Diss. ETH No. 20199

A REAL TIME BRAIN MACHINE INTERFACE FOR HAND GRASPING VIA SIGNALS FROM HIGHER ORDER CORTICAL AREAS

A dissertation submitted to the

ETH Zurich

for the degree of

Doctor of Sciences

presented by

ERK SUBAŞI

M.Sc. Computational Science, Koc University, Turkey
B.Sc. Electrical and Electronics Engineering, METU, Turkey
born May 29, 1980
citizen of Turkey

accepted on recommendation of

Prof. Dr. Rodney Douglas, examiner

Prof. Dr. Hansjörg Scherberger, co-examiner

Prof. Dr. Joachim Buhmann, co-examiner

Zürich, 2012

Abstract

Drawing on a wealth of knowledge about cortical movement processing, together with recent advances in signal processing and acquisition technology, the field of brain machine interfaces (BMIs) has the potential to become a viable assistive tool for patients with chronic spinal cord injury, stroke, and other motor debilitating diseases. Here, we present our efforts on development of a neural interface for decoding specific hand grasping postures in macaque monkeys. In contrast to the vast body of the literature, where M1 is used as the major source of motor related neural activity, our approach aims at decoding the neural activity in the anterior intraparietal cortex (AIP) and ventral premotor cortex (F5). These are higher-order motor planning areas, which play a role in sensorimotor integration during movement planning and believed to be holding an abstract representation of hand grasping actions. In this work, we formulated 3 main research questions and treated them subsequently;

First, as the major research question of this thesis, we investigated the feasibility of using multi-unit neural activity from AIP and F5 for decoding discrete hand postures in a real-time BMI setting with closed feedback. This approach builds upon the previous studies from our lab and targets conceptual proof of using neural signals from AIP and F5 in a real time setup with a clinical neuroprosthetic device ultimately in mind. Our initial analysis confirmed that real-time recorded signal characteristics have similar tuning properties to cells in previous single-unit recording studies. The maximum average decoding accuracy observed for two grasp types (power and precision grip) and five wrist orientations was 63% (chance level, 10%). Analysis of decoder performance showed that grip type decoding was highly accurate (90.6%), with most errors occurring during orientation classification. Furthermore, we observed significant differences in the contributions of F5 and AIP for grasp decoding, with F5 being better suited for classification of the grip type and AIP

contributing more toward decoding of object orientation. This work is published in Journal of Neuroscience in 2011 (Townsend et. al, 2011).

Second, using the same signal sources and modalities, we targeted decoding the temporal component of grasping, which has a fundamental importance in a realistic fully autonomous neuroprosthetic application. Employing initially the same analytical procedures from previous chapter we showed that, signals emerging from AIP and F5 are indeed usable for movement time decoding as well. Moving further we have utilized more sophisticated Markovian models to better capture stochastic and temporal structure of the underlying task and presented improved decoding accuracy and robustness. As the last attempt to draw a conclusion about maximum possible decoding accuracy, we defined our task in a data-mining setup and compared our results for different machine learning algorithms. Results have been presented in Neuroscience 2008, in Washington (Subasi et al., 2008).

Finally, similar to the last objective of temporal decoding, we took a data-mining approach for our initial task of decoding hand postures. Here, on our data set we systematically tested 24 different classification algorithms, which include standard machine learning algorithms and ensemble methods. Based on the observed results from different learner families, we have investigated the implementation of an improved learner for our problem. The proposed model showed better decoding accuracy and enhanced robustness on average compared to the learners utilized in the first part. However, at the end our benchmark learner, a Naïve Bayesian Classifier, was still showed to be one of the strongest learners for the task at hand. The outcome of this work was published and presented at an international IEEE conference (Subasi et al., 2010).

In sum, this thesis brings new insights into quantitative differences in the functional representation of grasp movements in AIP and F5 and represents a first step toward using these signals for developing functional neural interfaces for hand grasping.

