSN: 1531-135X. DOI: 10.3758/CABN.6.3.201.

Starrfelt, Randi and Christian Gerlach (2007). "The Visual What For Area: Words and pictures in the left fusiform gyrus". In: *NeuroImage* 35.1, pp. 334–342. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2006.12.003.

Binder, Jeffrey R. and Rutvik H. Desai (2011). "The neurobiology of semantic memory". In: *Trends in Cognitive Sciences* 15.11, pp. 527–536. ISSN: 1364-6613. DOI: 10.1016/j.tics.2011.10.001.

Zhang, Wen, Jiaojian Wang, Lingzhong Fan, Yuanchao Zhang, Peter T. Fox, Simon B. Eickhoff, Chunshui Yu, and Tianzi Jiang (2016). "Functional organization of the fusiform gyrus revealed with connectivity profiles". In:

Human Brain Mapping 37.8, pp. 3003–3016. ISSN: 1097-0193. DOI: 10.1002/hbm.23222.

Chao, L. L., J. V. Haxby, and A. Martin (1999). "Attribute-based neural substrates in temporal cortex for perceiving and knowing about objects". In: *Nature Neuroscience* 2.10, pp. 913–919. ISSN: 1097-6256. DOI: 10.1038/13217.

Downing, P. E., A. W.-Y. Chan, M. V. Peelen, C. M. Dodds, and N. Kanwisher (2006). "Domain Specificity in Visual Cortex". In: *Cerebral Cortex* 16.10, pp. 1453–1461. ISSN: 1047-3211. DOI: 10.1093/cercor/bhj086.

Bell, Andrew H., Fadila Hadj-Bouziane, Jennifer B. Frihauf, Roger B. H. Tootell, and Leslie G. Ungerleider (2009). "Object Representations in the Temporal Cortex of Monkeys and Humans as Revealed by Functional Magnetic Resonance Imaging". In: *Journal of Neurophysiology* 101.2, pp. 688–700. ISSN: 0022-3077. DOI: 10.1152/jn.90657.2008.

Wiggett, Alison J., Iwan C. Pritchard, and Paul E. Downing (2009). "Animate and inanimate objects in human visual cortex: Evidence for task-independent category effects". In: *Neuropsychologia* 47.14, pp. 3111–3117. ISSN: 0028-3932. DOI: 10.1016/j.neuropsychologia.2009.07.008.

Konkle, Talia and Aude Oliva (2012). "A Real-World Size Organization of Object Responses in Occipitotemporal Cortex". In: *Neuron* 74.6, pp. 1114–1124. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2012.04.036.

Konkle, Talia and Alfonso Caramazza (2013). "Tripartite Organization of the Ventral Stream by Animacy and Object Size". In: *Journal of Neuroscience* 33.25, pp. 10235–10242. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.0983-13.2013.

Aguirre, Geoffrey K., John A. Detre, David C. Alsop, and Mark D'Esposito (1996). "The Parahippocampus Subserves Topographical Learning in Man". In: *Cerebral Cortex* 6.6, pp. 823–829. ISSN: 1047-3211. DOI: 10.1093/cercor/6.6.823.

Epstein, Russell and Nancy Kanwisher (1998). "A cortical representation of the local visual environment". In: *Nature* 392.6676, pp. 598–601. ISSN: 1476-4687. DOI: 10.1038/33402.

Borghesani, Valentina, Fabian Pedregosa, Marco Buiatti, Alexis Amadon, Evelyn Eger, and Manuela Piazza (2016). "Word meaning in the ventral visual path: a perceptual to conceptual gradient of semantic coding". In: *NeuroImage* 143, pp. 128–140. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2016.08.068.

Gold, Brian T., David A. Balota, Sara J. Jones, David K. Powell, Charles D. Smith, and Anders H. Andersen (2006). "Dissociation of Automatic and

Strategic Lexical-Semantics: Functional Magnetic Resonance Imaging Evidence for Differing Roles of Multiple Frontotemporal Regions". In: *Journal of Neuroscience* 26.24, pp. 6523–6532. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.0808-06.2006.

Fyshe, Alona, Partha P Talukdar, Brian Murphy, and Tom M Mitchell (2014). "Interpretable Semantic Vectors from a Joint Model of Brain- and Text-Based Meaning". In: *Proceedings of the conference. Association for Computational Linguistics. Meeting* 2014, pp. 489–499. ISSN: 0736-587X.

Bullinaria, John A. and Joseph P. Levy (2013). "Limiting Factors for Mapping Corpus-Based Semantic Representations to Brain Activity". In: *PLOS ONE* 8.3, e57191. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0057191.

Chapter 6

Discrimination of Utterances using Spatiotemporal Matched Filter Templates in Single-trials

This chapter was submitted as an article to a special issue of Frontiers in Neuroscience on electrocorticographic brain-computer-interfaces, and as of the time of writing this dissertation, is still under review.

6.1 Introduction

Keyword spotting (KWS) has recently come to the forefront of human-computer-interaction with the advent of voice-assist technologies such as Amazon Alexa, Apple's Siri, and Google's Assistant. All of these systems employ local, low-resource acoustic keyword search in real-time to detect a "wake word" that activates server-side speech recognition for interaction with an intelligent agent. These systems have been commercially successful and lauded for their ease of use. There are scenarios where voice-activated system interaction is suboptimal, especially when many speaking voices make the acoustic speech

recognition less reliable and socially awkward to use. The ability to trigger an intelligent agent or perform menu selections with low latency and high specificity using neural control is of great practical interest.

A number of studies of neural speech decoding motivate the selection of electrocorticography (ECoG) for neural keyword spotting. Bouchard et al., 2013 were the first to examine the organization of articulation in ventral sensorimotor cortex (vSMC) using high-density ECoG recordings. Their study revealed that high frequency activity in the high-gamma range (70-110 Hz) encodes precise movements of speech articulators with a high degree of temporal specificity. Mugler et al., 2014; Mugler et al., 2015 similarly characterized the articulatory representation in this area and further showed that this activity is more related to the gestural trajectories of specific muscles in the vocal tract than it is related to the specific keywords or phonemes articulated. Kanas et al., 2014 used high frequency content of speech-active areas of the brain to perform voice-activity-detection, or VAD – segmenting periods of speech from non-speech periods. Moreover, high-gamma activity from ECoG arrays was used as input to a language model and a small-vocabulary continuous speech recognition from neural signals was created in a study by Herff et al., 2015. Decoding of phonemic (Bouchard and Chang, 2014; Pei et al., 2011a) and gestural (Mugler et al., 2015; Lotte et al., 2015) content from vSMC has repeatedly been shown as well. These studies provide evidence that the dynamics of speech require the spatiotemporal resolution of intracortical electrophysiological recordings; features derived from non-invasive modalities do not modulate at rates necessary to make short-time inferences about articulatory

processes. This study employs subdural ECoG recordings to determine the feasibility of neural keyword spotting using high quality neural recordings as a proof of concept.

In building a neural keyword spotter, we were inspired by acoustic keyword spotting, where this has been accomplished in a variety of ways. Hidden Markov Models (HMM) have been applied to this problem extensively. HMM based real-time keyword spotting tends to use a silent state, a keyword state (or series of states) and a set of "garbage" states that capture typical non-keyword speech. In "whole-word" approaches, each state of the HMM represents an entire word (Rahim, Lee, and Juang, 1997; Rohlicek et al., 1989), whereas phonetic-based approaches (Manos and Zue, 1997; Bourlard, D'hoore, and Boite, 1994; Rohlicek et al., 1993) break down the keyword and non-keyword utterances into sequences of phoneme sub-models. A keyword has been identified in the window of interest if the state sequence prediction proceeds through a keyword state (for whole-word modeling) or sequence of phonetic states corresponding to a keyword. Using a phonetic-based model to perform neural keyword spotting is risky: according to Mugler et al., 2014, a full set of American English phonemes has only been decoded at 36% accuracy from implanted ECoG arrays, motivating a whole-word approach.

Keshet, Grangier, and Bengio, 2009 suggested a low-latency acoustic keyword spotting using a discriminative approach rather than a HMM-based probabilistic model. In this approach, a linear classifier is trained to maximize the margin between acoustic feature sequences containing keywords and others that don't. As detailed in the aforementioned study, this approach does

not rely on computationally intensive Viterbi decoding and achieves higher keyword spotting performance than HMM-based systems.

We have chosen to use a neural voice-activity detection combined with an adaptation of the aforementioned discriminative (non-HMM-based) approach to perform neural keyword spotting. A flowchart that describes the signal processing chain and two-step discriminative decoding pipeline is described in Figure 6.1. Application of neural features to existing acoustic KWS approaches requires a few modifications. For example, mel-frequency cepstral coefficients derived from a single spectrally-rich microphone recording are sufficient to perform acoustic keyword recognition; by contrast, there are many electrodes in an ECoG recording, each with a single time-varying "activation" signal, corresponding to changes in neural population firing rates, in turn indexed by changes in high frequency activity. These activations capture neural processes necessary to sequence, control, and monitor the production of speech, as opposed to acoustic features that capture discriminable aspects of spoken acoustic waveforms. The motor representations of speech that capture the dynamics of articulators, and the auditory representations of speech that capture phonetic content during self-monitoring but also activate during perceived speech, are of particular interest to a neural keyword spotting system.

A recent study by Ramsey et al., 2017 has significantly influenced the approach we've developed to capture the spatiotemporal dynamics of neural features for the purpose of informing keyword discrimination. In the study, Ramsey discriminated phonemes from high density ECoG recordings of vSMC

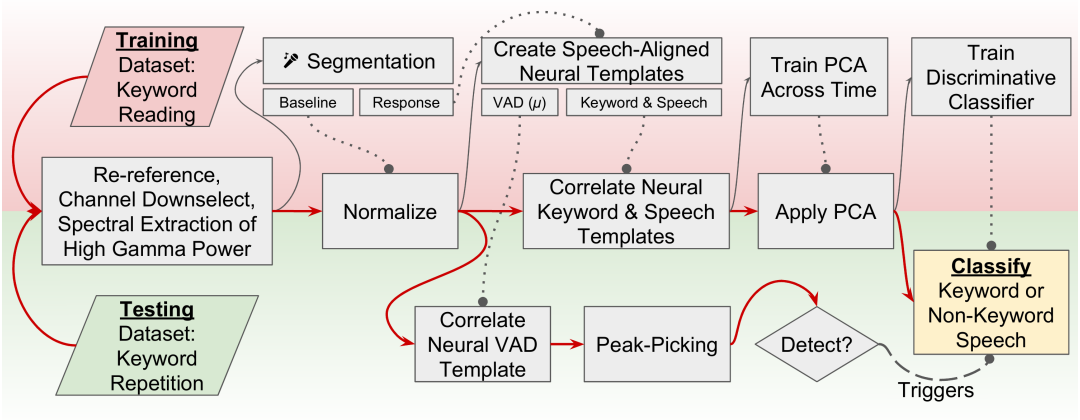

Figure 6.1: A flowchart representation of the keyword-spotting signal processing pipeline. Training and testing were split between two separate but related tasks. Red arrows indicate flow of data through the pipeline. Dotted lines with circles indicate models that were trained on the training dataset are used in this step for both the training and testing data. Training is performed using a visually presented keyword reading paradigm, and testing occurs across an auditory keyword repetition task. This study implements a two-stage detector; one neural VAD template is correlated across the testing dataset, and peak-picking indicates a detected utterance. When an utterance is detected, a discriminative classifier is used to decide if the utterance was a keyword or non-keyword speech. Channel downselection, normalization parameters, neural templates, feature dimensionality reduction, and classifiers are all trained on the reading (training) task and applied to the repetition (testing) task to simulate how keyword spotting would realistically perform in a separate recording session.

using the correlation of spatiotemporal matched filters as a means of identifying when the spatiotemporal pattern of high frequency activity matched stereotyped patterns for articulations (or gestural sequence of articulations). This method achieved 75% accuracy in a four-class phoneme discrimination problem, and highlighted the importance of including temporal relationships of high frequency activity between cortical sites in decoding models. We extend this methodology here to the creation of maximally discriminative "neural templates" to identify consonant-vowel "keyword" utterances instead of single phonemes.

"Wake-words" for voice-assist technologies are typically chosen to be low-frequency and phonetically complex to reduce the number of spontaneous detections. To simplify the problem of producing a more neurally detectable keyword, we examine monosyllabic, "consonant-vowel" keywords, varying the place of articulation, the consonant voicing, and the vowel height during phonation. We have chosen to examine keyword detection accuracy with respect to non-keyword speech and silence, as opposed to a multi-keyword decode to further simplify the problem and performance metrics. We will also limit ourselves to causal methods of feature extraction and classification for this study to realize how neural keyword detection would perform if deployed in a low-resource real-time scenario.

6.2 Materials and Methods

6.2.1 Data Collection

Subdural electrocorticographic recordings were made in eight subjects undergoing intracranial monitoring prior to resective surgery for drug-resistant epilepsy. Electrocorticographic (ECoG) arrays of platinum electrodes with varying exposed area and spatial density were placed for a one-to-two week period according to clinical requirements. Subjects performed both syllable-reading and syllable-repetition paradigms as part of a protocol approved by the Johns Hopkins University Institutional Review Board. All subjects gave written informed consent in accordance with the Declaration of Helsinki. Electrode localization was performed by aligning electrode locations from a post-operative computed tomography image with a pre-operative magnetic resonance image using Bioimage Suite (Papademetris et al., 2006). Neuroimaging and electrode locations are shown in Figure 6.2.

Subjects performed two tasks wherein they were asked to overtly produce consonant-vowel (CV) syllable utterances. In the (syllable) reading task, a textual representation of the utterance was visually presented for 1 second (see Table 6.2 for details) followed by an intertrial interval of 2-3 seconds during which the subject was instructed to fixate on a visible fixation cross. The (syllable) repetition paradigm was identical, except that the fixation cross remained on screen throughout the task and the utterance was aurally cued using a speaker. In both tasks, the subject was instructed to speak the prompted syllable aloud after stimulus delivery, and a microphone was used to record

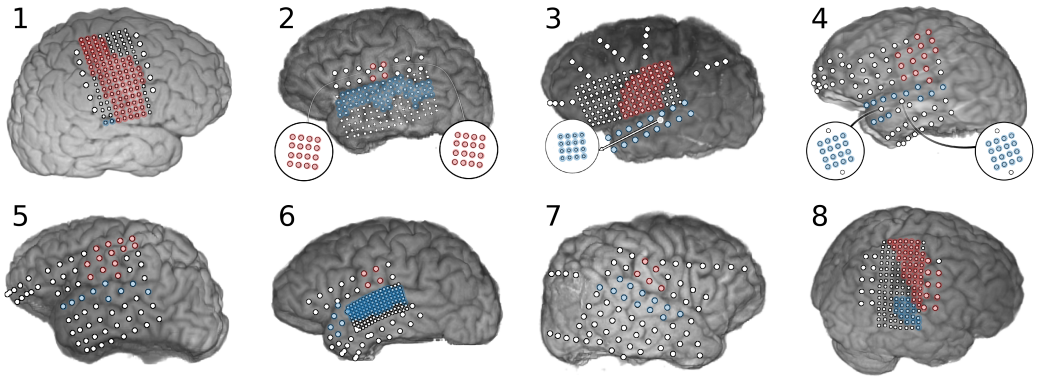

Figure 6.2: Neuroimaging and electrode localization for eight subjects implanted with subdural electrode arrays. Electrodes positioned over sensorimotor cortex are highlighted in red and electrodes over superior temporal gyrus are highlighted blue. Biographical and experimental details for these subjects can be found in Table 6.1. Subject 1 had a large lesion within pre-central gyrus, from which very little high frequency activity was recorded. Subject 3 had an ictal locus very near sensorimotor cortex with substantial inter-ictal activity that limited observation of neural features in this area. Subject 8 had a lesion in the right supramarginal gyrus.

the subject's responses to a high quality digital audio file. A monitor-output cable connected the microphone recording device to an auxiliary analog input on the electrophysiological amplifier (Neuroport, Blackrock Microsystems, Salt Lake City, UT; and EEG1200, Nihon Kohden, Tomioka, Japan), recording a lower-resolution version of the subject's speech synchronized with the ECoG data at 1000 samples per second. BCI2000 (Mellinger and Schalk, 2007) was used to present stimuli and record the data from the amplifier into a standardized format for offline analysis. Data was collected in blocks of 60 trials; 5 trials per utterance for all 12 syllables in a randomized order. The paradigm was split across two blocks of reading and two blocks of repetition for each subject, but time and clinical constraints limited collection to one-block of the tasks for some subjects. Details of data collection for each subject is documented in

Table 6.1.

6.2.2 Preprocessing and Segmentation

Noisy channels, identified via visual inspection of the raw ECoG signals, were removed from further analysis. Spatial filters were applied to re-reference recordings to the common-average of the included channels. Trial markers from BCI2000 that designated stimulus presentation (auditory or visual) were used to define the trial onset points. The 250-450 Hz band-power in the synchronized low-fidelity microphone recording captured the first formant of speech in each subject, and was thresholded to detect the utterance onset time for each trial. These threshold crossings tend to be associated with the voice-onset-time in CV keywords containing a voiced consonant and the plosive release in CV keywords containing an unvoiced consonant, due to the silent nature of consonant articulation. Templates were generated from a one-second "response" period centered around this threshold crossing to capture differences in the timing of neural features relative to the response onset (Ramsey et al., 2017; Mugler et al., 2014; Jiang et al., 2016). Neural features were normalized within each task individually to a pooled "baseline" period which was created from a one-second period prior to stimulus presentation across all trials within a single task. All trials from the reading dataset were used for training templates and classifiers that were applied across the repetition dataset. In this way, the training data were entirely separate from the testing data, and the templates generalized feature extraction across tasks.

ID	Side	Age	Sex	Read	Rep	Grid Specifications
1	R	17	M	120	60*	vSMC: 85 (85 HD-5) STG: 2 (2 HD-5) Total: 87
2	L	37	F	60	120	vSMC: 36 (32 μ, 4 SD) STG: 57 (57 HD-5) Total: 93
3	L	25	M	105**	120	vSMC: 30 (30 HD-5) STG: 32 (16 μ, 16 SD) Total: 62
4	L	39	M	120	120	vSMC: 14 (14 SD) STG: 48 (32 μ, 16 SD) Total: 62
5	L	40	M	120	120	vSMC: 13 (13 SD) STG: 9 (9 SD) Total: 22
6	L	40	F	60	120	vSMC: 4 (4 SD) STG: 87 (81 HD-3, 6 SD) Total: 91
7	R	27	M	120	60	vSMC: 5 (5 SD) STG: 12 (12 SD) Total: 17
8	R	19	M	120	120	vSMC: 52 (43 HD-5, 9 SD) STG: 19 (HD-5) Total: 71

Table 6.1: Biographical and experimental details for subjects. The implant hemisphere (side), age, sex, number of reading/repetition trials, and grid specifications for all eight subjects in the study are listed here. Associated neuroimaging and electrode localization can be found in Figure 6.2. Channels are delineated by region of interest, and further by the diameter of the electrode's exposed area, then by the inter-electrode spacing. SD: Standard macro-array (2 mm diameter, 1 cm pitch). HD-5: High density array (2 mm diameter, 5 mm pitch). HD-3: High density array (1 mm diameter, 3 mm pitch). μ: Micro-ECoG array (75 μm diameter, 1 mm pitch). (*120 trials were recorded, but the synchronized microphone recording failed for the second set of 60 trials. Neural keyword spotting can be applied to this second block, but ground truth timing metrics are unavailable.) (**Recording session ended early.)

/IPA/ (“Stim”)	**Bilabial**	**Alveolar**	**Velar**
Voiced	/ba/ (“BAH”)	/da/ (“DAH”)	/ga/ (“GAH”)
Unvoiced	/pa/ (“PAH”)	/ta/ (“TAH”)	/ka/ (“KAH”)

/IPA/ (“Stim”)	**Bilabial**	**Alveolar**	**Velar**
Voiced	/bi/ (“BEE”)	/di/ (“DEE”)	/gi/ (“GEE”)
Unvoiced	/pi/ (“PEE”)	/ti/ (“TEE”)	/ki/ (“KEE”)

Table 6.2: Keyword utterances and associated axes of articulation. Keyword utterances vary on three axes; three places of articulation, two ways of consonant voicing, and two vowel heights. Utterances are shown with their IPA notation as well as the visual text prompt as shown in the reading paradigm. "GEE" would typically be pronounced /ɟi/ but subjects were instructed to respond with /gi/ instead.

6.2.3 Feature Extraction and Electrode Downselection

Electrodes over sensorimotor cortex and superior temporal gyrus were manually identified by a neurologist; see Figure 6.2 for a summary. Electrodes lying outside these areas were excluded from further analysis. A 128 ms window sliding by 16 ms increments was used to perform spectral decomposition via the fast Fourier transform. Spectral power was log-transformed and z-scored to the baseline period, per-frequency. Frequency bins between 70 and 110 Hz were averaged together to form a time-varying feature capturing the band power modulations in the “high-gamma” range, a frequency range highly correlated with the firing of local neural populations (Ray et al., 2008). This feature was then re-normalized to the baseline period per-electrode.

6.2.4 Template Generation and Voice Activity Detection

Previous studies indicate that the timing of high gamma activity contributes significantly to decoding of speech from vSMC (Ramsey et al., 2017; Jiang et al., 2016). Neural templates were trained to capture spatiotemporal relationships of high gamma activity in an efficient, but causal representation. A "response template" was created by calculating the mean of the neural responses from all trials (N = 60-120) in the training dataset. A "keyword template" for each keyword was also created by calculating the mean of the neural responses for each of the keywords individually (5-10 trials). We additionally took advantage of our keyword design to create neural templates composed of higher trial counts across axes of articulation, as described in Table 6.2. The response template was then subtracted from each of these keyword templates, the resulting "discrimination template" captured spatiotemporal relationships that differed from the mean neural responses in the response template. A significance mask was created by z-scoring the condition mean (prior to subtraction of the response template) relative to the baseline period. A temporal smoothing kernel (hamming, 0.1 sec) was applied to reduce noise in the template before the significance mask was applied; elements with a z-score of less than 3.0 were set to zero to further reduce noise. The smoothed and regularized discrimination templates were correlated with the corresponding high-gamma features in both testing and training datasets – these features were further smoothed (hamming, 0.25 sec) to reduce the influence that slight timing mismatches could have on keyword discrimination. An example visualization of the generation of a discrimination template for bilabial keywords can be

found in Figure 6.3A. A principal component analysis (PCA) was trained to identify linear combinations of template output features that accounted for 90 percent of the variance across the entire reading task. Principle components of template outputs were calculated for both the reading and repetition datasets, reducing covariance in the template outputs and creating neural features which can be used for keyword discrimination.

Electrodes from STG were excluded from the response template; the resulting template was used as the neural VAD template. Auditory representations of speech in STG tend to have less specificity to self-generated speech and their inclusion in the VAD model can result in false-positive detections coincident with the perception of utterances, whether or not they are being produced. Neural VAD was calculated as the squared temporal correlation between the VAD template and the normalized high-gamma power. VAD output was further smoothed using a temporal smoothing kernel (hamming, 1.0 sec). A causal peak-picking algorithm was applied to identify utterance onset times – the derivative of the neural VAD signal was thresholded and the zero-crossing that follows a threshold crossing was chosen as the utterance detection time. Example templates and their corresponding correlational output are shown in Figure 6.3. Application of these templates to live neural features results in exactly one second of latency for neural VAD and keyword discrimination.

6.2.5 Discriminative Classification

A discriminative classifier similar to SVM, as described in great mathematical detail by Keshet, Grangier, and Bengio, 2009, was trained on the reading

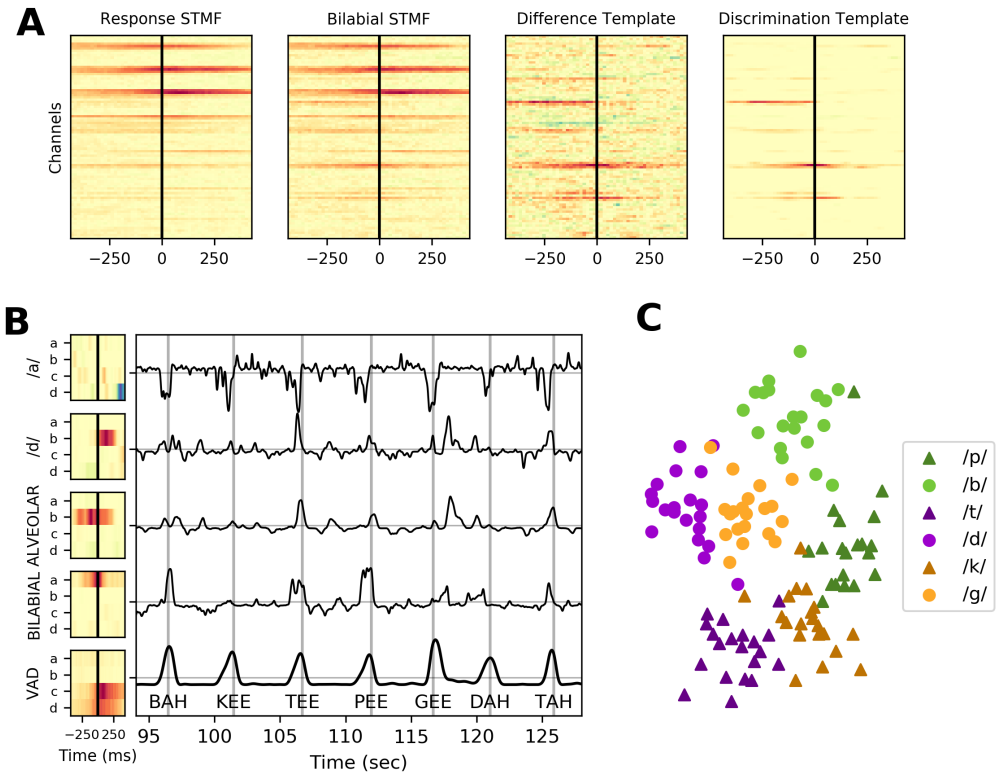

Figure 6.3: Example neural templates and utterance discrimination in Subject 1. (**A**) From left to right: the response spatiotemporal matched filter; an average of all keyword utterances, the bilabial spatiotemporal matched filter (STMF); an average of just keyword utterances with a bilabial place of articulation, the difference template; the subtraction of the response spatiotemporal matched filter from the bilabial spatiotemporal matched filter, and the discrimination template; the regularized and smoothed/denoised discrimination template for bilabial keywords. (**B**) Neural templates are shown for the four electrodes (**a**, **b**, **c**, and **d**) depicted in Figure 6.6. The VAD template, shown at the bottom, is the mean across all 120 trials in the task. The correlation of these templates with the high-gamma activity in the same task is shown in the plot to the right of the templates for a contiguous period of ~95 seconds to ~125 seconds into the reading task. Vertical grey lines in this plot indicate ground truth utterance times as recorded by a microphone. Peaks of the neural VAD output closely matched the utterance times. (**C**) The values of these template features reduced to two dimensions using MDS.

dataset. In broad strokes, the training step attempted to designate a linear discrimination boundary that maintains a constant margin of separation between pairs of feature-vectors corresponding to keyword and non-keyword utterances. For each pair, the training step searched for the feature-vector within +/- 100 ms of the alignment time for the non-keyword utterance that looked maximally "keyword-like", given the current discrimination boundary. The learning step adjusted the discrimination boundary using the difference between that maximized non-keyword feature-vector and the ground-truth keyword feature-vector. A significant advantage of this classifier is that it can be trained online as new observations become available.

Pairs of feature-vectors associated with keyword and non-keyword utterances were assigned within stimulus blocks. Additionally, feature-vectors associated with keyword utterances were paired with feature-vectors corresponding to silent periods (1.0 sec before stimulus onset) to adapt the classifier boundary to VAD false-detections during silent periods. In Figure 6.4, classifier output was calculated using ground-truth utterance detections derived from the microphone. During simulated testing, results of which are shown in Figure 6.5, the classifier output was calculated at times when the neural VAD model identified an utterance. The slight temporal misalignments between neural VAD and microphone-derived timing accounts for the different classifier performances between these figures.

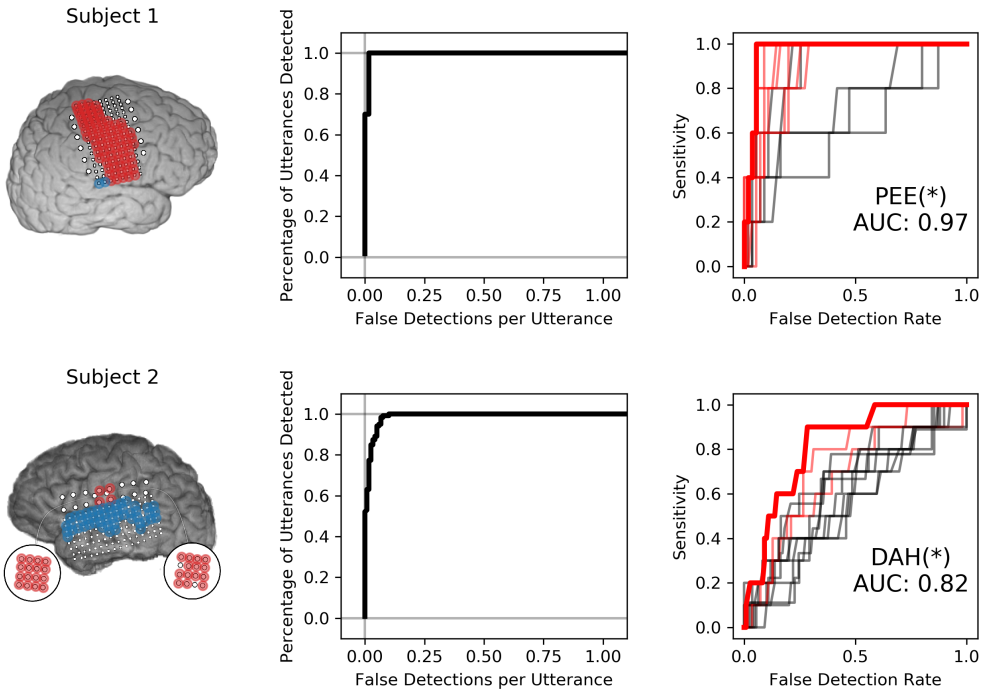

Figure 6.4: Isolated VAD and keyword discriminability for two subjects. The left-most panel shows the electrodes highlighted in red and blue that were used to discriminate syllables. Only electrodes highlighted in red were used to perform VAD. The center panel shows the VAD performance in sensitivity (percentage of utterance timings correctly identified) against the number of false detections per utterance for various VAD thresholds. The right-most panel shows ROC curves for all twelve keyword detectors. ROC curves with AUC values were significant at the 95% confidence interval are highlighted in red. The keyword detector that produced the highest AUC is highlighted in bold-red and indicated via annotation under the curves, followed by asterisks indicating significance at the $p < 0.05$ (*), $p < 0.01$ (**) and the $p < 0.001$ (***) level with respect to the distribution of maximum AUC models.

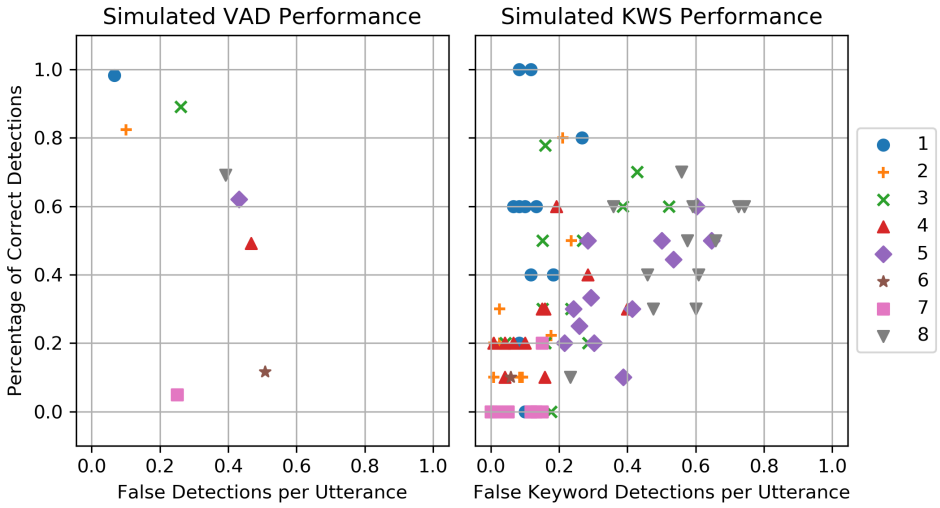

Figure 6.5: Simulated VAD and KWS performance on the testing dataset. On the left, the percentage of utterances correctly detected by neural VAD is plotted against the number of false utterance detections for all subjects in the study. On the right, performance for all twelve keyword spotters from each patient are plotted.

6.2.6 Testing and Performance Metrics

The templates, principle components, and discriminative classifiers were trained on all trials of the reading task. Testing and performance metrics were calculated from the application of these models to the repetition task. A VAD performance metric was calculated by sweeping the aforementioned VAD threshold value from 0 to 20 standard deviations (relative to baseline periods) and comparing the utterance detection times to the ground-truth microphone threshold crossings. An utterance detection within +/-100ms of a microphone event was classified as a true-positive, but subsequent detections for that utterance were considered false-positives.

An ROC curve was created for each of the keyword classifiers using

microphone-derived voice onsets in the repetition task. A classifier threshold was swept from -10 to 10 and the resulting keyword detections and false-positives were used to create an ROC curve and derive area-under-curve (AUC) metrics for each keyword classifier. Significance of the AUC statistic was calculated by scrambling the ground-truth utterance labels while training keyword detectors. A bootstrapped null-distribution of 1000 AUC metrics was generated for each keyword classifier, from which statistical significance thresholds for the metric were calculated. Keyword spotting performance using neural VAD times was also calculated for each classifier using a threshold that was chosen to maximize sensitivity while minimizing false detections—in particular, equalizing the error rates for false-negatives and false-positives, the so-called "equal error rate" condition (Motlicek, Valente, and Szoke, 2012)—on the training dataset.

6.3 Results

Within the context of this methodology, discrimination between keyword and non-keyword speech relies upon differences in timing and/or amplitude of high-gamma activity. Differences in high-gamma amplitude across keywords are useful in traditional decoding approaches where only single time-points of high-gamma activity are used to make classification decisions. Single-trial plots, as seen in Figure 6.6, suggest high-gamma amplitude in vSMC can be sufficient to decide the place-of-articulation for an utterance. Consonant voicing appears to be encoded in the timing of high-gamma activity relative to voice onset time. The sensation of pressure build-up in the vocal tract prior to

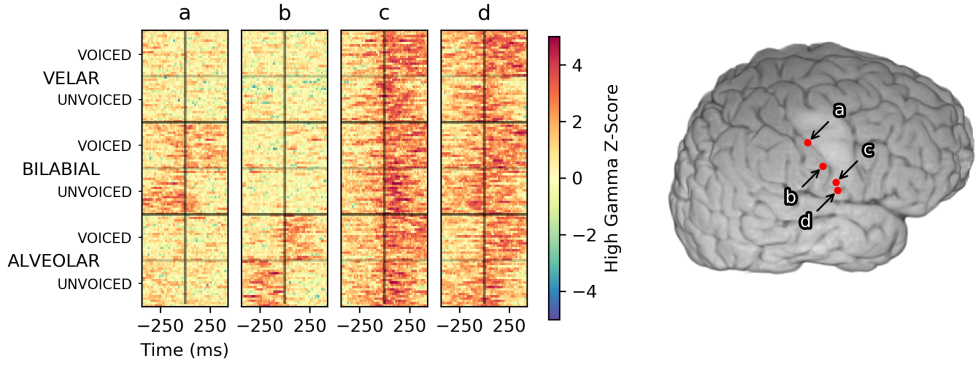

Figure 6.6: High-gamma single-trial rasters for Subject 1. High-gamma single-trial rasters across the reading task from four manually selected electrodes in Subject 1. Trials, plotted along the Y axis, were sorted first by the place of articulation for the consonant, then by consonant voicing. Trials were aligned with response-onset time set to 0 seconds, denoted by a black vertical line at the center of each raster. Color denotes the high-gamma feature z-score normalized to a pooled pre-trial baseline period. Activity in electrode **a** appears to represent a bilabial place of articulation, whereas activity in electrode **b** appears to indicate an alveolar place of articulation. Timing differences of high-gamma activity relative to the voice onset time encoded the voicing of bilabial and alveolar consonants in these areas. Electrode **c** exhibited consistent high-gamma amplitude and timing for all utterances; informing neural VAD but less useful for keyword discrimination. Electrode **d** appeared to encode consonant voicing across all places of articulation. No clear patterns emerged if the trials were sorted by vowel height (/a/ vs /i/) for any electrodes in Subject 1.

plosive release is a plausible explanation for the timing of this discriminable neural activation in electrodes **a** and **b**, especially given the placement of these electrodes in postcentral gyrus; an area typically associated with sensation.

The correlation of neural templates with high-gamma activity created high-level features that appeared to be useful for clustering utterances using these spatiotemporal relationships, as shown in Figure 6.3. The discriminative quality of a neural template appeared to rely primarily upon the number of trials used to create it; a decrease in the template noise was associated with a

higher number of trials. A neural template for a particular contrast highlights the difference from the mean template, which can be a problem if there is no discriminable difference between the contrasts. As seen in Figure 6.3, Subject 1 had very little discriminable activity within the vowel height condition (/a/ vs /i/), meaning the trial average across the '/a/' condition and the '/i/' condition were very similar to the trial grand-average. Subtracting the trial-average from the two condition averages resulted in a template that introduced significant noise to the feature set. The inclusion of these templates was less of a problem due to the following decomposition of these features into principal components; the noisy template outputs tended to be de-emphasized as they did not explain much of the variance of the features across time. Noting that template output appeared to fluctuate around neural VAD timings, temporal alignment was absolutely critical when interpreting these features.

Neural VAD and keyword discriminability appeared to be somewhat decoupled; several subjects showed consistent high-gamma modulation across utterances that was useful for performing VAD, but these features were less useful for keyword discrimination, as shown in Figure 6.4. Subject 1 exhibited exceptional VAD with highly significant discrimination of several keywords. Subject 2 showed similar VAD performance, but demonstrated relatively poor keyword discrimination. Classifiers in Subject 1 leveraged neural features that discriminated consonants well (shown in Figure 6.6), whereas classifiers from Subject 2 were only informed by features that discriminated vowel height and alveolar place of articulation, shown in Figure 6.7. VAD and keyword discrimination results for all subjects are shown in supplemental material.

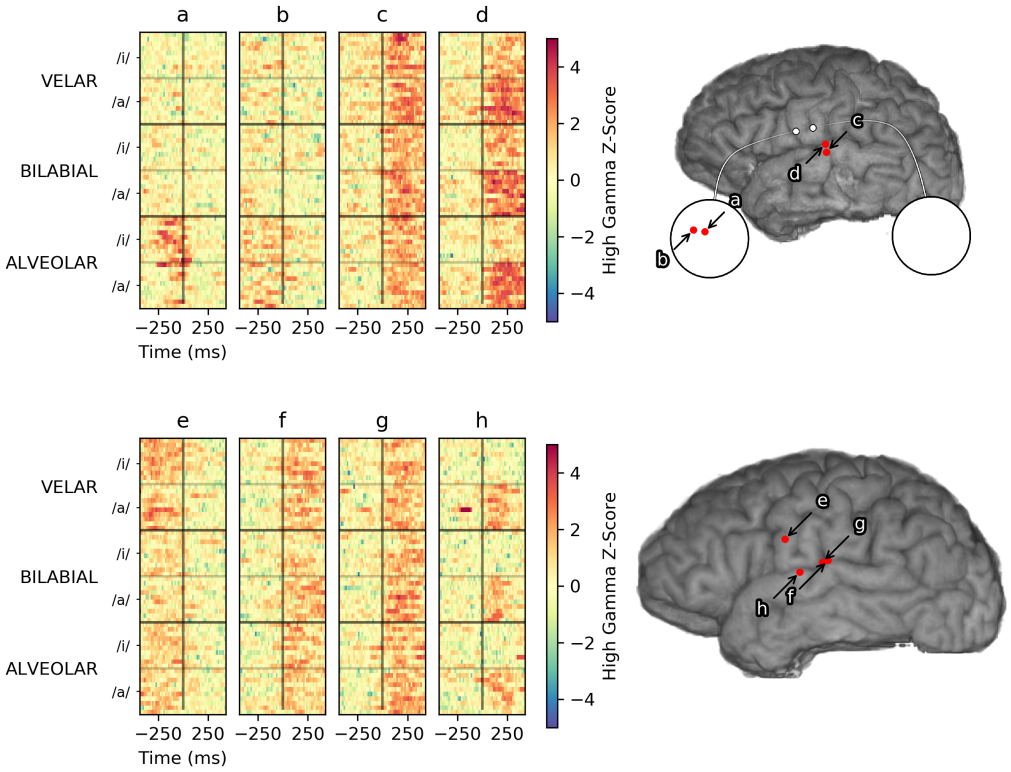

Figure 6.7: Vowel-specific high gamma activity. Vowel-specific high gamma activity from Subject 2 (top) and Subject 6 (bottom). Single-trial high-gamma rasters to the left are sorted first by place of articulation, then by vowel height. Electrode **d** and **h** appear to encode vowel height, in similar areas of STG. Electrode **a** and **b** are micro-ECoG electrodes over vSMC that appear to encode place of articulation. Electrodes **c**, **f**, and **g** appear to modulate consistently with all utterances and are more useful for VAD, but provide little discriminative information.

6.4 Discussion

This study is the first to examine keyword spotting with ECoG for the purpose of BCI control. These results were obtained by performing a two-step classification procedure involving neural VAD and keyword vs non-keyword-speech classification. As mentioned previously, neural voice activity detection has been performed before using spectral decomposition techniques and a discriminative classifier by Kanas et al., 2014. Performing VAD using this method of template-based "matched filtering" has a number of benefits over this prior work. Due to the fact that all utterances are roughly the same length and surrounded by silence, cross correlation with the neural VAD template actually provides a good alignment point for the application of a discriminative classifier. Furthermore, the cross correlation is computationally efficient and only relies on a peak-picking implementation to find utterances. The second-stage discriminative classifier tends to classify VAD false detections as "non-keyword utterances", and serves as a secondary filter before detecting keyword events.

Acoustic "wake word" spotting typically relies on keywords that are low frequency and dissimilar from typical non-keyword utterances, the most popular wake words being words/phrases like "Alexa", "Hey Siri", and "Okay Google". In this study, monosyllabic keywords were chosen to examine what makes keywords more distinguishable neurally as opposed to acoustically. The utterances used in our experiment were exceptionally similar to each-other, varying only by 1-3 distinctive articulatory features. Indeed, a particularly important feature – keyword length – was the same across all

utterances, making the keyword detection problem significantly more difficult. The simulated keyword spotting performance for all keyword spotters in Subject 1 is shown in supplemental figures, and the simulated keyword spotting summary performance metrics are shown for all subjects in Figure 6.5. While this performance is not comparable with the current state of the art in acoustic keyword spotting, neural VAD alone appears to provide a temporally precise 1-bit (silent vs speech) BCI and the addition of keyword discrimination would allow the user to trigger the BCI while not restricting speech between intended triggerings.

The most striking finding from this study was that vowel height was poorly represented in vSMC. This is consistent with the findings of Bouchard et al., 2013 in which syllable discrimination using a high-density grid in vSMC achieved lower cluster separability of vowel height than manner of consonant articulation. This result also corroborates a finding from Ramsey et al., 2017 that vowels are the least distinguishable phonemes in their test set; the authors speculated that lacking plosives, vowels differ only in lip positions, which may not be well represented in this area. Our findings suggest that vowel height is well represented in auditory association cortex areas STG, presumably due to self-monitoring, shown in Figure 6.7.