Zusammenfassung

Das zunehmende Wissen über die Verarbeitung kortikaler Bewegungspläne und die neuesten Fortschritte in den Bereichen Signal- und Datenverarbeitung in Betracht ziehend, haben Gehirn-Maschine-Schnittstellen (engl. BMIs, "brain machine interfaces") das Potential, eine zuverlässige Unterstützung für Schlaganfallpatienten, Patienten mit chronischen Rückenmarksverletzungen, und anderen degenerativen Krankheiten des motorischen Systems zu werden. In der vorliegenden Arbeit berichten wir von unseren Fortschritten bei der Entwicklung einer neuronalen Schnittstelle zur Dekodierung spezifischer Handgreifbewegungen bei Makaken. Im Gegensatz zum Grossteil der Veröffentlichungen in diesem Forschungsbereich, in denen M1 als Hauptquelle neuronaler Signale zur motorischen Kontrolle verwendet wird, zielt unsere Ansatz darauf ab, neuronale Aktivität in den Arealen AIP (dem anterioren intraparietalen Cortex) und F5 (dem ventralen prämotorischen Cortex) zu dekodieren. Es handelt sich dabei um übergeordnete motorische Areale, die im Rahmen der Bewegungsplanung eine Rolle bei der senso-motorischen Integration spielen und als Orte der abstrakten Repräsentation von Handgreifbewegungen gelten. In dieser Forschungsarbeit wurden drei zentrale Fragestellungen nacheinander untersucht.

Erstens untersuchten wir, als wissenschaftliche Kernfrage dieser Doktorarbeit, wie realistisch die Nutzung von neuronaler "Multi-Unit"-Aktivität der Areale AIP und F5 ist, um unterschiedliche Handbewegungen in einem Echtzeit-BMI-Setup mit geschlossener Rückkoppelungsschleife zu dekodieren. Dieser Ansatz baut auf vorhergehende Arbeiten unseres Labors auf und zielt darauf ab, einen konzeptionellen Beweis dafür zu liefern, dass neuronale Signale der Areale AIP und F5 in einem Echtzeit-Versuchsaufbau zur Bewegungs-Dekodierung benutzt warden koennen, nicht zuletzt hinsichtlich einer klinischen Umsetzung. Erste Analyse Ergebnisse bestätigten, dass die in Echtzeit aufgenommenen Signale in ihren Kodierungs-Eigenschaften vergleichbar mit den aus vorhergehenden Studien mit Einzelzellableitungen gewonnenen Daten waren. Die maximale 6

Durchschnittsgenauigkeit für die Dekodierung zweier verschiedener Greifbewegungen (Kraft- und Präzisions-Griff) in Kombination mit fünf unterschiedlichen Ausrichtungen des Handgelenks lag bei 63% (bei einer Zufallswahrscheinlichkeit von 10%). Die Analyse der Leistungsfähigkeit des Decoders ergab eine hohe Genauigkeit bei der Dekodierung des Griff-Typs (90.6%), wohingegen die meisten Fehler bei der Unterscheidung der Orientierung des Handgelenks auftraten. Desweiteren beobachteten wir erhebliche Beitragsunterschiede von F5 und AIP zur Dekodierung der Greifbewegung, wobei die neuronale Aktivität in F5 besser für die Dekodierung des Griff-Typs geeignet war, jene in AIP dagegen mehr zur Dekodierung der Ausrichtung des zu greifenden Objekts beitrug. Diese Studie wurde 2011 im Journal of Neuroscience veröffentlicht (Townsend et al, 2011).

Zweitens zielten wir, unter Nutzung derselben experimentellen Daten, auf die Dekodierung der zeitlichen Komponente von Greifbewegungen ab. Dieser kommt eine zentrale Bedeutung für die praktische Umsetzung einer autonom funktionierenden neuroprosthetischen Anwendung zu. Unter Verwendung derselben im Vorfeld gebrauchten analytischen Methoden zeigten wir, dass die neuronalen Signale der Areale AIP und F5 tatsächlich für die Dekodierung von zeitlichen Komponenten verwendet werden können. Im weiteren Verlauf benutzten wir anspruchsvollere Markov-Modelle, um stochastische Prozesse und die zeitliche Struktur des Versuchs detaillierter erfassen zu können, was zu einer verbesserten Genauigkeit und Verlässlichkeit der Dekodierung führte. Um ein abschliessendes Fazit zur höchstmöglichen Genauigkeit der Dekodierung zu ziehen, verglichen wir die Ergebnisse bei Benutzung unterschiedlicher Lern-Algorithmen miteinander. Die Ergebnisse wurden im Jahr 2008 im Rahmen der “Neuroscience” in Washington(D.C.) präsentiert (Subasi et al., 2008).