None of the subjects in the study exhibited high gamma activity that significantly encoded vowel height within vSMC. Many studies indicate vowel phones may be decoded from vSMC (Ramsey et al., 2017; Mugler et al., 2014; Bouchard and Chang, 2014; Pei et al., 2011a), although some studies also note that decoding accuracy is generally worse than consonant phones (Ramsey

et al., 2017; Mugler et al., 2014). None of the aforementioned studies report a failure to decode vowel phones from vSMC, which is contrary to our findings. This may be due to the fact that the vowels chosen for this study, /a/ and /i/, result from a slight variation in tongue height and do not involve differential activation of the lips, such as with the vowel contrasts selected for the aforementioned studies, /a/ and /u/, which can recruit sensorimotor areas related to the face. This said, STG has been shown to consistently modulate with differences in vowel height (Mesgarani et al., 2014) during audition and self monitoring. Practically, our results suggest that discrimination of vowels during keyword spotting with a neural interface may be improved by including auditory representations from STG with sensorimotor representations from vSMC. This finding also suggests that modulation and control of vowel height relies on interactions between auditory areas and motor areas more than consonant articulation which seems to be well represented in just suprasylvian cortex.

The subject with electrode coverage most analogous to the implant detailed in Bouchard et al., 2013 had a high-density grid with 2-mm electrode diameter and 5-mm interelectrode distance over somatosensory cortex. Although we showed no significant neural differences between low and high vowel height with this grid placement, the grid in Bouchard et. al. had a slightly smaller pitch and this higher resolution may have captured more information about vowel height than we observed. Similarly, we showed significantly worse performance with lower density coverage of vSMC, demonstrated by subjects with only standard-density (2 mm electrode diameter and 1 cm pitch)

coverage, indicating that standard ECoG arrays are likely insufficient for a comprehensive speech neuroprosthesis. Some subjects were also implanted with microelectrode array grommets (75 μm electrode diameter and 1 mm pitch); these arrays have a sensor density similar to what is thought to be the spatial limit of subdural neural recordings (Slutzky et al., 2010). Micro-ECoG was useful in discriminating place of articulation for utterances from Subject 2, but its utility was greatly dependent on placement due to its limited spatial extent. An ideal ECoG array would probably cover all of vSMC with the same 1 mm pitch, but this is not yet technically feasible with clinically approved ECoG electrodes and their connectors. Although our best results came from a subject with a high density grid over vSMC, our inability to observe neural activity associated with velar syllables indicates that even these high density arrays do not capture sufficient detail to distinguish all articulators (and hence, all phones) necessary for a speech neuroprosthesis. Further research into recording devices that cover a similar spatial extent but with higher sensor density and channel counts might be fruitful, but our results indicate that neural features recorded from high density ECoG arrays can, at a minimum, produce a usable neural interface for whole-word keyword spotting in overt speech.

Correlating spatiotemporal templates with streaming high gamma features was primarily motivated by existing keyword search methodology, as well as a recent study by Ramsey et al., 2017. The temporal encoding of consonant voicing in Subject 1 (see Figure 6.6) further motivated the application of spatiotemporal template methodology. To evaluate the contribution of neural

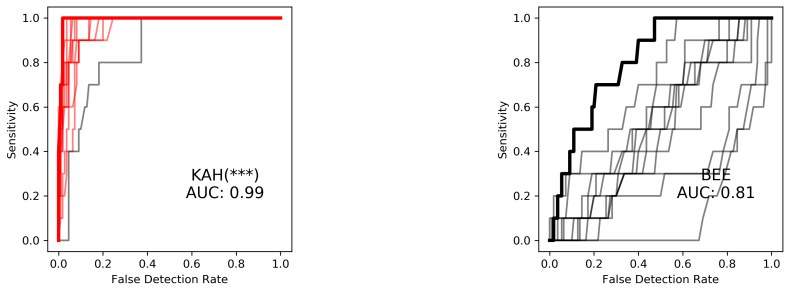

Figure 6.8: Decoupled Performance Plots. Keyword discrimination ROC curves for Subject 1 before (to the left) and after (to the right) replacement of neural templates with rectangular smoothing windows. Keyword discrimination performance dropped across all models suggesting inclusion of spatiotemporal relationships using neural templates aids keyword discrimination.

templates to keyword discrimination, the templates were replaced with a rectangular window of the same size, resulting in smoothing of the high gamma features on the same order as that of the templates. After making this change, we observed a marked drop in keyword discrimination, highlighted in Figure 6.8, suggesting that temporal relationships between high-gamma events provide information useful for discriminating keywords, and that these templates are an effective way of quantifying these relationships in single trials.

6.5 Conclusions

This study suggests that a high-sensitivity/specificity one-bit neural keyword spotting BCI can be created using ECoG recordings from vSMC and STG. Neural signals capturing speech motor representations from vSMC appear to be useful for low-latency (~1 second) and high-specificity VAD, while a

combination of neural signals from vSMC and auditory representations from STG may be useful for discriminating keyword utterances from non-keyword speech. Spatiotemporal relationships of high gamma activity across electrodes, captured and efficiently quantified using a method of neural template correlation, appear to be instrumental for keyword discrimination. In this study, keyword-spotting performance depended on several factors including electrode density and the number of electrodes within vSMC and STG. Our results suggest that high-density ECoG grids may be necessary and sufficient for capturing the spatial layout of cortical speech representations needed for a keyword-spotting neural interface. Neural features that provide information about consonant articulation appear to be best represented in vSMC, with place of articulation primarily encoded by the spatial location of high-gamma activity and consonant voicing encoded by the temporal dynamics of this activity. Vowel height during overt speech appeared to be poorly encoded by vSMC, but better represented in traditionally auditory areas along STG during self-monitoring. Although we did not test whether neural activity in STG during covert speech was sufficient for decoding vowel height, other studies have indicated that this may be possible (Pei et al., 2011b; Leuthardt et al., 2012). Together with these and other studies, our findings support the feasibility of keyword spotting with an ECoG BCI provided that relevant cortical areas are recorded with sufficient spatial sampling and that keywords are composed of neurally discriminable articulatory gestures.

Data Availability Statement

The datasets generated and analyzed for this study, analysis code, and latex source files for this paper can be found in the gigantum project located at gigantum.com/griffinmilsap/ecog-keyword-spotting.

References

Bouchard, Kristofer E., Nima Mesgarani, Keith Johnson, and Edward F. Chang (2013). "Functional organization of human sensorimotor cortex for speech articulation". In: *Nature* 495.7441, pp. 327–332. ISSN: 1476-4687. DOI: 10.1038/nature11911. URL: https://www.nature.com/articles/nature11911 (visited on 07/31/2018).

Mugler, Emily M., James L. Patton, Robert D. Flint, Zachary A. Wright, Stephan U. Schuele, Joshua Rosenow, Jerry J. Shih, Dean J. Krusienski, and Marc W. Slutzky (2014). "Direct classification of all American English phonemes using signals from functional speech motor cortex". In: *Journal of Neural Engineering* 11.3, p. 035015. ISSN: 1741-2552. DOI: 10.1088/1741-2560/11/3/035015. URL: http://stacks.iop.org/1741-2552/11/i=3/a=035015 (visited on 07/31/2018).

Mugler, E. M., M. Goldrick, J. M. Rosenow, M. C. Tate, and M. W. Slutzky (2015). "Decoding of articulatory gestures during word production using speech motor and premotor cortical activity". In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5339–5342. DOI: 10.1109/EMBC.2015.7319597.

Kanas, V. G., I. Mporas, H. L. Benz, K. N. Sgarbas, A. Bezerianos, and N. E. Crone (2014). "Real-time voice activity detection for ECoG-based speech brain machine interfaces". In: *2014 19th International Conference on Digital Signal Processing*, pp. 862–865. DOI: 10.1109/ICDSP.2014.6900790.

Herff, Christian, Dominic Heger, Adriana de Pesters, Dominic Telaar, Peter Brunner, Gerwin Schalk, and Tanja Schultz (2015). "Brain-to-text: decoding spoken phrases from phone representations in the brain". In: *Frontiers in Neuroscience* 9. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00217. URL: https://www.frontiersin.org/articles/10.3389/fnins.2015.00217/full (visited on 07/31/2018).

Bouchard, K. E. and E. F. Chang (2014). "Neural decoding of spoken vowels from human sensory-motor cortex with high-density electrocorticography". In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6782–6785. DOI: 10.1109/EMBC.2014.6945185.

Pei, Xiaomei, Dennis L. Barbour, Eric C. Leuthardt, and Gerwin Schalk (2011a). "Decoding vowels and consonants in spoken and imagined words using electrocorticographic signals in humans". In: *Journal of Neural Engineering* 8.4, p. 046028. ISSN: 1741-2552. DOI: 10.1088/1741-2560/8/4/046028. URL: http://stacks.iop.org/1741-2552/8/i=4/a=046028 (visited on 07/31/2018).

Lotte, Fabien, Jonathan S. Brumberg, Peter Brunner, Aysegul Gunduz, Anthony L. Ritaccio, Cuntai Guan, and Gerwin Schalk (2015). "Electrocorticographic representations of segmental features in continuous speech". In: *Frontiers in Human Neuroscience* 9. ISSN: 1662-5161. DOI: 10.3389/fnhum.2015.00097. URL: https://www.frontiersin.org/articles/10.3389/fnhum.2015.00097/full (visited on 08/02/2018).

Rahim, M. G., Chin-Hui Lee, and Biing-Hwang Juang (1997). "Discriminative utterance verification for connected digits recognition". In: *IEEE Transactions on Speech and Audio Processing* 5.3, pp. 266–277. ISSN: 1063-6676. DOI: 10.1109/89.568733.

Rohlicek, J. R., W. Russell, S. Roukos, and H. Gish (1989). "Continuous hidden Markov modeling for speaker-independent word spotting". In: *International Conference on Acoustics, Speech, and Signal Processing*, 627–630 vol.1. DOI: 10.1109/ICASSP.1989.266505.

Manos, A. S. and V. W. Zue (1997). "A segment-based wordspotter using phonetic filler models". In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2, 899–902 vol.2. DOI: 10.1109/ICASSP.1997.596081.

Bourlard, H., B. D'hoore, and J. Boite (1994). "Optimizing recognition and rejection performance in wordspotting systems". In: *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. i, I/373–I/376 vol.1. DOI: 10.1109/ICASSP.1994.389278.

Rohlicek, J. R., P. Jeanrenaud, K. Ng, H. Gish, B. Musicus, and M. Siu (1993). "Phonetic training and language modeling for word spotting". In: *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2, 459–462 vol.2. DOI: 10.1109/ICASSP.1993.319340.

Keshet, Joseph, David Grangier, and Samy Bengio (2009). "Discriminative keyword spotting". In: *Speech Communication* 51.4, pp. 317–329. ISSN: 0167-6393. DOI: 10.1016/j.specom.2008.10.002. URL: http://www.sciencedirect.com/science/article/pii/S0167639308001489 (visited on 07/31/2018).

Ramsey, N. F., E. Salari, E. J. Aarnoutse, M. J. Vansteensel, M. G. Bleichner, and Z. V. Freudenburg (2017). "Decoding spoken phonemes from sensorimotor cortex with high-density ECoG grids". In: *NeuroImage*. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2017.10.011. URL: http://www.sciencedirect.com/science/article/pii/S1053811917308248 (visited on 07/31/2018).

Papademetris, Xenophon, Marcel P Jackowski, Nallakkandi Rajeevan, Marcello DiStasio, Hirohito Okuda, R Todd Constable, and Lawrence H Staib (2006). "BioImage Suite: An integrated medical image analysis suite: An update". In: *The insight journal* 2006, p. 209.

Mellinger, Jürgen and Gerwin Schalk (2007). "BCI2000: a general-purpose software platform for BCI research". In: *Towards brain-computer interfacing*.

Jiang, W., T. Pailla, B. Dichter, E. F. Chang, and V. Gilja (2016). "Decoding speech using the timing of neural signal modulation". In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1532–1535. DOI: 10.1109/EMBC.2016.7591002.

Ray, Supratim, Nathan E. Crone, Ernst Niebur, Piotr J. Franaszczuk, and Steven S. Hsiao (2008). "Neural Correlates of High-Gamma Oscillations (60–200 Hz) in Macaque Local Field Potentials and Their Potential Implications in Electrocorticography". In: *Journal of Neuroscience* 28.45, pp. 11526–11536. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.2848-08.2008. URL: http://www.jneurosci.org/content/28/45/11526 (visited on 08/15/2018).

Motlicek, P., F. Valente, and I. Szoke (2012). "Improving acoustic based keyword spotting using LVCSR lattices". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4413–4416. DOI: 10.1109/ICASSP.2012.6288898.

Mesgarani, Nima, Connie Cheung, Keith Johnson, and Edward F. Chang (2014). "Phonetic Feature Encoding in Human Superior Temporal Gyrus". In: *Science* 343.6174, pp. 1006–1010. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1245994. URL: http://science.sciencemag.org/content/343/6174/1006 (visited on 08/02/2018).

Slutzky, Marc W., Luke R. Jordan, Todd Krieg, Ming Chen, David J. Mogul, and Lee E. Miller (2010). "Optimal spacing of surface electrode arrays for brain–machine interface applications". In: *Journal of Neural Engineering*

7.2, p. 026004. ISSN: 1741-2552. DOI: 10.1088/1741-2560/7/2/026004. URL: http://stacks.iop.org/1741-2552/7/i=2/a=026004 (visited on 08/02/2018).

Pei, Xiaomei, Eric C. Leuthardt, Charles M. Gaona, Peter Brunner, Jonathan R. Wolpaw, and Gerwin Schalk (2011b). "Spatiotemporal dynamics of electrocorticographic high gamma activity during overt and covert word repetition". In: *NeuroImage* 54.4, pp. 2960–2972. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2010.10.029. URL: http://www.sciencedirect.com/science/article/pii/S1053811910013212.

Leuthardt, Eric, Xiao-mei Pei, Jonathan Breshears, Charles Gaona, Mohit Sharma, Zachary Freudenburg, Dennis Barbour, and Gerwin Schalk (2012). "Temporal evolution of gamma activity in human cortex during an overt and covert word repetition task". In: *Frontiers in Human Neuroscience* 6. ISSN: 1662-5161. DOI: 10.3389/fnhum.2012.00099. URL: https://www.frontiersin.org/articles/10.3389/fnhum.2012.00099/full (visited on 08/15/2018).

Chapter 7

Conclusions and Future Work

The research detailed in this dissertation described software and methodology that will make a significant contribution to the next generation of neural speech decoders. WebFM permits mapping of eloquent cortex that is easier to perform and distinguishes between cortical representations of speech using timing information. Since publication, this software has been iterated to include 3D visualization for stereotactic EEG electrodes as well as the capability to render cortico-cortical evoked potentials for mapping sub-threshold stimulation-based effective connectivity. WebFM could be the basis of a multi-site study on functional mapping of eloquent cortex using ECoG recordings; a study that could turn passive ECoG mapping into a clinically accepted standard. BCI2000Web, on the other hand, enables developers to make web-enabled BCIs, and could be the underlying framework for a brain-controlled browser extension, or a web-based neural communication platform.

Methodology detailed in this dissertation can help identify spatiotemporal signatures of high frequency neural activity for the generation of correlational matched filters. As described previously, this methodology can be applied to

identify behavioral event onset times using only neural data. Initial research suggests this method can also be used to identify neural sub-processes without overt behavioral output to synchronize to. Matched filtering methodology was used for whole-word identification in the keyword spotting study described in Chapter 6, but could easily be applied to perform phone-modeling for a phoneme-based neural ASR approach. This methodology can also be extended to the creation of subspace manifolds for the visualization of neural encoding axes using multidimensional scaling (Milsap et al., 2017); research that could prove fruitful for functional anatomic atlassing and representational similarity analyses in ECoG.

Speech BCI could eventually be a useful means of human-computer-interaction in general use. Eyetracking combined with a covert speech-based neural click could be a useful means of interacting with augmented or virtual reality situations. Neural keyword spotting interfaces could allow for covert interaction with artificial intelligence agents. Neural speech recognition may benefit from the neural articulatory representations that encode speech differently; succeeding where acoustic speech recognition fails. Furthermore, if semantic representations of speech are more thoroughly understood, we may be able to use a neural semantic decoder for direct communication of high level thoughts without need to break thoughts down into a syntax and language for communication; drastically increasing the bandwidth of human communication.

Speech BCI will only make this level of impact after the development of

a new modality for neural recording; one that is capable of recording non-invasive neural signals comparable to ECoG. Many would argue that it is ethically dubious to perform risky brain surgery on healthy subjects for the augmentation of a speech neuroprosthesis. A headband or minimally invasive dermal implant that enables a speech BCI capable of even half the aforementioned functionality would likely see widespread adoption amongst healthy users. The future of human-computer-interaction may have a neural component, but not before significant advancement in neural recording modalities, neural signal processing methodology, and our understanding of the cortical representations of speech.

References

Milsap, G. W., M. J. Collard, K. Rupp, M. J. Roos, C. Caceres, C. Ratto, M. Wolmetz, and N. E. Crone (2017). "Intrinsic neural spaces from human electrocorticography". In: *Society for Neuroscience Abstract*, 589.12/GG7.

Appendices

Appendix A

Supplemental Figures

A.1 Supplemental Figures: Mapping of Visual Semantic Attributes to Spatiotemporal Features of Neural Recordings

This appendix contains supplemental figures for Chapter 5.

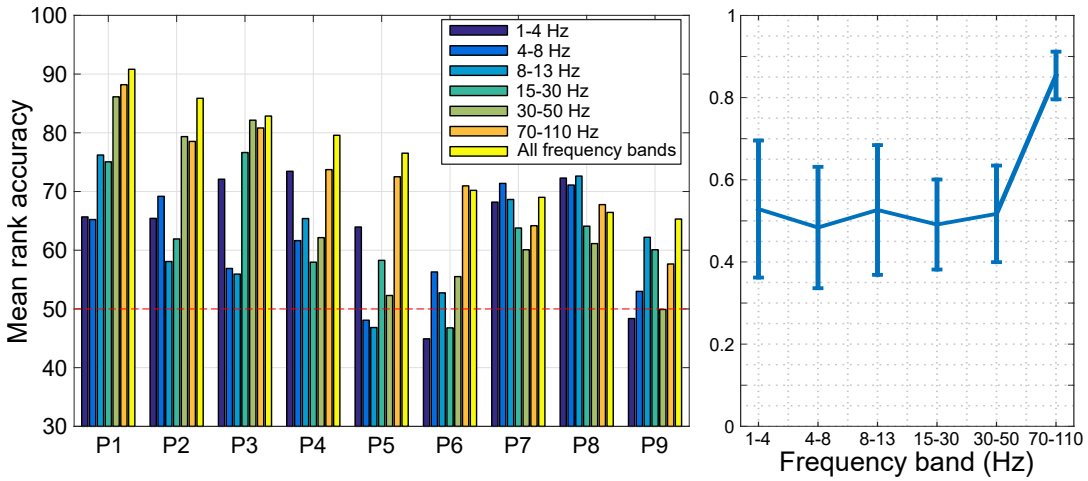

Figure A.1: Comparison of individual frequency bands. (A) Mean rank accuracy (MRA) calculated for each patient and each individual frequency band. All models used neural data averaged across all 6 presentations of an object. The dashed line represents chance performance. Higher frequency bands perform substantially better than low frequency bands. (B) The ratio of single-frequency-band MRA to full-model (all frequencies) MRA, averaged across all patients. Chance accuracy of 50% was subtracted from both MRAs before the ratio was calculated. Standard error bars are shown. The five lowest frequency bands perform about half as well as the full model, while high-gamma, 70-110 Hz, performs substantially better (about 85% of the full model performance)

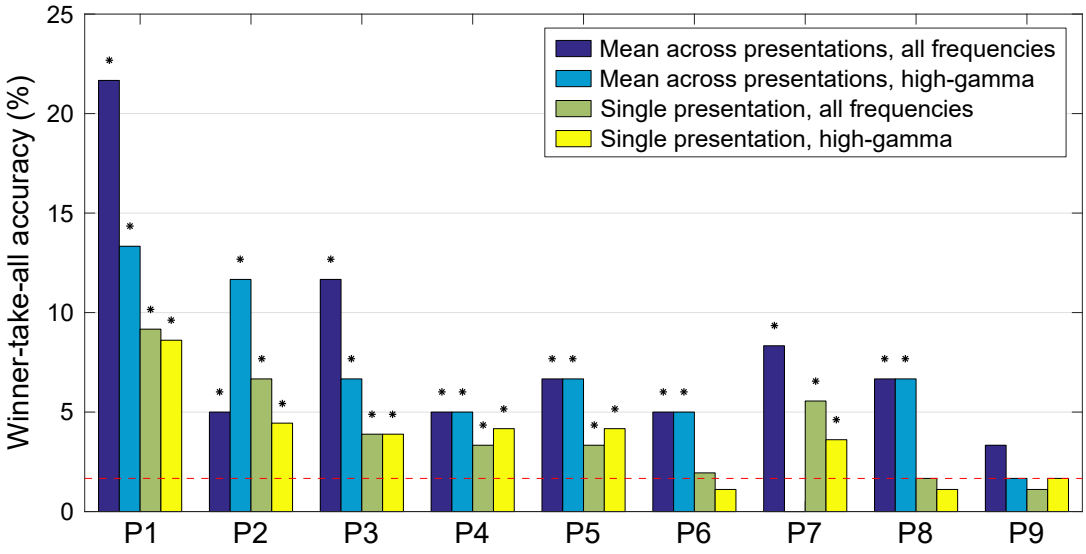

Figure A.2: Winner-take-all accuracy. This metric is calculated as the fraction of trials where the target object was ranked first out of the 60 possible objects. Chance accuracy is shown with a dashed line at 1.7%. Notably, when averaging across presentations and including all frequency bands, P1 had a success rate of 22%. In other words, the target object was ranked first in 13 out of 60 trials. Permutation tests, using shuffling of the noun labels in the semantic attribute matrix, were used to generate the significance results. FDR correction ($\alpha = .05$) was applied across all 9 patients and 4 decoding conditions.