Schliesslich wählten wir, ähnlich der Analyse der zeitlichen Dekodierung, einen “Data-Mining”-Ansatz, um die ursprüngliche Frage nach der Dekodierung von Handgreifbewegungen zu beantworten. Hierbei untersuchten wir 24 unterschiedliche Klassifizierungs-Algorithmen, die standardisierte Lern-Algorithmen und “Ensemble

Methoden" beinhalteten. Basierend auf den beobachteten Ergebnissen bei Nutzung unterschiedlicher Familien von Lern-Algorithmen schlugen wir ein verbessertes Modell für unsere Fragestellung vor. Dieser Lern-Algorithmus zeigte im Schnitt eine verbesserte Dekodierungs-Genauigkeit und Zuverlässigkeit. Jedoch erwies sich am Schluss unser Referenz-Algorithmus, ein naiver Bayes-Klassifikator, als eines der besten Lern-Modelle für den gegebenen Task. Das Ergebnis dieser Arbeit wurde im Rahmen einer internationalen IEEE-Konferenz veröffentlicht und präsentiert (Subasi et al., 2010).

Zusammengefasst gewährt die vorliegende Dissertation neue Einblicke zu quantitative Unterschieden der funktionellen Repräsentation von Handgreifbewegungen in AIP und F5 und zeigt erste Schritte auf, wie diese Signale zur Entwicklung neuronaler Schnittstellen für Handgreifbewegungen genutzt warden könnten.

Acknowledgements

First of all, I would like to thank my supervisor Hansjörg Scherberger not only just for making this work possible and providing the opportunity to study in such an intellectually rich environment but also for his continued support on various levels of my doctoral student life. Without my PostDoc and research partner Benjamin Townsend this work would be not possible to accomplish. The friendship of him and the other members of Grasping group at INI; Sebastian Lehmann, Markus Baumann, Marie Christine Fluet, were invaluable to me during those years. I always felt proud and lucky of being a part of Team-B!

I also would like to express my gratitude to the other members of my examining committee; Rodney Douglas and Joachim Buhmann, for their comments, encouragement and interest in my research.

I am grateful to all my friends and colleagues at the Institute of Neuroinformatics, University Zurich and ETH Zurich and especially to Kevan Martin and Rodney Douglas for creating such a unique and inspiring atmosphere. Special thanks go to my friends Emre Neftci, Jacqueline Baumann, Ufuk Olgac, Emre Sarigol and Barcin family for being around whenever I need them.

I would also like to thank my family, Metin, Can and Efe Subasi for their support and encouragement I received along my academic and non-academic life. Finally, I am grateful to Deniz Ural, my best friend, companion and fiancée, who was always on my side with her unwawering love and support throughout these years.

Disclaimer

All experiments involving monkeys were in accordance with the guidelines for the care and use of mammals in neuroscience and behavioral research (National Research Council, 2003) and approved by the cantonal Veterinary Office of Zurich.

Training of the monkeys was performed by Benjamin Townsend, with the help of Bernadette Disler and Gabi Stichel.

Building the setup for doing electrophysiology in behaving monkeys and in particular for the delayed grasping task that was used in the current thesis, was a joint effort of Hansjörg Scherberger, Benjamin Townsend, Markus Baumann, Marie-Christine Fluet and myself.

The software for the control of the behavioral setup and the acquisition of behavioral data was written by Benjamin Townsend and Markus Baumann in LabView. All other software, CerebusSimulator, CerebusDecoder and analysis routines were developed by myself.

The surgeries necessary to prepare for the recordings were performed by Hansjoerg Scherberger, with Markus Baumann and Benjamin Townsend assisting.

All data presented in this thesis were collected and analyzed by Benjamin Townsend and myself.

Contents

List of figures

List of tables

1 Introduction

"Except our own thoughts, there is nothing absolutely in our power."

-René Descartes

1.1 Overview

The human brain is believed to be the most complex computing machine in existence. It essentially captures a variety of environmental signals and extracts information from these multivariate signal streams to create cognition, imagery and different forms of behavior. Its computational capabilities are arising arguably from its unique architecture. Around 100 billion neurons work in parallel, with a massively distributed memory system consisting of over 100 trillion synapses. It manages to cope with virtually unlimited amount of complexity while being very robust to both external and internal noise and it operates even after some significant loss of its computation units. Numerous philosophers and scientists throughout the history attempted to explain the working mechanism and organizing principles of human brain, yet, without too much success. Under this lack of knowledge of governing principles, scientists and engineers have focused more on developing methods to directly interface with the brain in the last decades, mainly to help people with nervous system problems but also with a hope to gain a better understanding of the underlying principles of its operation. With advances in measurement and computing technologies, achieving those goals is becoming an increasingly realistic prospect nowadays. In this work, we document our efforts at the Institute of Neuroinformatics (University of Zurich and ETH Zurich) to contribute in the field via implementation of a model real-time Brain Machine Interface for hand grasping.

A brain-machine interface (BMI) is a communication system which collects some quantifiable form of neural signals, infers meaning (or decodes them in BMI terminology) and finally creates a mapping to a corresponding behavior or action. One of the primary motivations for developing BMIs has been to provide a means of communication for individuals with severe neurological problems, where the neural pathways for transmitting control signals to motor organs are damaged. In many of these patients, the cerebral activity creating these signals is still intact. Therefore, the decoding of this neural activity directly from the brain could be used to bypass their malfunctioning peripheral nervous system. It is possible to come across to BMI research in the literature with a variety of names; brain computer interfaces, direct neural interfaces, neurocortical interfaces, or neuroprosthetic devices are among the most common alternatives. They usually have some subtle differences and selection of a particular name is usually motivated by the specific application type. But the characteristics and working principles of all of these devices are the same; reading / understanding and bypassing natural nervous systems to create a direct communication pathway between brain and external world.

BMIs can be classified in two families in broadest sense depending on the source and method the signals are collected: Invasive methods vs. non-invasive ones. Invasive systems utilize implanted single or multi electrodes directly into central nervous system. They are well suited for decoding activity in the cerebral cortex due to high signal-to-noise ratio (SNR). Although brain represents information in a distributed fashion in general, cortical areas still show significant specialization and it is possible to target these specialized areas with invasive methods. Noninvasive systems on the other hand, are better suited for situations in which a surgical implementation is not possible or may be avoided due to the obvious risk of a surgical procedure. Electroencephalography (EEG) is the most commonly used measurement modality for noninvasive recordings. The challenge with EEG and with non-invasive methods in general, is typically a low SNR and low information throughput. The current state of art signal collection and decoding techniques of noninvasive methods are beyond the reach of the goal of decoding natural hand movements, thus we are concentrating only on invasive approaches throughout this work.

One can track the research that makes the concept of BMIs possible to the early 1900s. By the beginning of previous century Fedor Krause, a German neuro-surgeon, was able to do a systematic electrical mapping of the human brain, using conscious patients undergoing brain surgery (Morgan, 1982). Later, Hans Berger (Berger, 1929) discovered the externally

measurable electrical activity of human brain, which is the ancestor of modern EEG devices. Later at 50's, prominent Nobel laureates; Hodgkin, Huxley (Hodgkin et. al, 1952), enabled a breakthrough in membrane and intracellular electrophysiology, which is a key element of today's invasive BMIs. The same decade Hubel and Wiesel (Hubel et. al., 1959), other Nobel laureates, made significant contribution to our understanding of information processing in the visual system. Later on, Eberhard E. Fetz's famous publication (Fetz, 1969) on operant conditioning of cortical activity described the path for many modern BMI experiments. In the 1980s, Georgopoulos (Georgopoulos et. al., 1982) recorded single-neuron firing rates as the monkey reached in different directions. In 90s we have witnessed minitiarisations, advances in signal processing, computation power, and new algorithms have led to modern BMIs. There is an exponential growth in the research field in the last 15 years and finally in late 90s and in 2000s, first successful human trials for cortical motor prostheses started and promising results were obtained (Hochberg et. al., 2006). Cortical motor prosthesis are still in early research and development phase, whereas other forms of BMIs like cochlear implants or deep-brain stimulation devices (for symptomatic treatment of Parkinson's disease or chronic pain) already found their way in the daily lives of thousands of the patients in the last few decades (Loeb, 1990; Merzenich, 1983; Follett, 2000).