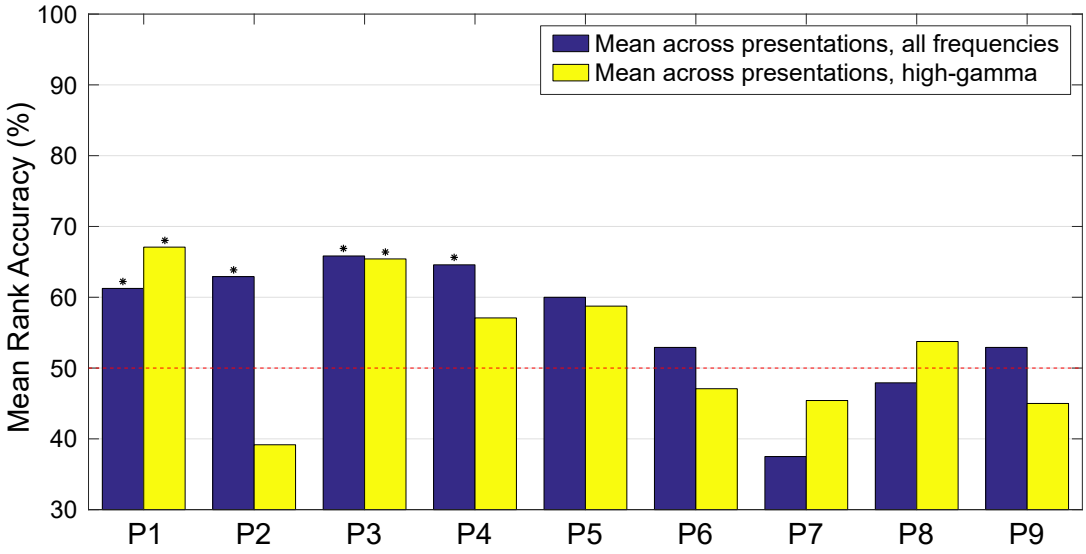

Figure A.3: Within-category MRA For the top 4 patients, models are able to discriminate between objects of the same category above chance levels. FDR-correction ($\alpha = .05$) was applied across all 9 patients and 2 decoding conditions. The dashed line represents chance performance.

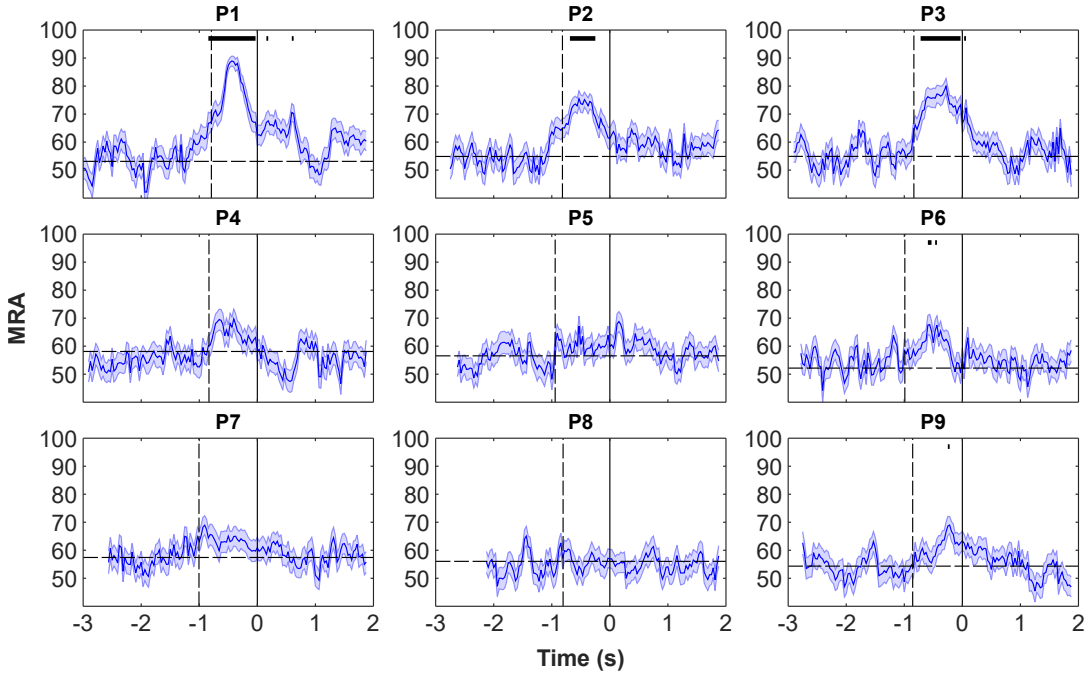

Figure A.4: Sliding window decoding analysis on speech-aligned data. The solid vertical line represents speech onset, and the dashed vertical line represents the median stimulus onset relative to the spoken response. Dashed lines represent patient-specific chance performance, calculated as the mean MRA during the baseline period. Notably, 6 subjects had decoding results that exceeded baseline, and the onset of significant decoding occurred before stimulus onset for 5 of these subjects. Significance was determined via rank-sum tests at each individual time point, Bonferroni-corrected over all time points for each individual patient. Baselines for each patient were identical to those used in the stimulus-aligned analysis (Figure 5.3).

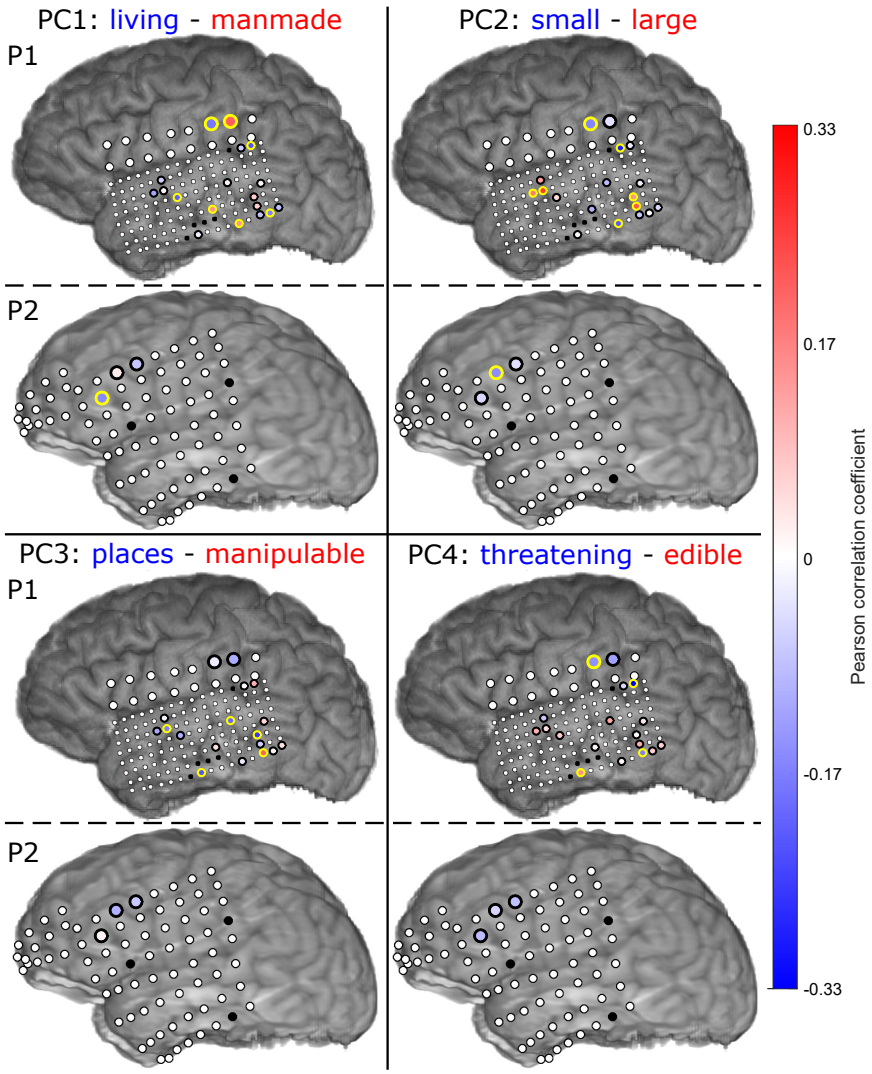

Figure A.5: Correlation of semantic PCs with high-gamma for significant lateral electrodes. Analysis was restricted to those electrodes that were well-predicted by the encoding model (Figure 5.4) and are represented with thick black or yellow rings. The color inside the rings represents the signed correlation values, with yellow rings denoting statistical significance with FDR-correction across all 21 electrodes and four PCs ($\alpha = .05$). In P1, the anterior superior and posterior middle temporal gyri both have pairs of electrodes where high-gamma correlated positively with the size dimension, representing nouns that have large canonical size.

A.2 Supplemental Figures: Discrimination of Utterances using Spatiotemporal Matched Filter Templates in Single-trials

This appendix contains supplemental figures for Chapter 6.

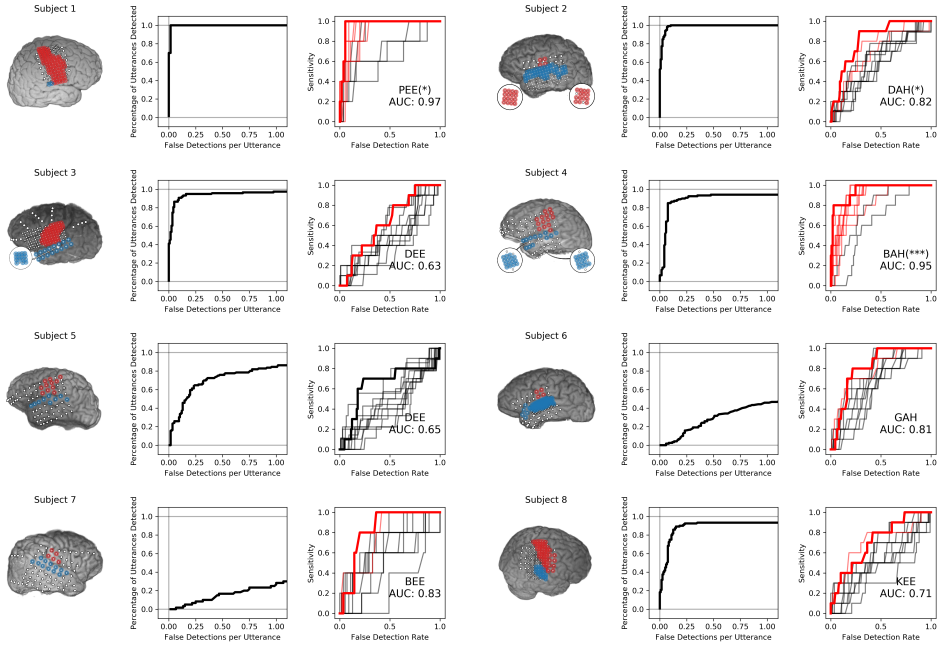

Figure A.6: Isolated VAD and keyword ROCs for all subjects. The left-most panel shows the electrodes highlighted in red and blue that were used to discriminate syllables. Only electrodes highlighted in red were used to perform VAD. The center panel shows the VAD performance in sensitivity (percentage of utterance timings correctly identified) against the number false detections per utterance for various VAD thresholds. The right-most panel shows ROC curves for all twelve keyword detectors. ROC curves with AUC values were significant at the 95% confidence interval are highlighted in red. The keyword detector that produced the highest AUC is highlighted in bold-red and indicated via annotation under the curves, followed by asterisks indicating significance at the $p < 0.05$ (*), $p < 0.01$ (**) and the $p < 0.001$ (***) level with respect to the distribution of maximum AUC models.

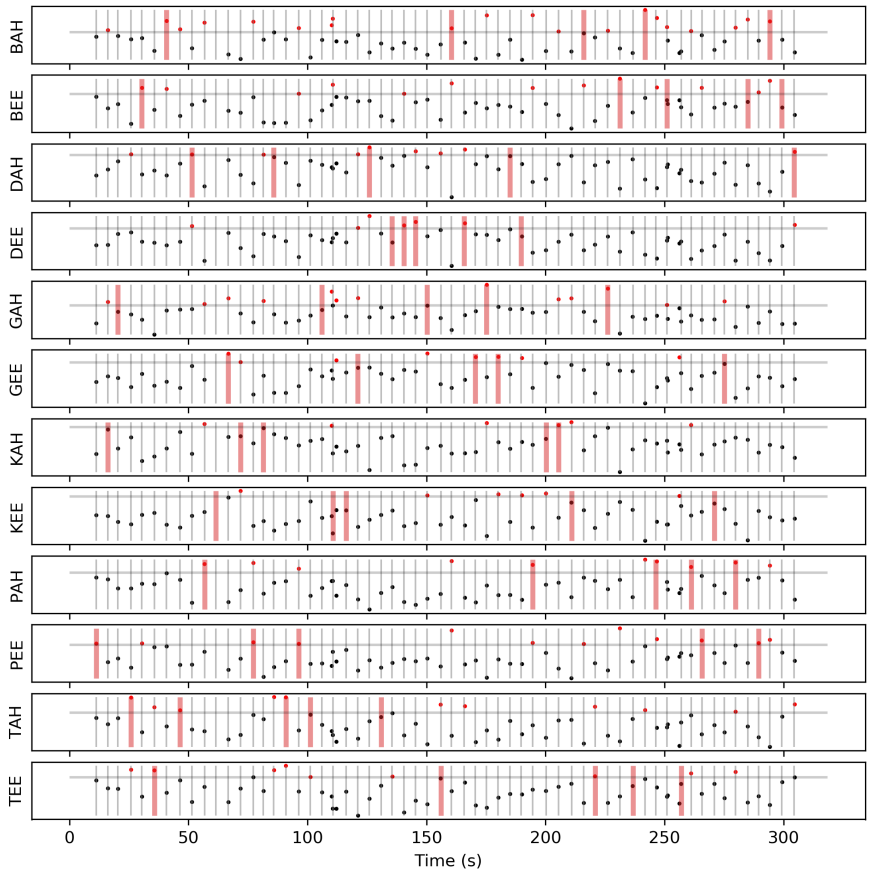

Figure A.7: Simulated real-time performance for Subject 1 across the repetition task. Performance for models trained to identify one keyword vs non-keyword speech is visualized for every one of the twelve keyword models. Ground truth utterances are denoted by vertical lines, and utterances that the model should identify as "keyword" utterances are highlighted in bold red. Times during which the VAD model identified utterances are indicated with black dots, the Y-value of these dots indicates a unit-less classifier output from the features at the time of the detection; higher placement indicates a more "keyword-like" utterance. Each keyword spotter has had a threshold of discrimination set at 0.75 the equal error rate, so as to promote higher sensitivity at the expense of more false-detections. Super-threshold classifier output is highlighted with a red dot rather than a black one, indicating the detection of a keyword.

Appendix B

Software Listings

B.1 bci2k.js

```
// ========================================================================== //
//
// bci2k.js
// A javascript connector for BCI2000
//
// ========================================================================== //


// REQUIRES
import $ from 'jquery'

// Needed to allow operation in Node outside of a browser
var WebSocket = WebSocket || require( 'websocket' ).w3cwebsocket;


export class BCI2K_Connection{
    constructor(){
        this.onconnect = function( event ) {};
        this.ondisconnect = function( event ) {};

        this._socket = null;
        this._execid = 0;
        this._exec = {}
    };


    connect( address ) {

        var connection = this;

        return new Promise( function( resolve, reject ) {

            if ( address === undefined )
                // TODO Browser-dependent
                address = window.location.host;
```

```
            connection.address = address;

            connection._socket = new WebSocket( 'ws://' + connection.address );

            connection._socket.onerror = function( error ) {
                // This will only execute if we err before connecting, since
                // Promises can only get triggered once
                reject( 'Error connecting to BCI2000 at ' + connection.address );
            }

            connection._socket.onopen = function( event ) {
                connection.onconnect( event );
                resolve( event );
            }

            connection._socket.onclose = function( event ) {
                connection.ondisconnect( event );
            }

            connection._socket.onmessage = function( event ) {
                connection._handleMessageEvent( event );
            }

        } );

    };

    _handleMessageEvent( event ) {
        var arr = event.data.split( ' ' );

        var opcode = arr[0];
        var id = arr[1];
        var msg = arr.slice( 2 ).join(' ');

        switch( opcode ) {
            case 'S': // START: Starting to execute command
                if( this._exec[ id ].onstart )
                    this._exec[ id ].onstart( this._exec[ id ] );
                break;
            case 'O': // OUTPUT: Received output from command
                this._exec[ id ].output += msg + ' \n';
                if( this._exec[ id ].onoutput )
                    this._exec[ id ].onoutput( this._exec[ id ] );
                break;
            case 'D': // DONE: Done executing command
                this._exec[ id ].exitcode = parseInt( msg );
                if( this._exec[ id ].ondone )
                    this._exec[ id ].ondone( this._exec[ id ] );
                delete this._exec[ id ];
                break;
            default:
                break;
        }
    };

    tap( location, onSuccess, onFailure ) {

        var connection = this;
```

```
94
95          var locationParameter = "WS" + location + "Server";
96
97          return this.execute( 'Get Parameter ' + locationParameter )
98                      .then( function( location ) {
99
100                         if ( location.indexOf( 'does not exist' ) >= 0 ) {
101                             return Promise.reject( 'Location parameter does not
    exist' );
102                         }
103
104                         if ( location === '' ) {
105                             return Promise.reject( 'Location parameter not set' );
106                         }
107
108                         var dataConnection = new BCI2K_DataConnection();
109
110                         // TODO We used to "resolve" here, before doiing the
111                         // actual connecting bit, but I think it makes much
112                         // more sense to have tap "success" be actually
113                         // connecting to the source, rather than just getting
114                         // a sensical address
115
116                         // Use our address plus the port from the result
117                         return dataConnection.connect( connection.address + ':' +
    location.split( ':' )[1] )
118                                              .then( function( event ) {
119                                                  // To keep with our old API,
    we actually want to wrap the
120                                                  // dataConnection, and not the
     connection event
121                                                  // TODO This means we can't
    get the connection event!
122                                                  return dataConnection;
123                                              } );
124
125                 } );
126
127     };
128
129     connected() {
130         return ( this._socket !== null && this._socket.readyState === WebSocket.
    OPEN );
131     };
132
133     execute( instruction, ondone, onstart, onoutput ) {
134
135         var connection = this;
136
137         if ( this.connected() ) {
138
139             return new Promise( function( resolve, reject ) {
140
141                 var id = ( ++( connection._execid ) ).toString();
142
143                 // TODO Properly handle errors from BCI2000
144                 connection._exec[ id ] = {
145                     onstart: onstart,
```

```
                    onoutput: onoutput,
                    ondone: function( exec ) {
                        if ( ondone ) {
                            ondone( exec );
                        }
                        resolve( exec.output );     // TODO Should pass whole thing?
                    },
                    output: '',
                    exitcode: null
                };

                var msg = 'E ' + id + ' ' + instruction;
                connection._socket.send( msg );

            } );

        }

        // Cannot execute if not connected
        return Promise.reject( 'Cannot execute instruction: not connected to BCI2000' );

    };

    getVersion( fn ) {
        this.execute( "Version", function( exec ) {
            fn( exec.output.split(' ')[1] );
        } );
    };

    showWindow() {
        return this.execute( "Show Window" );
    };

    hideWindow() {
        return this.execute( "Hide Window" );
    };

    setWatch(state, ip, port) {
        return this.execute( "Add watch " + state + " at " + ip + ":" + port );
    };

    resetSystem() {
        return this.execute( "Reset System" );
    };

    // TODO Is argument necessary now with Promise API?
    setConfig( fn ) {
        return this.execute( "Set Config", fn );
    };

    start() {
        return this.execute( "Start" );
    };

    stop() {
        return this.execute( "Stop" );
```

```
202     };
203
204     kill() {
205         return this.execute( "Exit" );
206     };
207 }
208
209
210 export class BCI2K_DataConnection{
211     constructor(){
212     this._socket = null;
213
214     this.onconnect = function( event ) {};
215     this.onGenericSignal = function( data ) {};
216     this.onStateVector = function( data ) {};
217     this.onSignalProperties = function( data ) {};
218     this.onStateFormat = function( data ) {};
219     this.ondisconnect = function( event ) {};
220
221     this.signalProperties = null;
222     this.stateFormat = null;
223     this.stateVecOrder = null;
224     }
225
226
227     connect( address ) {
228
229         var connection = this;
230
231         return new Promise( function( resolve, reject ) {
232
233             connection._socket = new WebSocket( "ws://" + address );
234
235             connection._socket.onerror = function( event ) {
236                 // This will only execute if we err before connecting, since
237                 // Promises can only get triggered once
238                 reject( 'Error connecting to data source at ' + connection.address
     );
239             };
240
241             connection._socket.onopen = function( event ) {
242                 connection.onconnect( event );
243                 resolve( event );
244             };
245
246             connection._socket.onclose = function( event ) {
247                 connection.ondisconnect( event );
248             };
249
250             connection._socket.onmessage = function( event ) {
251                 connection._handleMessageEvent( event );
252             };
253
254         } );
255
256     };
257
258     _handleMessageEvent( event ) {
```

```
259
260         var connection = this;
261
262         var messageInterpreter = new FileReader();
263         messageInterpreter.onload = function( e ) {
264             connection._decodeMessage( e.target.result );
265         };
266         messageInterpreter.readAsArrayBuffer( event.data );
267
268     };
269
270     connected() {
271         return ( this._socket != null && this._socket.readyState === WebSocket.
    OPEN );
272     };
273
274     SignalType: {
275         INT16 : 0,
276         FLOAT24 : 1,
277         FLOAT32 : 2,
278         INT32 : 3
279     }
280
281     _decodeMessage( data ) {
282
283         // var dv = new BCI2K_DataView( data, 0, data.byteLength, true );
284         var dv = new DataView(data,0,data.byteLength,true)
285         dv.getNullTermString = function() {
286             var val = "";
287             while ( this._offset < this.byteLength ) {
288                 var v = this.getUint8();
289                 if( v === 0 ) break;
290                 val += String.fromCharCode( v );
291             }
292             return val;
293         };
294         var descriptor = dv.getUint8();
295
296         switch ( descriptor ) {
297
298             case 3:
299                 this._decodeStateFormat( dv ); break;
300
301             case 4:
302                 var supplement = dv.getUint8();
303
304                 switch ( supplement ) {
305                     case 1:
306                         this._decodeGenericSignal( dv ); break;
307                     case 3:
308                         this._decodeSignalProperties( dv ); break;
309                     default:
310                         console.error( "Unsupported Supplement: " + supplement.
    toString() );
311                         break;
312                 } break;
313
314             case 5:
```

```
                this._decodeStateVector( dv ); break;

            default:
                console.error( "Unsupported Descriptor: " + descriptor.toString()
); break;

        }

    };

    _decodePhysicalUnits( unitstr ) {
        var units = {};
        var unit = unitstr.split( ' ' );
        var idx = 0;
        units.offset = Number( unit[ idx++ ] );
        units.gain = Number( unit[ idx++ ] );
        units.symbol = unit[ idx++ ];
        units.vmin = Number( unit[ idx++ ] );
        units.vmax = Number( unit[ idx++ ] );
        return units;
    };

    _decodeSignalProperties( dv ) {
        var propstr = dv.getNullTermString();

        // Bugfix: There seems to not always be spaces after '{' characters
        propstr = propstr.replace( /{/g, ' { ' );
        propstr = propstr.replace( /}/g, ' } ' );

        this.signalProperties = {};
        var prop_tokens = propstr.split( ' ' );
        var props = [];
        for( var i = 0; i < prop_tokens.length; i++ ) {
            if( $.trim( prop_tokens[i] ) === "" ) continue;
            props.push( prop_tokens[i] );
        }

        var pidx = 0;
        this.signalProperties.name = props[ pidx++ ];

        this.signalProperties.channels = [];
        if( props[ pidx ] === '{' ) {
            while( props[ ++pidx ] !== '}' )
                this.signalProperties.channels.push( props[ pidx ] );
            pidx++; // }
        } else {
            let numChannels = parseInt( props[ pidx++ ] );
            for( let i = 0; i < numChannels; i++ )
                this.signalProperties.channels.push( ( i + 1 ).toString() );
        }

        this.signalProperties.elements = [];
        if( props[ pidx ] === '{' ) {
            while( props[ ++pidx ] !== '}' )
                this.signalProperties.elements.push( props[ pidx ] );
            pidx++; // }
        } else {
            let numElements = parseInt( props[ pidx++ ] );
```

```
            for( let i = 0; i < numElements; i++ )
                this.signalProperties.elements.push( ( i + 1 ).toString() );
        }

        // Backward Compatibility
        this.signalProperties.numelements = this.signalProperties.elements.length;
        this.signalProperties.signaltype = props[ pidx++ ];
        this.signalProperties.channelunit = this._decodePhysicalUnits(
            props.slice( pidx, pidx += 5 ).join( ' ' )
        );

        this.signalProperties.elementunit = this._decodePhysicalUnits(
            props.slice( pidx, pidx += 5 ).join( ' ' )
        );

        pidx++; // '{'

        this.signalProperties.valueunits = []
        for( let i = 0; i < this.signalProperties.channels.length; i++ )
            this.signalProperties.valueunits.push(
                this._decodePhysicalUnits(
                    props.slice( pidx, pidx += 5 ).join( ' ' )
                )
            );

        pidx++; // '}'

        this.onSignalProperties( this.signalProperties );
    };

    _decodeStateFormat( dv ) {
        this.stateFormat = {};
        let formatStr = dv.getNullTermString();

        let lines = formatStr.split( '\n' );
        for( let lineIdx = 0; lineIdx < lines.length; lineIdx++ ){
            if( $.trim( lines[ lineIdx ] ).length === 0 ) continue;
            let stateline = lines[ lineIdx ].split( ' ' );
            let name = stateline[0];
            this.stateFormat[ name ] = {};
            this.stateFormat[ name ].bitWidth = parseInt( stateline[1] );
            this.stateFormat[ name ].defaultValue = parseInt( stateline[2] );
            this.stateFormat[ name ].byteLocation = parseInt( stateline[3] );
            this.stateFormat[ name ].bitLocation = parseInt( stateline[4] );
        }

        let vecOrder = []
        for( let state in this.stateFormat ) {
            let loc = this.stateFormat[ state ].byteLocation * 8;
            loc += this.stateFormat[ state ].bitLocation
            vecOrder.push( [ state, loc ] );
        }

        // Sort by bit location
        vecOrder.sort( function( a, b ) {
            return a[1] < b[1] ? -1 : ( a[1] > b[1] ? 1 : 0 );
        } );

```

```
        // Create a list of ( state, bitwidth ) for decoding state vectors
        this.stateVecOrder = [];
        for( let i = 0; i < vecOrder.length; i++ ) {
            let state = vecOrder[i][0]
            this.stateVecOrder.push( [ state, this.stateFormat[ state ].bitWidth ]
     );
        }

        this.onStateFormat( this.stateFormat );
    };

    _decodeGenericSignal( dv ) {

        let signalType = dv.getUint8();
        let nChannels = dv.getLengthField( 2 );
        let nElements = dv.getLengthField( 2 );

        let signal = [];
        for( let ch = 0; ch < nChannels; ++ch ) {
            signal.push( [] );
            for( let el = 0; el < nElements; ++el ) {
                switch( signalType ) {

                    case this.SignalType.INT16:
                        signal[ ch ].push( dv.getInt16() );
                        break;

                    case this.SignalType.FLOAT32:
                        signal[ ch ].push( dv.getFloat32() );
                        break;

                    case this.SignalType.INT32:
                        signal[ ch ].push( dv.getInt32() );
                        break;

                    case this.SignalType.FLOAT24:
                        // TODO: Currently Unsupported
                        signal[ ch ].push( 0.0 );
                        break;
                    default:
                        break;
                }
            }
        }

        this.onGenericSignal( signal );
    };

    _decodeStateVector( dv ) {
        if( this.stateVecOrder == null ) return;

        // Currently, states are maximum 32 bit unsigned integers
        // BitLocation 0 refers to the least significant bit of a byte in the
    packet
        // ByteLocation 0 refers to the first byte in the sequence.
        // Bits must be populated in increasing significance

        var stateVectorLength = parseInt( dv.getNullTermString() );
```

```
        var numVectors = parseInt( dv.getNullTermString() );

        // var vecOff = dv.tell();

        var states = {};
        for( var state in this.stateFormat )
            states[ state ] = Array( numVectors ).fill( this.stateFormat[ state ].
    defaultValue ) ;

        for( var vecIdx = 0; vecIdx < numVectors; vecIdx++ ) {
            var vec = dv.getBytes( stateVectorLength, dv.tell(), true, false );
            var bits = [];
            for( var byteIdx = 0; byteIdx < vec.length; byteIdx++ ) {
                bits.push( ( vec[ byteIdx ] & 0x01 ) !== 0 ? 1 : 0 );
                bits.push( ( vec[ byteIdx ] & 0x02 ) !== 0 ? 1 : 0 );
                bits.push( ( vec[ byteIdx ] & 0x04 ) !== 0 ? 1 : 0 );
                bits.push( ( vec[ byteIdx ] & 0x08 ) !== 0 ? 1 : 0 );
                bits.push( ( vec[ byteIdx ] & 0x10 ) !== 0 ? 1 : 0 );
                bits.push( ( vec[ byteIdx ] & 0x20 ) !== 0 ? 1 : 0 );
                bits.push( ( vec[ byteIdx ] & 0x40 ) !== 0 ? 1 : 0 );
                bits.push( ( vec[ byteIdx ] & 0x80 ) !== 0 ? 1 : 0 );
            }

                for( var stateIdx = 0; stateIdx < this.stateVecOrder.length;
    stateIdx++ ) {
                var fmt = this.stateFormat[ this.stateVecOrder[ stateIdx ][ 0 ] ];
                var offset = fmt.byteLocation * 8 + fmt.bitLocation;
                var val = 0; var mask = 0x01;
                for( var bIdx = 0; bIdx < fmt.bitWidth; bIdx++ ) {
                    if( bits[ offset + bIdx ] ) val = ( val | mask ) >>> 0;
                    mask = ( mask << 1 ) >>> 0;
                }
                states[ this.stateVecOrder[ stateIdx ][0] ][ vecIdx ] = val;
                }
        }
        this.onStateVector( states );
    };
}
```

B.2 WSIOFilter.cpp/h

```
////////////////////////////////////////////////////////////////////////////////
// $Id: $
// Author: griffin.milsap@gmail.com
// Description: A filter that sends/receives states and signals over a
//         TCP facilitated WebSocket (RFC6455) connection. Can be instantiated
//         several times using subclasses
//
// $BEGIN_BCI2000_LICENSE$
//
// This file is part of BCI2000, a platform for real-time bio-signal research.
// [ Copyright (C) 2000-2012: BCI2000 team and many external contributors ]
//
// BCI2000 is free software: you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the Free Software
// Foundation, either version 3 of the License, or (at your option) any later
// version.
//
// BCI2000 is distributed in the hope that it will be useful, but
//                         WITHOUT ANY WARRANTY
// - without even the implied warranty of MERCHANTABILITY or FITNESS FOR
// A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along with
// this program.  If not, see <http://www.gnu.org/licenses/>.
//
// $END_BCI2000_LICENSE$
////////////////////////////////////////////////////////////////////////////////
#ifndef WSIO_FILTER_H
#define WSIO_FILTER_H

#include "GenericFilter.h"

#include "Sockets.h"
#include "Lockable.h"
#include "Thread.h"

#include <iostream>
#include <list>

class WSIOFilter : public GenericFilter, public Thread
{
 public:
          WSIOFilter( std::string section, std::string name, uint16_t default_port
     );
  virtual ~WSIOFilter();
  virtual void Preflight( const SignalProperties&, SignalProperties& ) const;
  virtual void Initialize( const SignalProperties&, const SignalProperties& );
  virtual void Halt();
  virtual void StartRun();
  virtual void StopRun();
  virtual void Process( const GenericSignal&, GenericSignal& );
  virtual bool AllowsVisualization() const { return false; }

 protected:
  int OnExecute();

```

```
 private:
  void DeleteServers();

  std::string mStateVectorFormat;
  SignalProperties mProperties;
  std::string mAddressPrm;
  ServerTCPSocket mListeningSocket;
  std::string mWebRoot;

  class Connection;
  friend class Connection;
  struct : std::list<Connection*>, Lockable<NonrecursiveSpinLock> {} mConnections;
};
#endif // WSIO_FILTER_H
```

```
////////////////////////////////////////////////////////////////////////////////
// $Id: $
// Author: griffin.milsap@gmail.com
// Description: A filter that sends/receives states and signals over a
//         TCP facilitated WebSocket (RFC6455) connection. Can be instantiated
//         several times using subclasses
//
// $BEGIN_BCI2000_LICENSE$
//
// This file is part of BCI2000, a platform for real-time bio-signal research.
// [ Copyright (C) 2000-2012: BCI2000 team and many external contributors ]
//
// BCI2000 is free software: you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the Free Software
// Foundation, either version 3 of the License, or (at your option) any later
// version.
//
// BCI2000 is distributed in the hope that it will be useful, but
//                         WITHOUT ANY WARRANTY
// - without even the implied warranty of MERCHANTABILITY or FITNESS FOR
// A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along with
// this program.  If not, see <http://www.gnu.org/licenses/>.
//
// $END_BCI2000_LICENSE$
////////////////////////////////////////////////////////////////////////////////
#include "PCHIncludes.h"
#pragma hdrstop

#include "WSIOFilter.h"
#include "HTTPInterpreter.h"
#include "WebSocketInterpreter.h"
#include "Streambuf.h"
#include "BCIException.h"
#include "FileUtils.h"

#include <string>
#include <sstream>
#include <fstream>

using namespace std;
using namespace bci;

class WSIOFilter::Connection : public Environment, private Thread, private
    WebSocketInterpreter, private HTTPInterpreter
{
 public:
  Connection( WSIOFilter* parent );

  // Thread Interface
  int OnExecute();

  // HTTPInterpreter interface
  bool OnRequest( const HTTPMessage& );

  void OnConnect();
  void WriteMessages( const GenericSignal &signal );
```

```
58    void Abort();
59
60   private:
61    ~Connection();
62    Synchronized<WSIOFilter*> mpParent;
63
64    TCPSocket mSocket;
65    BufferedIO mBuffer;
66    iostream mStream;
67
68    Lockable< Mutex > mWriteLock;
69
70    friend class HTTPInterpreter;
71    friend class WebSocketInterpreter;
72
73  };
74
75
76  WSIOFilter::WSIOFilter( string section, string name, uint16_t default_port )
77  {
78    mAddressPrm = "WS" + name + "Server";
79
80    ostringstream ss;
81    ss << section << " string " << mAddressPrm << "= % localhost:" << default_port
        << " % % ";
82    ss << "// IP address/port to host a WebSocketServer on, e.g. localhost:" <<
        default_port;
83    BEGIN_PARAMETER_DEFINITIONS
84      ss.str().c_str(),
85    END_PARAMETER_DEFINITIONS
86  }
87
88  WSIOFilter::~WSIOFilter()
89  {
90    Halt();
91  }
92
93  void
94  WSIOFilter::Preflight( const SignalProperties& inSignalProperties,
95                               SignalProperties& outSignalProperties ) const
96  {
97    string address = string( Parameter( mAddressPrm ) );
98    if( address != "" )
99    {
100     ServerTCPSocket preflightSocket;
101     preflightSocket.Open( address );
102     if( !preflightSocket.IsOpen() )
103       bcierr << "Could not start server on " << address << endl;
104     preflightSocket.Close();
105   }
106
107   // Pre-flight access each state in the list.
108   for( int state = 0; state < States->Size(); ++state )
109     State( ( *States )[ state ].Name() );
110
111   outSignalProperties = inSignalProperties;
112 }
113
```

```
114 void
115 WSIOFilter::Initialize( const SignalProperties&, const SignalProperties& Output )
116 {
117   string connectorAddress = string( Parameter( mAddressPrm ) );
118   mProperties = Output;
119   stringstream sStateFormat;
120   States->InsertInto( sStateFormat );
121   mStateVectorFormat = sStateFormat.str();
122
123   if( connectorAddress != "" )
124   {
125     mListeningSocket.SetTCPNodelay( true );
126     mListeningSocket.Open( connectorAddress );
127     if( !mListeningSocket.IsOpen() )
128       throw bciexception << "Cannot listen at " << connectorAddress;
129     Thread::Start();
130   }
131 }
132
133 void
134 WSIOFilter::Halt()
135 {
136   if( mListeningSocket.IsOpen() )
137     mListeningSocket.Close();
138   list<Connection*> c;
139   WithLock( mConnections )
140     c = mConnections;
141   for( list<Connection*>::iterator i = c.begin(); i != c.end(); ++i )
142     (*i)->Abort();
143   if( Thread::Running() )
144     Thread::TerminateAndWait();
145 }
146
147 int
148 WSIOFilter::OnExecute()
149 {
150   while( mListeningSocket.Wait() )
151     new Connection( this );
152   return 0;
153 }
154
155 void
156 WSIOFilter::StartRun()
157 {
158
159 }
160
161 void
162 WSIOFilter::StopRun()
163 {
164
165 }
166
167 void
168 WSIOFilter::Process( const GenericSignal& Input, GenericSignal& Output )
169 {
170   Output = Input;
```

```
171   for( list< Connection* >::iterator i = mConnections.begin(); i != mConnections.
        end(); ++i )
172     ( *i )->WriteMessages( Input );
173   bcidbg << "Running Process" <<endl;
174 }
175
176 WSIOFilter::Connection::Connection( WSIOFilter* pParent ) :
177   Thread( true ),
178   mpParent( pParent ),
179   mStream( &mBuffer )
180 {
181   mBuffer.SetIO( &mSocket );
182   WithLock( mpParent->mConnections )
183     mpParent->mConnections.push_back( this );
184   mpParent->mListeningSocket.WaitForAccept( mSocket, 0 );
185   Thread::Start();
186 }
187
188 WSIOFilter::Connection::~Connection()
189 {
190   WithLock( mpParent->mConnections )
191     mpParent->mConnections.remove( this );
192 }
193
194 void
195 WSIOFilter::Connection::Abort()
196 {
197   Thread::TerminateAndWait();
198 }
199
200 int
201 WSIOFilter::Connection::OnExecute()
202 {
203   HTTPInterpreter::HTTPListen( this, mSocket, mStream );
204   return 0;
205 }
206
207 bool
208 WSIOFilter::Connection::OnRequest( const HTTPInterpreter::HTTPMessage& msg )
209 {
210   if( msg.command == "GET" )
211   {
212     Header::const_iterator upgrade = msg.header.find( "upgrade" );
213     if( upgrade != msg.header.end() && (
214       upgrade->second.find( "websocket" ) != string::npos ||
215       upgrade->second.find( "WebSocket" ) != string::npos ) )
216     {
217       // Upgrade connection to websocket protocol
218       Header::const_iterator key = msg.header.find( "sec-websocket-key" );
219       if( key == msg.header.end() ) return false;
220       WebSocketInterpreter::Listen( this, mSocket, mStream, key->second );
221     } else HTTPRespond( mStream, 404 );
222     return false;
223
224   } else {
225
226     // This interface only supports GET requests for websockets
227     HTTPRespond( mStream, 500 );
```

```
228       return false;
229     }
230
231     // Should never get here, something bad happened
232     return false;
233
234
235 }
236
237 void
238 WSIOFilter::Connection::OnConnect()
239 {
240     // Serialize the StateFormat
241     stringstream sStateFormat;
242     sStateFormat << uint8_t( 0x03 ) << mpParent->mStateVectorFormat;
243     WriteMessage( mStream, Opcode::Binary, sStateFormat.str() );
244
245     // Serialize the SignalProperties
246     stringstream sSignalProperties;
247     sSignalProperties << uint8_t( 0x04 ) << uint8_t( 0x03 );
248     mpParent->mProperties.InsertInto( sSignalProperties );
249     WriteMessage( mStream, Opcode::Binary, sSignalProperties.str() );
250 }
251
252 void
253 WSIOFilter::Connection::WriteMessages( const GenericSignal& signal )
254 {
255     WriteMessage(mStream, Ping, "A");
256
257     if( Connected() && Listening())
258     {
259
260       // Serialize the GenericSignal
261       stringstream ssSignal;
262       ssSignal << uint8_t( 0x04 ) << uint8_t( 0x01 );
263       signal.Serialize( ssSignal );
264       WriteMessage( mStream, Opcode::Binary, ssSignal.str() );
265
266       // Serialize the State Vector
267       stringstream ssStates;
268       ssStates << uint8_t( 0x05 );
269       Statevector->Serialize( ssStates );
270       WriteMessage( mStream, Opcode::Binary, ssStates.str() );
271
272       mListening = false;
273      }
274 }
```

B.3 BlackrockADC.cpp/h

```
////////////////////////////////////////////////////////////////////////////////
// $Id$
// Authors: griffin.milsap@gmail.com
// Description: Implementation of a source module for Blackrock systems
//
// $BEGIN_BCI2000_LICENSE$
//
// This file is part of BCI2000, a platform for real-time bio-signal research.
// [ Copyright (C) 2000-2012: BCI2000 team and many external contributors ]
//
// BCI2000 is free software: you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the Free Software
// Foundation, either version 3 of the License, or (at your option) any later
// version.
//
// BCI2000 is distributed in the hope that it will be useful, but
//                         WITHOUT ANY WARRANTY
// - without even the implied warranty of MERCHANTABILITY or FITNESS FOR
// A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along with
// this program.  If not, see <http://www.gnu.org/licenses/>.
//
// $END_BCI2000_LICENSE$
////////////////////////////////////////////////////////////////////////////////

#ifndef INCLUDED_BLACKROCKADC_H
#define INCLUDED_BLACKROCKADC_H

#include "BufferedADC.h"
#include "PrecisionTime.h"
#include "OSMutex.h"
#include "Expression.h"

#include "cbsdk.imports.h"

#include <vector>
#include <queue>

class BlackrockADC : public BufferedADC
{
 public:
          BlackrockADC();
  virtual ~BlackrockADC();
  virtual void OnAutoConfig();
  virtual void OnHalt();
  virtual void OnPreflight( SignalProperties& Output ) const;
  virtual void OnInitialize( const SignalProperties& Output );
  virtual void OnProcess();
  virtual void StartRun();
  virtual void OnStartAcquisition();
  virtual void DoAcquire( GenericSignal& Output );
  virtual void OnStopAcquisition();
  virtual void StopRun();

  // Bit of a hack necessary for digital output based on expressions
```

```
57   virtual void Process( const GenericSignal& Input, GenericSignal& Output );
58
59  private:
60   struct ChanInfo
61   {
62     unsigned int inst;
63     unsigned int idx;
64     double gain;
65     double offset;
66     std::string label;
67   };
68
69   bool Connect( int nInstances = 1 ) const;
70   void Disconnect( int nInstances = 1 ) const;
71   bool GetChannelConfig( int iNumInstances, int iGroup,
72                          std::vector< ChanInfo > &oChanConfig,
73                          std::vector< int > &oSyncChans ) const;
74   static void DataCallback( UINT32 iInstance, const cbSdkPktType iType, const void
       * iData, void* iBlackrockADC );
75   static bool CereLinkError( cbSdkResult res );
76   int GetRequestedSampleGroup() const;
77   double ScalingToGain( cbSCALING scaling ) const;
78
79   // Debug Functionality
80   void OutputChannelDebugInfo( int nInstances ) const;
81
82   OSMutex mDataMutex, mCommandMutex;
83   std::vector< std::queue< std::vector< INT16 > > > mDataPacketBuffers;
84   std::vector< ChanInfo > mChannelConfig;
85   std::vector< int > mSyncChans;
86   int mSampleGroup;
87   unsigned int mNSPInstances;
88   unsigned int mSampleBlockSize;
89   bool mSyncEnabled, mPauseForSync;
90
91   struct DigitalOutputExpression
92   {
93     unsigned int inst;
94     unsigned int dout;
95     Expression exp;
96   };
97   std::vector< DigitalOutputExpression > mDigitalOutputEx;
98
99 };
100
101 #endif // INCLUDED_BLACKROCKADC_H
```

```
////////////////////////////////////////////////////////////////////////////////
// $Id$
// Authors: griffin.milsap@gmail.com
// Description: Implementation of a source module for Blackrock systems
//
// This module works in a slightly nonstandard way.  The Blackrock NSP is very
// configurable using the Central application.  The initial version of this
// source module was originally intended to configure the NSP entirely
// from BCI2000, but the device is SOOO configurable (even to the extent of
// being able to set different sampling rates for individual channels), it
// became infeasable.  As such, this module will simply query the current
// channel configuration from the device and work with that.  This module is
// no longer capable of changing the Blackrock configuration; all configuration
// changes must be made in Central.
//
// The only really configurable parameter is the SamplingRate parameter which
// decides a sampling group to record from.  Because the device can record
// individual channels at different sampling rates, you must choose the sampling
// rate of the system, and channels that are currently being sampled at that rate
// are recorded into the BCI2000 data stream.  BCI2000 does not support per-
    channel
// sampling rates, so all channels must be configured to sample at the same rate.
// That is, you can still have some channels sampled at a different rate, but
// BCI2000 will ignore them.
//
// See more documentation on the BCI2000 WIKI.
//
// $BEGIN_BCI2000_LICENSE$
//
// This file is part of BCI2000, a platform for real-time bio-signal research.
// [ Copyright (C) 2000-2012: BCI2000 team and many external contributors ]
//
// BCI2000 is free software: you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the Free Software
// Foundation, either version 3 of the License, or (at your option) any later
// version.
//
// BCI2000 is distributed in the hope that it will be useful, but
//                         WITHOUT ANY WARRANTY
// - without even the implied warranty of MERCHANTABILITY or FITNESS FOR
// A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along with
// this program.  If not, see <http://www.gnu.org/licenses/>.
//
// $END_BCI2000_LICENSE$
////////////////////////////////////////////////////////////////////////////////
#include "BlackrockADC.h"
#include "BCIStream.h"

using namespace std;

// This list should REALLY be included in the CBSDK library somewhere
// There are a maximum of 8 sample groups. When you query sample groups
// from the API, they are 1-indexed (hence, GroupRates[0] = 0).  These
// sample groups correspond to the output NSx files.  NS2 files are sampled
// at 1k, ns4 at 10k, etc.  The raw sample group is cbRAWGROUP (6) and is
// also sampled at 30k.  Currently, there is no way to query this information
```

```
// from the CBSDK API, but may change in the future.
int gGroupRates[] = { 0, 500, 1000, 2000, 10000, 30000 }; // samples per second
int gBlockSizes[] = { 0, 10,  20,   40,   200,   600   }; // samples per block (50 Hz)

// IT IS CRUCIAL THAT THE SYSTEM IS NOT RUN THROUGH MORE THAN ONE SWITCH!
// IF IT ISN'T, THERE WILL VERY LIKELY BE DROPPED PACKETS DURING THE
// CONFIGURATION PROCESS AND CBSDK HAS UNDEFINED BEHAVIOR.
string gPktErrMsg =
  "This error suggests an incomplete system configuration due to dropped "
  "packets early in the connection process.  Please ensure that this machine "
  "is connected to the NSP through a SINGLE switch of high commercial quality.";

// Multi-NSP Synchronization Channels
// Connect the output sync channel on ONE amp to the input sync channel on all amps.
// The analog sync input channels on all amps will be monitored and delays will be adjusted
#define NSP_SYNC_OUTPUT_CHANNEL cbFIRST_DIGOUT_CHAN + 1 // Digital Output 1 (153)
#define NSP_SYNC_INPUT_CHANNEL cbFIRST_ANAIN_CHAN + cbNUM_ANAIN_CHANS // Analog Input 16 (144)
#define NSP_SYNC_THRESHOLD 1000 // Threshold for sync pulse from Digital Output 1

RegisterFilter( BlackrockADC, 1 );

BlackrockADC::BlackrockADC()
{
 BEGIN_PARAMETER_DEFINITIONS

    "Source:Signal%20Properties float SamplingRate= 10000Hz "
       "10000Hz 1000 30000 // sample rate",

    // Although you CAN specify these, it'd be best if you let the module
    // autoconfigure them individually based on the SamplingRate.
    "Source auto SourceCh= auto ",
    "Source auto SampleBlockSize= auto ",
    "Source auto ChannelNames= auto ",
    "Source auto SourceChOffset= auto ",
    "Source auto SourceChGain= auto ",

    // Experimental Multi-NSP support
    "Source:Blackrock int NSPInstances= 1 "
      "1 1 4 // Number of synchronized NSPs to record from",

    // Digital Output expressions
    "Source:Blackrock matrix DigitalOutput= 1 { Instance Output Expression } % % % "
      "% % % // Expressions to control the digital outputs"

 END_PARAMETER_DEFINITIONS

 BEGIN_STREAM_DEFINITIONS
    "NSPSyncState 2 0 0 0",
 END_STREAM_DEFINITIONS
}

BlackrockADC::~BlackrockADC()
{
```

```
111
112 }
113
114 void
115 BlackrockADC::OnAutoConfig()
116 {
117   // Attempt autoconfiguration
118   int nInstances = Parameter( "NSPInstances" );
119   if( Connect( nInstances ) ) {
120     int group = 0;
121     if( group = GetRequestedSampleGroup() )
122     {
123       // Suggest a good SampleBlockSize
124       Parameter( "SampleBlockSize" ) = gBlockSizes[ group ];
125
126       // Acquire the channel configuration
127       vector< ChanInfo > chanConfig;
128       vector< int > syncChans;
129       if( GetChannelConfig( nInstances, group, chanConfig, syncChans ) )
130       {
131         // Resize signal source parameters accordingly
132         size_t numChans = chanConfig.size();
133         Parameter( "SourceCh" ) = numChans;
134         Parameter( "SourceChOffset" )->SetNumValues( numChans );
135         Parameter( "SourceChGain" )->SetNumValues( numChans );
136         Parameter( "ChannelNames" )->SetNumValues( numChans );
137
138         // Populate channel information
139         for( unsigned int i = 0; i < numChans; i++ )
140         {
141           Parameter( "SourceChOffset" )( i ) = chanConfig[i].offset;
142           Parameter( "SourceChGain" )( i ) = chanConfig[i].gain;
143           Parameter( "ChannelNames" )( i ) = chanConfig[i].label;
144         }
145       } else bcierr << "Error acquiring channel configuration from NSP." << endl;
146     } else bcierr << "The requested SamplingRate does not correspond to a valid
      NSP SampleGroup." << endl;
147   } else bcierr << "Could not establish connection to cbsdk" << endl;
148
149   Disconnect( nInstances );
150 }
151
152 void
153 BlackrockADC::OnPreflight( SignalProperties& Output ) const
154 {
155   int nInstances = Parameter( "NSPInstances" );
156   if( Connect( nInstances ) ) {
157     int group = 0;
158     if( group = GetRequestedSampleGroup() )
159     {
160       // Acquire the channel configuration
161       vector< ChanInfo > chanConfig;
162       vector< int > syncChans;
163       if( GetChannelConfig( nInstances, group, chanConfig, syncChans ) )
164       {
165         bool sync_procedure_enabled = true;
166         for( size_t i = 0; i < syncChans.size(); i++ )
167           if( syncChans[ i ] == -1 )
```

```
            sync_procedure_enabled = false;
        if( nInstances > 1 && !sync_procedure_enabled )
          bciwarn << "Multi-NSP synchronization procedure not configured properly.
    "
                  << "Please ensure Analog Input 16 on all NSPs is sampled at the
    "
                  << "requested sampling rate and that all are connected to "
                  << "Digital Output 1 of any ONE amplifier. PROCEED WITH CAUTION!
    ";

        int numChans = chanConfig.size();

        bool goodOffsets = true,
             goodGains   = true,
             goodNames   = true;
        string matchMessage = " parameter must match the number of channels"
                              " in the requested sample group";

        // Check the SourceCh parameter
        if( Parameter( "SourceCh" ) != numChans )
          bcierr << "The SourceCh "
                 << matchMessage
                 << " (" << numChans << ") ";

        // Check the channel offsets
        if( Parameter( "SourceChOffset" )->NumValues() != numChans )
          bcierr << "The number of values in the SourceChOffset"
                 << matchMessage
                 << " (" << numChans << ") ";
        else
          for( unsigned int i = 0; i < numChans; ++i )
          {
            double chOffset = chanConfig[i].offset;
            double prmoffset = Parameter( "SourceChOffset" )( i );
            bool same = ( 1e-3 > ::fabs( prmoffset - chOffset ) / ( chOffset ?
    chOffset : 1.0 ) );
            goodOffsets &= same;

            if ( !same ) bciwarn << "CBSDK suggests the offset of"
                                 << " channel " << i + 1
                                 << " is " << chOffset
                                 << " whereas the corresponding value in the"
                                 << " SourceChOffset parameter is " << prmoffset;
          }

        if( !goodOffsets )
          bcierr << "The SourceChOffset values must match the channel "
                 << "resolutions settings in the recording software.";

        // Check gains and ensure they match up with what is reported by cbsdk
        if( Parameter( "SourceChGain" )->NumValues() != numChans )
          bcierr << "The number of values in the SourceChGain"
                 << matchMessage
                 << " (" << numChans << ") ";
        else
          for( unsigned int i = 0; i < numChans; ++i )
          {
            double chGain = chanConfig[i].gain;
```

```
222             double prmgain = Parameter( "SourceChGain" )( i );
223             bool same = ( 1e-3 > ::fabs( prmgain - chGain ) / ( chGain ? chGain :
    1.0 ) );
224             goodGains &= same;
225
226             if ( !same ) bciwarn << "CBSDK suggests the gain of"
227                                  << " channel " << i + 1
228                                  << " is " << chGain
229                                  << " whereas the corresponding value in the"
230                                  << " SourceChGain parameter is " << prmgain;
231           }
232
233         if( !goodGains )
234           bcierr << "The SourceChGain values must match the channel "
235                  << "resolutions settings in the recording software.";
236
237         // Check names and ensure they match up with what is reported by cbsdk
238         if( Parameter( "ChannelNames" )->NumValues() != numChans )
239           bcierr << "The number of values in the ChannelNames"
240                  << matchMessage
241                  << " (" << numChans << ") ";
242         else
243           for( unsigned int i = 0; i < numChans; ++i )
244           {
245             string prmlabel = Parameter( "ChannelNames" )( i );
246             string sdklabel = chanConfig[i].label;
247             bool same = prmlabel == sdklabel;
248             goodNames &= same;
249
250             if( !same ) bciwarn << "The CBSDK says channel " << i + 1
251                                 << " is labeled " << sdklabel
252                                 << " whereas the corresponding value in the"
253                                 << " ChannelNames parameter is " << prmlabel;
254           }
255
256         if( !goodNames )
257           bciwarn << "The ChannelNames values should ideally match the channel "
258                   << "labels in Central to avoid confusion later.";
259
260       } else bcierr << "Error acquiring channel configuration from NSP." << endl;
261     } else bcierr << "The requested SamplingRate does not correspond to a valid
    NSP SampleGroup." << endl;
262
263     // Check the Digital Output Expressions
264     GenericSignal preflightSignal = GenericSignal( Output );
265     for( int i = 0; i < Parameter( "DigitalOutput" )->NumRows(); i++ )
266     {
267       if( Parameter( "DigitalOutput" )( i, "Instance" ) == "" )
268         continue;
269
270       DigitalOutputExpression exp;
271       exp.inst = Parameter( "DigitalOutput" )( i, "Instance" );
272       exp.dout = Parameter( "DigitalOutput" )( i, "Output" );
273       exp.exp = Expression( Parameter( "DigitalOutput" )( i, "Expression" ) );
274
275       if( exp.inst < 0 || exp.inst >= nInstances )
276         bcierr << "Row " << i + 1 << " of DigitalOutput specifies an invalid NSP
    instance (Hint: These are 0 indexed)" << endl;
```

```
277       if( exp.dout <= 0 || exp.dout > cbNUM_DIGOUT_CHANS )
278         bcierr << "Row " << i + 1 << " of DigitalOutput specifies an invalid
      Digital Output Channel. "
279                << "Valid digital outputs are 1-" << cbNUM_DIGOUT_CHANS << ")." <<
       endl;
280
281       int dig_chan = cbFIRST_DIGOUT_CHAN + exp.dout;
282       if( CereLinkError( cbSdkSetDigitalOutput( exp.inst, dig_chan, 0 ) ) )
283         bcierr << "CBSDK digital output test failed for port " << exp.dout;
284
285       bciout << "Evaluating " << Parameter( "DigitalOutput" )( i, "Expression" );
286       exp.exp.Evaluate( &preflightSignal );
287     }
288
289   } else bcierr << "Could not re-establish connection to cbsdk" << endl;
290
291   Disconnect( nInstances );
292   State( "NSPSyncState" );
293
294   // We will place no limits on SampleBlockSize because cbsdk
295   // receives individual frames from the NSP anyway.
296   int numberOfChannels = Parameter( "SourceCh" );
297   int samplesPerBlock  = Parameter( "SampleBlockSize" );
298   SignalType sigType = SignalType::int16;
299   Output = SignalProperties( numberOfChannels, samplesPerBlock, sigType );
300 }
301
302 void
303 BlackrockADC::OnInitialize( const SignalProperties& Output )
304 {
305   mSampleGroup = GetRequestedSampleGroup();
306   mSampleBlockSize  = Parameter( "SampleBlockSize" );
307   mNSPInstances = ( unsigned int )Parameter( "NSPInstances" );
308   mDataPacketBuffers.clear();
309   mPauseForSync = false;
310   State( "NSPSyncState" ) = 1;
311
312   // Populate the Digital Output Expressions
313   mDigitalOutputEx.clear();
314   for( int i = 0; i < Parameter( "DigitalOutput" )->NumRows(); i++ )
315   {
316     if( Parameter( "DigitalOutput" )( i, "Instance" ) == "" )
317       continue;
318
319     DigitalOutputExpression exp;
320     exp.inst = Parameter( "DigitalOutput" )( i, "Instance" );
321     exp.dout = Parameter( "DigitalOutput" )( i, "Output" );
322     exp.exp = Expression( Parameter( "DigitalOutput" )( i, "Expression" ) );
323     mDigitalOutputEx.push_back( exp );
324   }
325 }
326
327 void
328 BlackrockADC::OnStartAcquisition()
329 {
330   if( Connect( mNSPInstances ) )
331   {
332     // Acquire the channel configuration
```

```
333     if( !GetChannelConfig( mNSPInstances, mSampleGroup, mChannelConfig, mSyncChans
        ) )
334       bcierr << "Couldn't acquire channel configuration" << endl;
335
336     // Determine if synchronization is enabled...
337     mSyncEnabled = true;
338     for( size_t i = 0; i < mSyncChans.size(); i++ )
339       if( mSyncChans[ i ] == -1 )
340         mSyncEnabled = false;
341
342     mDataMutex.Acquire();
343     for( int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
344     {
345       // Ready a queue for our data packets
346       mDataPacketBuffers.push_back( queue< vector< INT16 > >() );
347
348       // Register data packet callback
349       CereLinkError( cbSdkRegisterCallback( instIdx, CBSDKCALLBACK_CONTINUOUS,
    DataCallback, this ) );
350     }
351     mDataMutex.Release();
352   }
353   else
354   {
355     // Error and ensure we disconnect from any NSPs we connected to.
356     bcierr << "Could not re-establish connection to cbsdk" << endl;
357     Disconnect( mNSPInstances );
358   }
359 }
360
361 void
362 BlackrockADC::DoAcquire( GenericSignal& Output )
363 {
364   // Multi-NSP Synchronization Procedure
365   if( mPauseForSync )
366   {
367     // Set the digital output low (is currently high) and collect some data
368     bciout << "Starting Multi-NSP Synchronization Procedure....";
369     mCommandMutex.Acquire();
370     for( int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
371       cbSdkSetDigitalOutput( instIdx, NSP_SYNC_OUTPUT_CHANNEL, 0 );
372     mCommandMutex.Release();
373     ThreadUtils::SleepForMs( 100 ); // Average Latency is about 20 ms
374
375     // Calculate the delays that exist before the two synchronization input steps.
376     for( int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
377     {
378       int delay = 0;
379       stringstream debugStream;
380       for( int i = 0; i < mDataPacketBuffers[ instIdx ].size(); i++ )
381       {
382         int value = mDataPacketBuffers[ instIdx ].front()[ mSyncChans[ instIdx ]
    ];
383         debugStream << value << " ";
384         if( value < NSP_SYNC_THRESHOLD ) break; // Falling edge
385         mDataPacketBuffers[ instIdx ].pop();
386         delay++;
387       }
```

```
388       bcidbg << "DEBUG SYNC: Inst " << instIdx << ": " << debugStream.str();
389       bciout << "Corrected Instance " << instIdx << " by " << delay << " samples."
      ;
390     }
391
392     bciout << "Synchronization Complete.";
393     mPauseForSync = false;
394   }
395
396   // Wait while there isn't enough data in the queue
397   while( true )
398   {
399     bool dataQueued = true;
400     for( int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
401       dataQueued &= mDataPacketBuffers[ instIdx ].size() >= mSampleBlockSize;
402     if( dataQueued == true ) break;
403
404     ThreadUtils::SleepForMs( 1 );
405   }
406
407   // Dequeue data into the signal output
408   mDataMutex.Acquire();
409   for( int fr_idx = 0; fr_idx < Output.Elements(); fr_idx++ )
410   {
411     for( int ch_idx = 0; ch_idx < Output.Channels(); ch_idx++ )
412     {
413       int inst = mChannelConfig[ ch_idx ].inst;
414       int idx = mChannelConfig[ ch_idx ].idx;
415       Output( ch_idx, fr_idx ) = mDataPacketBuffers[ inst ].front()[ idx ];
416     }
417
418     // Pop the recorded frame off the buffers
419     for( int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
420       mDataPacketBuffers[ instIdx ].pop();
421   }
422   mDataMutex.Release();
423 }
424
425 void
426 BlackrockADC::StartRun()
427 {
428 }
429
430 void
431 BlackrockADC::OnProcess()
432 {
433   // Handle state machine for synchronization procedure
434   if( State( "NSPSyncState" ) == 1 )
435   {
436     mCommandMutex.Acquire();
437     for( int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
438       cbSdkSetDigitalOutput( instIdx, NSP_SYNC_OUTPUT_CHANNEL, 1 );
439     State( "NSPSyncState" ) = 2;
440     mCommandMutex.Release();
441   }
442   else if( State( "NSPSyncState" ) == 2 )
443   {
444     mCommandMutex.Acquire();
```

```
445     if( mSyncEnabled ) mPauseForSync = true;
446     else for( int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
447       cbSdkSetDigitalOutput( instIdx, NSP_SYNC_OUTPUT_CHANNEL, 0 );
448     State( "NSPSyncState" ) = 0;
449     mCommandMutex.Release();
450   }
451 }
452
453 void
454 BlackrockADC::Process( const GenericSignal& Input, GenericSignal& Output )
455 {
456   // Call Superclass!
457   BufferedADC::Process( Input, Output );
458
459   // Set digital outputs as necessary
460   mCommandMutex.Acquire();
461   for( size_t i = 0; i < mDigitalOutputEx.size(); i++ )
462   {
463     DigitalOutputExpression ex = mDigitalOutputEx[ i ];
464     int dig_chan = cbFIRST_DIGOUT_CHAN + ex.dout;
465     int value = int( bool( ex.exp.Evaluate( &Output ) ) );
466     cbSdkSetDigitalOutput( ex.inst, dig_chan, value );
467   }
468   mCommandMutex.Release();
469 }
470
471 void
472 BlackrockADC::StopRun()
473 {
474 }
475
476 void
477 BlackrockADC::OnStopAcquisition()
478 {
479   Disconnect( mNSPInstances );
480 }
481
482 void
483 BlackrockADC::OnHalt()
484 {
485 }
486
487 bool
488 BlackrockADC::Connect( int nInstances ) const
489 {
490   if( nInstances < 1 ) nInstances = 1;
491   for( int instIdx = 0; instIdx < nInstances; instIdx++ )
492   {
493     // Open a connection to the SDK
494     cbSdkConnectionType conType = CBSDKCONNECTION_DEFAULT;
495     if( CereLinkError( cbSdkOpen( instIdx, conType, cbSdkConnection() ) ) )
496     {
497       bcierr << "Could not establish connection to cbsdk for instance " << instIdx
      << ".  "
498              << "CBSDK may not always connect even if your system is set up
    properly. "
499              << "Try to 'ping " << cbNET_UDP_ADDR_CNT << "'" << endl;
500       return false;
```

```
501     }
502
503     // Determine the SDK connection type
504     cbSdkInstrumentType instType;
505     if( CereLinkError( cbSdkGetType( instIdx, &conType, &instType ) ) )
506     {
507       bcierr << "Unable to determine connection type for instance " << instIdx <<
        endl;
508       return false;
509     }
510
511     // Get the NSP Version
512     cbSdkVersion ver;
513     if( CereLinkError( cbSdkGetVersion( instIdx, &ver ) ) )
514     {
515       bcierr << "Unable to get NSP version for instance " << instIdx << ".  "
516              << "Is device connected and on?" << endl;
517       return false;
518     }
519
520     // Assemble a debug string for the connection information
521     if( conType < 0 || conType > CBSDKCONNECTION_COUNT ) conType =
        CBSDKCONNECTION_COUNT;
522     if( instType < 0 || instType > CBSDKINSTRUMENT_COUNT ) instType =
        CBSDKINSTRUMENT_COUNT;
523     string strConnection[] = { "Default", "Central", "UDP", "Closed", "Unknown" };
524     string strInstrument[] = { "NSP", "nPlay", "Local NSP", "Remote nPlay", "
        Unknown" };
525     bciout << "Instance " << instIdx << ": " << strConnection[ conType ]
526            << " real-time interface to " << strInstrument[ instType ]
527            << "(V" << ver.nspmajor << "." << ver.nspminor << "."
528            << ver.nsprelease << "." << ver.nspbeta << ")";
529   }
530
531   return true;
532 }
533
534 bool
535 BlackrockADC::GetChannelConfig( int iNumInstances, int iGroup,
536                                 vector< ChanInfo > &oChanConfig,
537                                 vector< int > &oSyncChans ) const
538 { // BlackrockADC::Connect must be called first!
539
540   oChanConfig.clear();
541   oSyncChans.clear();
542
543   // Get the total number of channels in this sample group across all instances
544   for( int instIdx = 0; instIdx < iNumInstances; instIdx++ )
545   {
546     int sync_input_ch = -1;
547
548     // Determine the number of channels in the requested channel group on this
        instance.
549     unsigned int nLength = 0;
550     CereLinkError( cbSdkGetSampleGroupList( instIdx, 1, iGroup, &nLength, NULL ) )
        ;
551     if( nLength <= 0 ) continue;
552
```

```
553     // Get the list of channels in this sample group
554     unsigned int *pGroupList = new unsigned int[ nLength ];
555     CereLinkError( cbSdkGetSampleGroupList( instIdx, 1, iGroup, &nLength,
        pGroupList ) );
556
557     // Populate output structures with information about this channel
558     for( unsigned int i = 0; i < nLength; i++ )
559     {
560       cbPKT_CHANINFO* pChanInfo = new cbPKT_CHANINFO();
561       if( !CereLinkError( cbSdkGetChannelConfig( instIdx, pGroupList[i], pChanInfo
         ) ) )
562       {
563         ChanInfo newChan;
564         newChan.offset = 0;
565         newChan.gain = ScalingToGain( pChanInfo->physcalin );
566         newChan.label = string( pChanInfo->label );
567         newChan.inst = instIdx;
568         newChan.idx = i;
569         oChanConfig.push_back( newChan );
570       } else bcierr << "Bad Channel Index: " << pGroupList[i] << endl <<
        gPktErrMsg;
571       delete pChanInfo;
572
573       // Find the Synchronization input channel index
574       if( pGroupList[i] == NSP_SYNC_INPUT_CHANNEL ) sync_input_ch = i;
575     }
576     delete [] pGroupList;
577     oSyncChans.push_back( sync_input_ch );
578   }
579
580   // Quick sanity check
581   if( oChanConfig.size() == 0 )
582   {
583     bcierr << "There are no channels sampled at the desired SamplingRate." << endl
        ;
584     return false;
585   }
586
587   return true;
588 }
589
590 void
591 BlackrockADC::Disconnect( int nInstances ) const
592 {
593   if( nInstances < 1 ) nInstances = 1;
594   for( int instIdx = 0; instIdx < nInstances; instIdx++ )
595     // Disconnect from the instrument
596     CereLinkError( cbSdkClose( instIdx ) );
597 }
598
599 void BlackrockADC::DataCallback( UINT32 iInstance, const cbSdkPktType iType, const
       void* iData, void* iBlackrockADC )
600 {
601   // Re-establish communication to the class
602   BlackrockADC* bradc = reinterpret_cast< BlackrockADC* >( iBlackrockADC );
603
604   switch( iType )
605   {
```

```
606   case cbSdkPkt_PACKETLOST:
607     bcierr << "Packet loss.  Data has been lost.  Reduce system load." << endl;
608     break;
609   case cbSdkPkt_CONTINUOUS:
610     if( bradc && iData )
611     {
612       // Grab the packet and ensure that it is the right sample group
613       const cbPKT_GROUP *pPkt = reinterpret_cast< const cbPKT_GROUP* >( iData );
614       if( pPkt->type != bradc->mSampleGroup ) break;
615 
616       // We'll just queue the data packet
617       bradc->mDataMutex.Acquire();
618       size_t pkt_len = sizeof( pPkt->data ) / sizeof( pPkt->data[0] );
619       bradc->mDataPacketBuffers[ iInstance ].push( vector< INT16 >( pPkt->data,
    pPkt->data + pkt_len ) );
620       bradc->mDataMutex.Release();
621     }
622     break;
623   default:
624     break;
625   }
626   return;
627 }
628 
629 // Check the SamplingRate parameter and return a Blackrock group index
630 // If this returns 0, the sampling rate is not a valid rate.
631 int
632 BlackrockADC::GetRequestedSampleGroup() const
633 {
634   int samplingRate = ( int )Parameter( "SamplingRate" );
635   int num_rates = sizeof( gGroupRates ) / sizeof( int );
636   for( int rate_idx = 0; rate_idx < num_rates; rate_idx++ )
637     if( samplingRate == gGroupRates[ rate_idx ] )
638       return rate_idx;
639   return 0;
640 }
641 
642 // Convert a cbSCALING parameter to a SourceChGain
643 double
644 BlackrockADC::ScalingToGain( cbSCALING scaling ) const
645 {
646   double anaRange = scaling.anamax - scaling.anamin;
647   double digRange = scaling.digmax - scaling.digmin;
648   return ( anaRange / digRange );
649 }
650 
651 // Debug channel configurations
652 void
653 BlackrockADC::OutputChannelDebugInfo( int nInstances ) const
654 {
655   for( unsigned int instIdx = 0; instIdx < mNSPInstances; instIdx++ )
656   {
657     for( unsigned int ch = 0; ch < cbMAXCHANS; ch++ )
658     {
659       cbPKT_CHANINFO* pChanInfo = new cbPKT_CHANINFO();
660       if( !CereLinkError( cbSdkGetChannelConfig( instIdx, ch, pChanInfo ) ) )
661         bciout << pChanInfo->label << " -- "
662                << "Inst: " << instIdx  << ", "
```

```
                << "Bank: " << pChanInfo->bank << ", "
                << "Chan: " << pChanInfo->chan;
      delete pChanInfo;
    }
  }
}

// Misc Cerelink error checking
bool
BlackrockADC::CereLinkError( cbSdkResult res )
{
  cbSdkInstrumentType instType;
  switch( res )
  {
  case CBSDKRESULT_WARNCONVERT:
    bciwarn << "File conversion is needed..." << endl;
    return false;
  case CBSDKRESULT_WARNCLOSED:
    bciwarn << "Library is already closed" << endl;
    return false;
  case CBSDKRESULT_WARNOPEN:
    bciwarn << "Library is already opened" << endl;
    return false;
  case CBSDKRESULT_SUCCESS:
    //bcidbg( 0 ) << "Success" << endl;
    return false;
  case CBSDKRESULT_NOTIMPLEMENTED:
    bcierr << "Not implemented" << endl;
    return true;
  case CBSDKRESULT_INVALIDPARAM:
    bcierr << "Invalid parameter" << endl;
    return true;
  case CBSDKRESULT_CLOSED:
    bcierr << "Interface is closed cannot do this operation" << endl;
    return true;
  case CBSDKRESULT_OPEN:
    bcierr << "Interface is open cannot do this operation" << endl;
    return true;
  case CBSDKRESULT_NULLPTR:
    bcierr << "Null pointer" << endl;
    return true;
  case CBSDKRESULT_ERROPENCENTRAL:
    bcierr << "Unable to open Central interface" << endl;
    return true;
  case CBSDKRESULT_ERROPENUDP:
    bcierr << "Unable to open UDP interface (might happen if default)" << endl;
    return true;
  case CBSDKRESULT_ERROPENUDPPORT:
    bcierr << "Unable to open UDP port" << endl;
    return true;
  case CBSDKRESULT_ERRMEMORYTRIAL:
    bcierr << "Unable to allocate RAM for trial cache data" << endl;
    return true;
  case CBSDKRESULT_ERROPENUDPTHREAD:
    bcierr << "Unable to open UDP timer thread" << endl;
    return true;
  case CBSDKRESULT_ERROPENCENTRALTHREAD:
    bcierr << "Unable to open Central communication thread" << endl;
```

```
    return true;
  case CBSDKRESULT_INVALIDCHANNEL:
    bcierr << "Invalid channel number" << endl;
    return true;
  case CBSDKRESULT_INVALIDCOMMENT:
    bcierr << "Comment too long or invalid" << endl;
    return true;
  case CBSDKRESULT_INVALIDFILENAME:
    bcierr << "Filename too long or invalid" << endl;
    return true;
  case CBSDKRESULT_INVALIDCALLBACKTYPE:
    bcierr << "Invalid callback type" << endl;
    return true;
  case CBSDKRESULT_CALLBACKREGFAILED:
    bcierr << "Callback register/unregister failed" << endl;
    return true;
  case CBSDKRESULT_ERRCONFIG:
    bcierr << "Trying to run an unconfigured method" << endl;
    return true;
  case CBSDKRESULT_INVALIDTRACKABLE:
    bcierr << "Invalid trackable id, or trackable not present" << endl;
    return true;
  case CBSDKRESULT_INVALIDVIDEOSRC:
    bcierr << "Invalid video source id, or video source not present" << endl;
    return true;
  case CBSDKRESULT_ERROPENFILE:
    bcierr << "Cannot open file" << endl;
    return true;
  case CBSDKRESULT_ERRFORMATFILE:
    bcierr << "Wrong file format" << endl;
    return true;
  case CBSDKRESULT_OPTERRUDP:
    bcierr << "Socket option error (Possibly permission issue)" << endl;
    return true;
  case CBSDKRESULT_MEMERRUDP:
    bcierr << "Socket memory assignment error" << endl
           << " Consider using sysctl -w net.core.rmem_max=8388608" << endl
           << " or sysctl -w kern.ipc.maxsockbuf=8388608" << endl;
    return true;
  case CBSDKRESULT_INVALIDINST:
    bcierr << "Invalid range or instrument address" << endl;
    return true;
  case CBSDKRESULT_ERRMEMORY:
    bcierr << "Library memory allocation error" << endl;
    return true;
  case CBSDKRESULT_ERRINIT:
    bcierr << "Library initialization error" << endl;
    return true;
  case CBSDKRESULT_TIMEOUT:
    bcierr << "Connection timeout error" << endl;
    return true;
  case CBSDKRESULT_BUSY:
    bcierr << "Resource is busy" << endl;
    return true;
  case CBSDKRESULT_ERROFFLINE:
    bcierr << "Instrument is offline" << endl;
    return true;
  case CBSDKRESULT_UNKNOWN:
```

```
779   default:
780     bcierr << "Unknown error.  Sorry!" << endl;
781     return true;
782   }
783 }
```

B.4 NihonKohdenADC.cpp/h

```
////////////////////////////////////////////////////////////////////////////////
// $Id: $
// Authors: Kienan Knight-Boehm (kienan {at} kienankb.com)
// Description: NihonKohdenADC header
//
//
// $BEGIN_BCI2000_LICENSE$
//
// This file is part of BCI2000, a platform for real-time bio-signal research.
// [ Copyright (C) 2000-2012: BCI2000 team and many external contributors ]
//
// BCI2000 is free software: you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the Free Software
// Foundation, either version 3 of the License, or (at your option) any later
// version.
//
// BCI2000 is distributed in the hope that it will be useful, but
//                         WITHOUT ANY WARRANTY
// - without even the implied warranty of MERCHANTABILITY or FITNESS FOR
// A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along with
// this program.  If not, see <http://www.gnu.org/licenses/>.
//
// $END_BCI2000_LICENSE$
////////////////////////////////////////////////////////////////////////////////
#ifndef INCLUDED_NIHONKOHDEN_ADC_H
#define INCLUDED_NIHONKOHDEN_ADC_H

#include "BufferedADC.h"
#include "EegDataSource.h"

class NihonKohdenADC : public BufferedADC
{
 public:
           NihonKohdenADC();
  virtual ~NihonKohdenADC();
  virtual void OnPublish();
  virtual void OnAutoConfig();
  virtual void OnPreflight( SignalProperties& Output ) const;
  virtual void OnInitialize( const SignalProperties& Output );
  virtual void OnStartAcquisition();
  virtual void DoAcquire( GenericSignal& Output );
  virtual void OnStopAcquisition();

 private:
  static void CALLBACK DataSourceStateChanged( int nState, int nSubState, void *
    pAddInfo );
  bool NKErrorCheck( int val ) const;

  unsigned long Connect( const std::string &deviceAddress ) const;
  void Disconnect( unsigned long identifier ) const;

  bool mAutoCh;
  std::map< int, int > mChannelIndices;
  unsigned long mIdentifier;
```

```
  unsigned int mBufferChannels;
  int mNumberOfSignalChannels;
  float* mpBuffer;
  unsigned int mChannelCount;

};

#endif // INCLUDED_NIHONKOHDEN_ADC_H
```

```
////////////////////////////////////////////////////////////////////////////////
// $Id: $
// Authors: Kienan Knight-Boehm (kienan {at} kienankb.com}
//          Griffin Milsap (griffin.milsap@gmail.com)
// Description: NihonKohdenADC implementation
//
// $BEGIN_BCI2000_LICENSE$
//
// This file is part of BCI2000, a platform for real-time bio-signal research.
// [ Copyright (C) 2000-2012: BCI2000 team and many external contributors ]
//
// BCI2000 is free software: you can redistribute it and/or modify it under the
// terms of the GNU General Public License as published by the Free Software
// Foundation, either version 3 of the License, or (at your option) any later
// version.
//
// BCI2000 is distributed in the hope that it will be useful, but
//                         WITHOUT ANY WARRANTY
// - without even the implied warranty of MERCHANTABILITY or FITNESS FOR
// A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along with
// this program.  If not, see <http://www.gnu.org/licenses/>.
//
// $END_BCI2000_LICENSE$
////////////////////////////////////////////////////////////////////////////////

// ?? -- Griff
#define _X86_

#include "NihonKohdenADC.h"
#include "BCIStream.h"
#include "BCIEvent.h"
#include "ThreadUtils.h"
#include "StringUtils.h"
#include "FilePath.h"

#include <map>
#include <vector>

using namespace std;

class Setting
{
 public:
  Setting( unsigned int sourceCh = 0, int sampleBlockSize = 0 ) :
    maxSourceCh( sourceCh ),
    recSampleBlockSize( sampleBlockSize )
  { }

  unsigned int maxSourceCh;
  int recSampleBlockSize;
};

typedef map< unsigned int, Setting > SettingMap;

// Map sampling rate to amp settings
SettingMap CreateSettings()
```

```
{
  SettingMap settings;
  settings[ 100   ] = Setting( 256, 10 );
  settings[ 200   ] = Setting( 256, 20 );
  settings[ 500   ] = Setting( 256, 60 );
  settings[ 1000  ] = Setting( 256, 100 );
  settings[ 2000  ] = Setting( 256, 200 );
  settings[ 5000  ] = Setting( 128, 500 );
  settings[ 10000 ] = Setting( 64,  1000 );
  return settings;
}

static SettingMap Settings = CreateSettings();

vector< int > CreateEEGIndices()
{
  vector< int > indices;
  for( int i = 22; i <= 85; ++i ) // Bank A
    indices.push_back( i );
  for( int i = 93; i <= 156; ++i ) // Bank B
    indices.push_back( i );
  for( int i = 157; i <= 220; ++i ) // Bank C
    indices.push_back( i );
  for( int i = 221; i <= 284; ++i ) // Bank D
    indices.push_back( i );
  return indices;
}

static vector< int > EEGIndices = CreateEEGIndices();

vector< string > CreateDefaultElectrodeLabels()
{
  vector< string > labels;

  char banks[] = { 'A', 'B', 'C', 'D' };
  for( size_t bank_idx = 0; bank_idx < sizeof( banks ) / sizeof( *banks ); ++
    bank_idx )
    for( int ch_idx = 1; ch_idx <= 64; ++ch_idx )
    {
      stringstream ss;
      ss << banks[ bank_idx ] << ch_idx;
      labels.push_back( ss.str() );
    }

  return labels;
}

static vector< string > DefaultElectrodeLabels = CreateDefaultElectrodeLabels();

vector< int > CreateDCIndices()
{
  vector< int > indices;
  for( int i = 0; i <= 15; ++i ) // DC01 - DC16
    indices.push_back( i );
  return indices;
}

static vector< int > DCIndices = CreateDCIndices();
```

```
116
117 vector< int > CreateAuxIndices()
118 {
119   vector< int > indices;
120   for( int i = 16; i <= 21; ++i )
121     indices.push_back( i );
122   for( int i = 86; i <= 92; ++i )
123     indices.push_back( i );
124   return indices;
125 }
126
127 static vector< int > AuxIndices = CreateAuxIndices();
128
129 RegisterFilter( NihonKohdenADC, 1 );
130
131 NihonKohdenADC::NihonKohdenADC() :
132   mpBuffer( NULL ),
133   mIdentifier( 0 )
134 {
135 }
136
137 NihonKohdenADC::~NihonKohdenADC()
138 {
139   if( mIdentifier != 0 )
140     Disconnect( mIdentifier );
141   delete[] mpBuffer;
142 }
143
144 void
145 NihonKohdenADC::OnPublish()
146 {
147  BEGIN_PARAMETER_DEFINITIONS
148   "Source:Signal%20Properties string DeviceAddress= auto "
149   " // Device IP address",
150
151   "Source:Signal%20Properties int SourceCh= auto auto 1 %"
152   " // number of digitized and stored channels",
153
154   "Source:Signal%20Properties int SampleBlockSize= auto auto 1 %"
155   " // number of samples transmitted at a time",
156
157   "Source:Signal%20Properties float SamplingRate= auto auto 0.0 %"
158   " // sample rate",
159
160   "Source:Signal%20Properties list SourceChGain= 1 auto "
161   " // physical units per raw A/D unit",
162
163   "Source:Signal%20Properties list SourceChOffset= 1 auto "
164   " // raw A/D offset to subtract, typically 0",
165
166   "Source:Signal%20Properties list ChannelNames= 1 auto "
167   " // names of amplifier channels",
168  END_PARAMETER_DEFINITIONS
169
170   char stateDef[ 32 ];
171   for( int i = 0; i < DCIndices.size(); ++i )
172   {
173     sprintf( stateDef, "DC%02d 16 0 0 0", i + 1 );
```

```
174     BEGIN_STREAM_DEFINITIONS
175       stateDef
176     END_STREAM_DEFINITIONS
177   }
178 }
179
180 void CALLBACK
181 NihonKohdenADC::DataSourceStateChanged( int nState, int nSubState, void * pAddInfo
        )
182 {
183   switch( nState ) {
184   case DATASOURCE_DLL_STS_ERR:
185     switch( nSubState ) {
186     case DS_WAVE_RR_TIMEOUT:
187     case DS_CMD_RR_TIMEOUT:
188     case DS_MMFILE_ERR_NEED_RESET:
189     case DS_MMFILE_OVERRUN_READ_OFFSET:
190     case DS_RECEIVE_IRREGULAR_PACKET:
191     case DS_DISCONNECT_SOCKET:
192       // Could attempt reconnection procedure here, but probably best to just
193       // throw an exception especially if we're already recording. -- Griff
194       throw bcierr << "An error occurred, necessitating reset.  System shutting
        down";
195     }
196     break;
197   case DATASOURCE_DLL_STS_RECONNECT_SUCCESS:
198     // Shouldn't ever be called; at least right now. -- Griff
199     bciout << "Reconnection succeeded.";
200   default:
201     break;
202   }
203 }
204
205 unsigned long
206 NihonKohdenADC::Connect( const string &deviceAddress ) const
207 {
208   unsigned long id = InitializeDll( DS_MODE_READER, NULL, NULL, 0 );
209   if( id == 0 ) throw bcierr << "Could not initialize NK DLL";
210
211   READER_MODE_INIT_INFO initInfo;
212
213   if( deviceAddress == "auto" )
214     initInfo.bSelectDataSource = true;
215   else {
216     initInfo.ulIpAddress = htonl( ::inet_addr( deviceAddress.c_str() ) );
217     initInfo.bSelectDataSource = false;
218   }
219
220   if( NKErrorCheck( InitializeReaderMode( id, initInfo, &NihonKohdenADC::
        DataSourceStateChanged ) ) )
221     throw bcierr << "Could not initialize reader mode!";
222   if( NKErrorCheck( ReaderModeConnect( id ) ) )
223     throw bcierr << "Could not connect to device!";
224
225   return id;
226 }
227
228 void
```

```
NihonKohdenADC::Disconnect( unsigned long id ) const
{
  if( NKErrorCheck( ReaderModeClose( id ) ) )
    bcierr << "Could not close reader mode";
  if( NKErrorCheck( ReaderModeEnd( id ) ) )
    bcierr << "Could not close dll";
}

void
NihonKohdenADC::OnAutoConfig()
{
  mIdentifier = Connect( string( Parameter( "DeviceAddress" ) ) );

  unsigned int samplingRate = 0;
  if( NKErrorCheck( MMFileGetSamplingRate( mIdentifier, samplingRate ) ) )
    bciwarn << "Couldn't get sampling rate";
  Parameter( "SamplingRate" ) << samplingRate << "Hz";

  // Set a recommended SampleBlockSize
  Parameter( "SampleBlockSize" ) = Settings[ samplingRate ].recSampleBlockSize;

  unsigned int channels = 0;
  if( NKErrorCheck( MMFileGetElectrodeCount( mIdentifier, channels ) ) )
    bciwarn << "Couldn't determine electrode count";

  // There appears to be a memory issue with the GetElectrodeName function
  // Placing the electrodeNames variable in the stack seems to mitigate
  // this to some extent.
  MMFILE_ELECTRODE_NAME* electrodeNames = new MMFILE_ELECTRODE_NAME();
  if( NKErrorCheck( MMFileGetElectrodeName( mIdentifier, *electrodeNames ) ) )
    bciwarn << "Couldn't get electrode names";

  MMFILE_ELECTRODE_CODE electrodeCodes;
  if( NKErrorCheck( MMFileGetElectrodeCode( mIdentifier, electrodeCodes ) ) )
    bciwarn << "Couldn't get electrode codes";

  // Determine how many of the default eeg channels this particular amp supports
  int numEEGChannels = 0;
  for( int i = 0; i < EEGIndices.size(); ++i )
    if( EEGIndices[ i ] < channels )
      numEEGChannels++;

  // Determine how many of the channels have non-default labels
  vector< bool > isDefaultLabel;
  int numInterestingChannels = 0;
  for( int i = 0; i < numEEGChannels; ++i )
  {
    string chName( electrodeNames->pszName[ EEGIndices[ i ] ] );
    isDefaultLabel.push_back( chName == DefaultElectrodeLabels[ i ] );
    if( !isDefaultLabel.back() ) numInterestingChannels++;
  }

  mAutoCh = string( ActualParameter( "SourceCh" ) ) == "auto";
  Parameter( "SourceCh" ) = ( ( mAutoCh && numInterestingChannels ) ?
    numInterestingChannels : numEEGChannels );
  if( mAutoCh ) Parameter( "SourceCh" ) = int( Parameter( "SourceCh" ) ) +
    DCIndices.size();
  Parameter( "ChannelNames" )->SetNumValues( ActualParameter( "SourceCh" ) );
```

```
  Parameter( "SourceChGain" )->SetNumValues( ActualParameter( "SourceCh" ) );
  Parameter( "SourceChOffset" )->SetNumValues( ActualParameter( "SourceCh" ) );

  map< string, bool > name_map;
  mChannelIndices.clear();

  int ch_idx = 0;
  for( int i = 0; i < numEEGChannels; ++i )
  {
    if( mAutoCh && numInterestingChannels && isDefaultLabel[ i ] ) continue;
    if( ch_idx >= ActualParameter( "SourceCh" ) - ( ( mAutoCh ) ? DCIndices.size()
     : 0 ) ) continue;

    mChannelIndices[ ch_idx ] = i;

    string chName( electrodeNames->pszName[ EEGIndices[ mChannelIndices[ ch_idx ]
    ] ] );
    if( chName == "" || chName == " " ) chName = "EMPTY";
    if( name_map.find( chName ) == name_map.end() )
    {
      name_map[ chName ] = true;
      Parameter( "ChannelNames" )( ch_idx ) << String() << chName;
    } else {
      Parameter( "ChannelNames" )( ch_idx ) << String() << chName << "_" << ch_idx
     + 1;
    }

    Parameter( "SourceChGain" )( ch_idx ) << 1.0;
    Parameter( "SourceChOffset" )( ch_idx ) = 0;

    ch_idx++;
  }

  // Add the DC channels as signal in addition to adding them as stream, if we are
     auto-configuring channels
  if( mAutoCh )
  {
    for( size_t i = 0; i < DCIndices.size(); ++i )
    {
      char syncChName[ 16 ];
      sprintf( syncChName, "DC%02d", i + 1 );
      Parameter( "ChannelNames" )( ch_idx ) = string( syncChName );
      Parameter( "SourceChGain" )( ch_idx ) << 1.0;
      Parameter( "SourceChOffset" )( ch_idx ) = 0;
      ch_idx++;
    }
  }

  delete electrodeNames;
}

void
NihonKohdenADC::OnPreflight( SignalProperties& Output ) const
{
  unsigned int samplingRate = 0;
  if( NKErrorCheck( MMFileGetSamplingRate( mIdentifier, samplingRate ) ) )
    bcierr << "Couldn't verify sampling rate";
  if( samplingRate != unsigned int( Parameter( "SamplingRate" ).InHertz() ) )
```

```
339     bcierr << "SamplingRate doesn't match reported sample rate of " <<
        samplingRate;
340
341   if( ( unsigned int( Parameter( "SourceCh" ) ) - ( ( mAutoCh ) ? DCIndices.size()
         : 0 ) ) > Settings[ samplingRate ].maxSourceCh )
342     bcierr << "SamplingRate of " << samplingRate
343            << " does not support recording more than "
344            << Settings[ samplingRate ].maxSourceCh << " channels. "
345            << "Please adjust SourceCh accordingly.";
346
347   SignalType sigType = SignalType::float32;
348   int samplesPerBlock = Output.Elements();
349   int numberOfSignalChannels = Output.Channels();
350   int outputSize = numberOfSignalChannels + DCIndices.size();
351   Output = SignalProperties( outputSize, samplesPerBlock, sigType );
352
353   Parameter( "DeviceAddress" );
354
355   // Append the streams
356   int chStart = numberOfSignalChannels;
357   for( size_t i = 0; i < DCIndices.size(); ++i )
358   {
359     char syncChName[ 16 ];
360     sprintf( syncChName, "@DC%02d", i + 1 );
361     Output.ChannelLabels()[ chStart + i ] = string( syncChName );
362   }
363 }
364
365 void
366 NihonKohdenADC::OnInitialize( const SignalProperties& Output )
367 {
368   mBufferChannels = 0;
369   if( NKErrorCheck( MMFileGetElectrodeCount( mIdentifier, mBufferChannels ) ) )
370     bciwarn << "Couldn't determine electrode count";
371
372   mNumberOfSignalChannels = Parameter( "SourceCh" );
373
374   // Allocate a sample buffer
375   delete[] mpBuffer;
376   int bufSize = mBufferChannels * Output.Elements();
377   mpBuffer = new float[ bufSize ];
378   ::memset( mpBuffer, 0, bufSize * sizeof( float ) );
379
380   Disconnect( mIdentifier );
381   mIdentifier = 0;
382 }
383
384 void
385 NihonKohdenADC::OnStartAcquisition()
386 {
387   mIdentifier = Connect( string( Parameter( "DeviceAddress" ) ) );
388 }
389
390 void
391 NihonKohdenADC::DoAcquire( GenericSignal& Output )
392 {
393   unsigned int frameCount = 0;
394   while( frameCount < Output.Elements() )
```

```
395   {
396     if( NKErrorCheck( GetDataFrameCount( mIdentifier, frameCount ) ) )
397       throw bcierr << "Could not get DataFrameCount";
398     ThreadUtils::SleepForMs( 1 );
399   }
400
401   if( NKErrorCheck( GetFloatData( mIdentifier, Output.Elements(),
402     frameCount, NULL, mpBuffer, NULL, NULL ) ) )
403     throw bcierr << "Could not acquire data";
404   if( frameCount != Output.Elements() )
405     throw bcierr << "Did not get requested amount of data.";
406
407   // Copy values from raw buffer into output signal.
408   for( int ch = 0; ch < mNumberOfSignalChannels - ( ( mAutoCh ) ? DCIndices.size()
       : 0 ); ch++ )
409     for( int sample = 0; sample < Output.Elements(); sample++ )
410       Output( ch, sample ) = mpBuffer[ EEGIndices[ mChannelIndices[ ch ] ] + (
      sample * mBufferChannels ) ];
411
412   int chStart = mNumberOfSignalChannels;
413   for( int i = 0; i < DCIndices.size(); i++ )
414     for( int sample = 0; sample < Output.Elements(); sample++ )
415     {
416       float datum = mpBuffer[ DCIndices[ i ] + ( sample * mBufferChannels ) ];
417       unsigned short dig_datum = int( datum / 366.3 ) + 0x8000;
418       Output( chStart + i, sample ) = dig_datum;
419       if( mAutoCh ) Output( chStart - DCIndices.size() + i, sample ) = dig_datum;
420     }
421 }
422
423 void
424 NihonKohdenADC::OnStopAcquisition()
425 {
426   Disconnect( mIdentifier );
427   mIdentifier = 0;
428 }
429
430 bool
431 NihonKohdenADC::NKErrorCheck( int val ) const
432 {
433   // Throw exception with error text if an error occurred,
434   // or return false if everything is alright.
435
436   switch( val )
437   {
438   case DS_ALREADY_DONE:
439     bciwarn << "Operation already done";
440   case DS_NO_ERR:
441     return false;
442   case DS_CANCEL_REQUEST:
443     bcierr << "Request is canceled"; break;
444   case DS_MMFILE_OVERRUN_READ_OFFSET:
445     bcierr << "Read pointer over run for memory mapped file."; break;
446   case DS_MMFILE_WRITE_APP_CLOSE:
447     bcierr << "Memory mapped file server is closed."; break;
448   case DS_MMFILE_RESETTING:
449     bcierr << "Resetting the memory mapped file."; break;
450   case DS_APPUDP_RR_TIMEOUT:
```

```
451       bcierr << "RR time out for application communication."; break;
452     case DS_DISCONNECT_SOCKET:
453       bcierr << "Socket disconnection."; break;
454     case DS_SELECT_DATA_SOURCE_CANCELED:
455       bcierr << "Selecting data source is canceled."; break;
456     case DS_FAIL_TO_OPEN_MMFILE:
457       bcierr << "Failed to open memory mapped file."; break;
458     case DS_RECEIVE_IRREGULAR_PACKET:
459       bcierr << "Received packet is invalid."; break;
460     case DS_WAVE_RR_TIMEOUT:
461       bcierr << "RR time out for collecting waveform."; break;
462     case DS_CMD_RR_TIMEOUT:
463       bcierr << "RR time out for commands."; break;
464     case DS_SEND_CMD_ERR_RETRY_MAX:
465       bcierr << "Retry error of sending commands."; break;
466     case DS_MMFILE_ERR_NEED_RESET:
467       bcierr << "Error which needs the reset of memory mapped file."; break;
468     case DS_MMFILE_ERR:
469       bcierr << "General error for memory mapped file."; break;
470     case DS_FAIL_TO_INIT_SEQUENCE:
471       bcierr << "Failed to initialize the sequence."; break;
472     case DS_FAIL_TO_READ_XML_SETTING_FILE:
473       bcierr << "Error in reading the setting xml file."; break;
474     case DS_FAIL_TO_DECIDE_DEST_IPADDRESS:
475       bcierr << "Could not find destination IP address."; break;
476     case DS_FAIL_TO_CONNECT_SOCKET:
477       bcierr << "Failed to connect socket."; break;
478     case DS_FAIL_TO_CREATE_SOCKET:
479       bcierr << "Failed to create socket."; break;
480     case DS_FAIL_TO_CREATE_SOCKET_INFO:
481       bcierr << "Failed to create destination socket information."; break;
482     case DS_FAIL_TO_CREATE_MMFILE:
483       bcierr << "Failed to carete memory mapped file."; break;
484     case DS_REQ_DIF_MODE:
485       bcierr << "Mode is invalid."; break;
486     case DS_NOT_CONNECTED:
487       bcierr << "Not connected."; break;
488     case DS_NOT_READY:
489       bcierr << "Not ready. "; break;
490     case DS_NO_MEMORY:
491       bcierr << "No enough memory."; break;
492     case DS_INVALID_PARAM:
493       bcierr << "Invalid parameters."; break;
494     case DS_ERR:
495     default:
496       bcierr << "General error."; break;
497     }
498     return true;
499 }
```

Griffin W. Milsap

929 N. Wolfe St, Apt 610-A, Baltimore MD 21205 — (608) 609-9729 — griffin.milsap@gmail.com

May 8, 2018

Education

2012– **Johns Hopkins University**
Ph.D Candidate in Biomedical Engineering
Expected Graduation: Sept. 2018

2008–2012 **B.Sc., Rensselaer Polytechnic University, *Cum Laude***
Electrical and Computer Systems Engineering
Overall GPA: 3.68/4.0

Professional/Research Experience

2012– **Ph.D Candidate in Biomedical Engineering**
Johns Hopkins University Whiting School of Engineering
- Collected/Analyzed inpatient/intraoperative electrocorticographic (ECoG) recordings
- Co-authored 10+ publications, co-authored a successful NIH R01 grant
- Coursework in neuroscience, statistics, machine learning, and instrumentation

2012–2013 BCI2000 Consultant
Pacific Development and Technology, LLC
- Providing advice and programming help with remote deployment of BCI2000 system

2008–2012 **Software and Test Engineer/Undergraduate Researcher**
Wadsworth Center, Schalk Brain Computer Interfacing (BCI) Lab
- Developed, maintained and tested the BCI2000 software, used by hundreds of labs world-wide
- Interfaced eyetracking, motionsensing, and signal acquisition hardware with BCI2000
- Implemented experimental paradigms, listed in several publications

2010–2012 **Principle Software Engineer/Partner**
Motalen Inc.
- Developed musical signal processing and procedural game engine architecture
- Designed, programmed, and tested one "shipped" game title on Android, "Wave"
- Deployed BCI2000 based system for internal BCI research and development

2010–2011 Undergraduate Researcher
Rensselaer Artificial Intelligence and Reasoning Laboratory
- Worked on audio signal-processing algorithms for a creative artifical intelligence agent in collaboration with machine learning scientists and musicians.

Professional Expertise

Research	Worked in collaborative research settings for a decade throughout undergraduate and Ph.D education under Dr. Nathan Crone (`https://cronelab.github.io`) and Dr. Gerwin Schalk (`https://schalklab.org`).
Clinical	Recorded research-quality human electrocorticography in a world-class clinical epilepsy monitoring unit, as well as intraoperatively during awake DBS surgeries while interacting directly with patients and clinicians. Familiar with surgical practices and HIPAA compliant data storage/de-identification procedures.
Computational	Extensive background writing real-time software packages to facilitate signal processing of audio and electrophysiolgy. Familiar with theory and practices for data mining and machine learning. Strong background in parallelization, realtime graphics, profiling, optimization, debugging, and net-code.
Writing	Authored articles, posters, technical communications, and patent applications. Co-authored a successful NIH R01 grant application, and familiar with IRB practices, grantsmanship, and peer-review procedures.
Engineering	Developed high performance real-time systems for neural signal processing, graphics, network-communication, and interactions with the outside-world via connected hardware and embedded systems. Strong background in software engineering, system design, computer architecture, networking, and reverse engineering.
Neuroscience	Made contributions to the state-of-the-art in cortical speech representations. Familiar with classical speech/language models, and ventral-stream image identification theories.

Technical Expertise

C/C++	13+ Years of Experience, Expert Level
Python	6+ Years of Experience, Expert Level
Java(Android)	4+ Years of Experience, Strong Background
Javascript	2+ Years of Experience, Full Stack
BCI2000	10+ Years of Experience, Active Contributor/Developer
OpenGL	12+ Years of Experience, Strong Background in Realtime Graphics
Matlab	Good Working Knowledge
Linux	6+ years sysadmin experience
Packages	Numpy, Scikit-Learn, Matplotlib, Qt, three.js, d3.js
Software	Jupyter, Unity, BCI2000, Chrome Developer Console, Visual Studio, Node.js, LaTeX, vim, git, svn, Blender, GIMP, Photoshop, Renoise, MS Office, Wireshark
Embedded	Arduino, Raspberry Pi, BeagleBone, TI MSP430, OpenBCI
Hardware	Written Realtime Software Interfaces with: Oculus/Vive, Tobii Eyetrackers, Nintendo Wii Remote, Clinical and Research EEG amplifiers, and audio interfaces.

Publications

Selected Articles

2018 Griffin Milsap, Max Collard, Chris Coogan, and Nathan Crone. BCI2000Web and WebFM: Browser-based Tools for Brain Computer Interfaces and Functional Brain Mapping. *IEEE Journal of Transactions on Biomedical Engineering*, (Feb 2018, Under Review)

Kiyohide Usami, Griffin W Milsap, Anna Korzeniewska, Maxwell J Collard, Yujing Wang, Ronald P Lesser, William S Anderson, and Nathan E Crone. Cortical Responses to Input From Distant Areas are Modulated by Local Spontaneous Alpha/Beta Oscillations. *Cerebral Cortex*, 2018

2017 Kyle Rupp, Matthew Roos, Griffin Milsap, Carlos Caceres, Christopher Ratto, Mark Chevillet, Nathan E Crone, and Michael Wolmetz. Semantic attributes are encoded in human electrocorticographic signals during visual object recognition. *NeuroImage*, 148:318–329, 2017

Carlos A Caceres, Matthew J Roos, Kyle M Rupp, Griffin Milsap, Nathan E Crone, Michael E Wolmetz, and Christopher R Ratto. Feature Selection Methods for Zero-Shot Learning of Neural Activity. *Frontiers in neuroinformatics*, 11:41, 2017

Ravindra Arya, J Adam Wilson, Hisako Fujiwara, Leonid Rozhkov, James L Leach, Anna W Byars, Hansel M Greiner, Jennifer Vannest, Jason Buroker, Griffin Milsap, and others. Presurgical language localization with visual naming associated ECoG high-gamma modulation in pediatric drug-resistant epilepsy. *Epilepsia*, 58(4):663–673, 2017

2016 Maxwell J Collard, Matthew S Fifer, Heather L Benz, David P McMullen, Yujing Wang, Griffin W Milsap, Anna Korzeniewska, and Nathan E Crone. Cortical subnetwork dynamics during human language tasks. *Neuroimage*, 135:261–272, 2016

2015 Vasileios G. Kanas, Iosif Mporas, Griffin W. Milsap, Kyriakos N. Sgarbas, Nathan E. Crone, and Anastasios Bezerianos. Time-Varying Parametric Modeling of ECoG for Syllable Decoding. In *Brain Informatics and Health*, number 9250 in Lecture Notes in Computer Science, pages 222–231. Springer International Publishing

Miaomiao Guo, Guizhi Xu, Lei Wang, Matthew Masters, Griffin Milsap, Nitish Thakor, and Alcimar Barbosa Soares. The anterior contralateral response improves performance in a single trial auditory oddball BMI. *Biomedical Signal Processing and Control*, 22:74–84, September 2015

2014 Nitish V Thakor, Matthew S Fifer, Guy Hotson, Heather L Benz, Geoffrey I Newman, Griffin W Milsap, and Nathan E Crone. Neuroprosthetic limb control with electrocorticography: approaches and challenges. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 5212–5215. IEEE, 2014

2013 Griffin Milsap, Matthew Fifer, Nathan Crone, and Nitish Thakor. Listening to the music of the brain: Live analysis of ECoG recordings using digital audio workstation software. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 682–685, 2013

M.S. Fifer, G.W. Milsap, E. Greenwald, D.P. McMullen, W.S. Anderson, N.V. Thakor, N.E. Crone, and R. Vinjamuri. Design and implementation of a human ECoG simulator for testing brain-machine interfaces. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1311–1314, November 2013

2011 Selmer Bringsjord, Colin Kuebler, Joshua Taylor, Griffin Milsap, Sean Austin, Jonas Braasch, Pauline Oliveros, Doug Van Nort, Adam Rosenkrantz, and Kasia Hayden. Creativity and conducting: handle in the CAIRA project. In *Proceedings of the 8th ACM conference on Creativity and cognition*, pages 319–320, New York, NY, USA, 2011. ACM

Posters

2017 P. Kudela, R. Oh, G. Milsap, N. E. Crone, and W. S. Anderson. A study of single-pulse cortical stimulation effects across cortical micro-domains. In *Society for Neuroscience Abstract*, page 291.03/G4, November 2017. Session: Session 291 - Signal Propagation; Session Type: Poster

G. W. Milsap, M. J. Collard, K. Rupp, M. J. Roos, C. Caceres, C. Ratto, M. Wolmetz, and N. E. Crone. Intrinsic neural spaces from human electrocorticography. In *Society for Neuroscience Abstract*, page 589.12/GG7, November 2017. Session: Session 589 - Representation of Objects and Scenes; Session Type: Poster

2014 M.J. Collard, M.S. Fifer, Y. Wang, H.L. Benz, G.W. Milsap, A Korzeniewska, N.V. Thakor, and N.E. Crone. Identifying functional subnetworks in human language tasks using electrocorticography, November 2014

G. Hotson, G. Milsap, D.P. McMullen, B.A. Wester, W.S. Anderson, J.W. Krakauer, N.E. Crone, and N. V. Thakor. Electrocorticographic decoding of high-level action goals following verbal instruction, November 2014

A Korzeniewska, S. Dalvin, G. Milsap, A. Flinker, and N.E. Crone. The impact of lexical retrieval on high-gamma effective connectivity in human language networks, November 2014

2013 Griffin Milsap, Matt Fifer, Nathan Crone, and Nitish Thakor. Listening to the Music of the Brain: Live Analysis of ECoG Recordings using Digital Audio Workstation Software, November 2013

P. Brunner, A. Gunduz, W.G. Coon, G. Milsap, A. L. Ritaccio, and G. Schalk. Toward real-time identification of selective auditory attention in a cocktail party using electrocorticographic signals (ECoG) in humans, November 2013

M.S. Fifer, G.W. Milsap, E. Greenwald, D.P. McMullen, W.S. Anderson, N.V. Thakor, N.E. Crone, and R. Vinjamuri. Design and implementation of a human ECoG simulator for testing brain-machine interfaces. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1311–1314, November 2013

2012 Griffin Milsap, Beth Werbaneth, Colin Neville, Catherine Mickey, Justin Renga, and Mukkai Krishnamoorthy. Reroot: A novel method of using mobile devices as human computer interfaces. Third Annual Undergraduate Research Symposium, Rensselaer Polytechnic Institute, 2012

2010 Griffin Milsap, Gerwin Schalk, and Lester Gerhardt. Brain computer interfacing software development. First Annual Undergraduate Research Symposium, Rensselaer Polytechnic Institute, 2010

Invited Talks and Teaching

2017 Griffin Milsap and Peter Brunner. National Center for Adaptive Neurotechnologies Summer Course 2017: BCI2000 Fundamentals and Interfacing, July 2017

2016 Griffin Milsap and Peter Brunner. National Center for Adaptive Neurotechnologies Summer Course 2016: BCI2000 Fundamentals and Interfacing, July 2016

2013 Griffin Milsap, Nancy Hanrahan, Don Yanaitis, and Law Blank. Leading Edge Initiatives in Games and Health, October 2013