In this thesis, we aim to present our research on developing a simple brain machine interface for hand grasping in macaques that can distinguish between various grip types and wrist orientations in real time. With this, we hope to contribute to ongoing research on neuro-cortical prostheses which aims to make the lives of patients suffering from neuro-motor diseases easier. Using predominantly multi-unit activity recorded simultaneously from AIP and F5 (higher order grasping related cortical areas in non-human primates), we demonstrate that real time decoding of hand grasping is possible. This conceptual proof is first in the literature (Townsend et al., 2011) and is the main achievement in this work. Below we provide a description of the anatomical properties of the hand-grasping related cortical areas under interest first. In the parietal cortex this is the anterior intraparietal area (AIP) and in the frontal lobe the area F5. Then, a brief literature survey on hand-grasping neural prosthetic BMI research will follow. Finally we will finish this chapter with the descriptions of the main objectives of this work.

1.2 Higher order hand-grasping related cortical areas

One of the defining characteristics of this study is the brain regions we have utilized to collect neural signals for the BMI implemented. Throughout this work we have collected and analyzed data from only two higher order cortical regions, namely AIP and F5. By the term "higher order" we want to emphasize the fact that the areas under interest don't have direct motor output projections but relay predominantly to the primary motor cortex and furthermore are characterized by the presence of planning activity. Most of the relevant studies to date show that it is feasible to operate a hand/arm BMI via the signals collected from primary motor cortex (M1). By demonstrating the possibility of the utilization of the signals from these higher order / abstract grasping related regions, we aim to contribute to the literature. Below we provide a summary of the characteristics of these two cortical areas.

1.2.1 AIP

The anatomical connections locate AIP right at the interface between sensory, motor and cognitive areas related to hand movement control. It receives visual input via several higher order visual areas in the parietal cortex, in particular LIP, V6A and CIP (Nakamura et al., 2001; Borra et al., 2008; Gamberini et al., 2009). It is also connected with ventral visual stream areas in the temporal cortex (areas TEO, TEa, TEp, Borra et al., 2008). These connections might inform AIP about parameters of familiar and identified objects. In the frontal lobe, AIP is strongly and reciprocally connected with area F5 (Luppino et al., 1999 ;Borra et al., 2008), an area which exhibits similar activity related to hand movements (Rizzolatti et al., 1988; Murata et al., 1997; Raos et al., 2006; Stark et al., 2007) and is considered to be part of the cortical output structures for controlling the hand due to its projections to primary motor cortex and the spinal cord (Rizzolatti et al., 1988; Luppino et al., 1999; Lemon, 2008). AIP is also directly connected with the prefrontal cortex (areas 12 and 46, Borra et al., 2008), areas which are believed to be involved in higher order cognitive processing like working memory, rule learning or the representation of abstract concepts.

Mountcastle and his colleagues (Mountcastle et al., 1975) were the first who described neurons in the inferior parietal lobe which were involved in goal directed, visually guided hand movements. Later, Taira and colleagues found neurons with similar characteristics in the rostral part of the lateral bank of the intra-parietal sulcus (Taira et al., 1990), which was named the anterior intraparietal area (AIP, Gallese et al., 1994). Most of the task related

neurons in this area were selectively active for the grasping of different objects used in the experiment. Most importantly, they found that these neurons were not influenced by the position of the object in space, indicating that their activity was specifically related to the hand, not the arm movement. In a later study, Sakata and colleagues classified these neurons further as object type vs. non-object type neurons (Sakata et al., 1995), according to their behavior during mere fixation of the graspable object (Figure 1.1). Object type cells were found to represent aspects of the 3D shape of the graspable objects and some were also selective for the size and/or the orientation (Murata et al., 2000). Many of these object type cells preferred the same object during object fixation and during grasp execution. Motor-dominant or non-object type selective cells, on the other hand, were found to be more related to the shape of the hand during grasping. In another study, AIP neurons were found to show sustained activity in a delayed grasping task, suggesting that they play a role in the visual memory of 3-dimensional object features (Murata and Kitahara, 1996). Altogether, these electrophysiological findings suggest that AIP neurons play an important role in matching the pattern of the hand movement to visuo-spatial characteristics of the object to be grasped. The functional relevance of AIP during visually guided hand movements was also tested by inactivation studies in monkeys (Gallese et al., 1994) by utilization of microinjections of muscimol (a GABA agonist) which leads to localized and reversible inactivation of AIP. Finally, these properties were also confirmed by the previous single-electrode recording studies carried out in our lab (Baumann et al., 2009) which also demonstrated the context specific nature of the grasp planning signals in these areas.

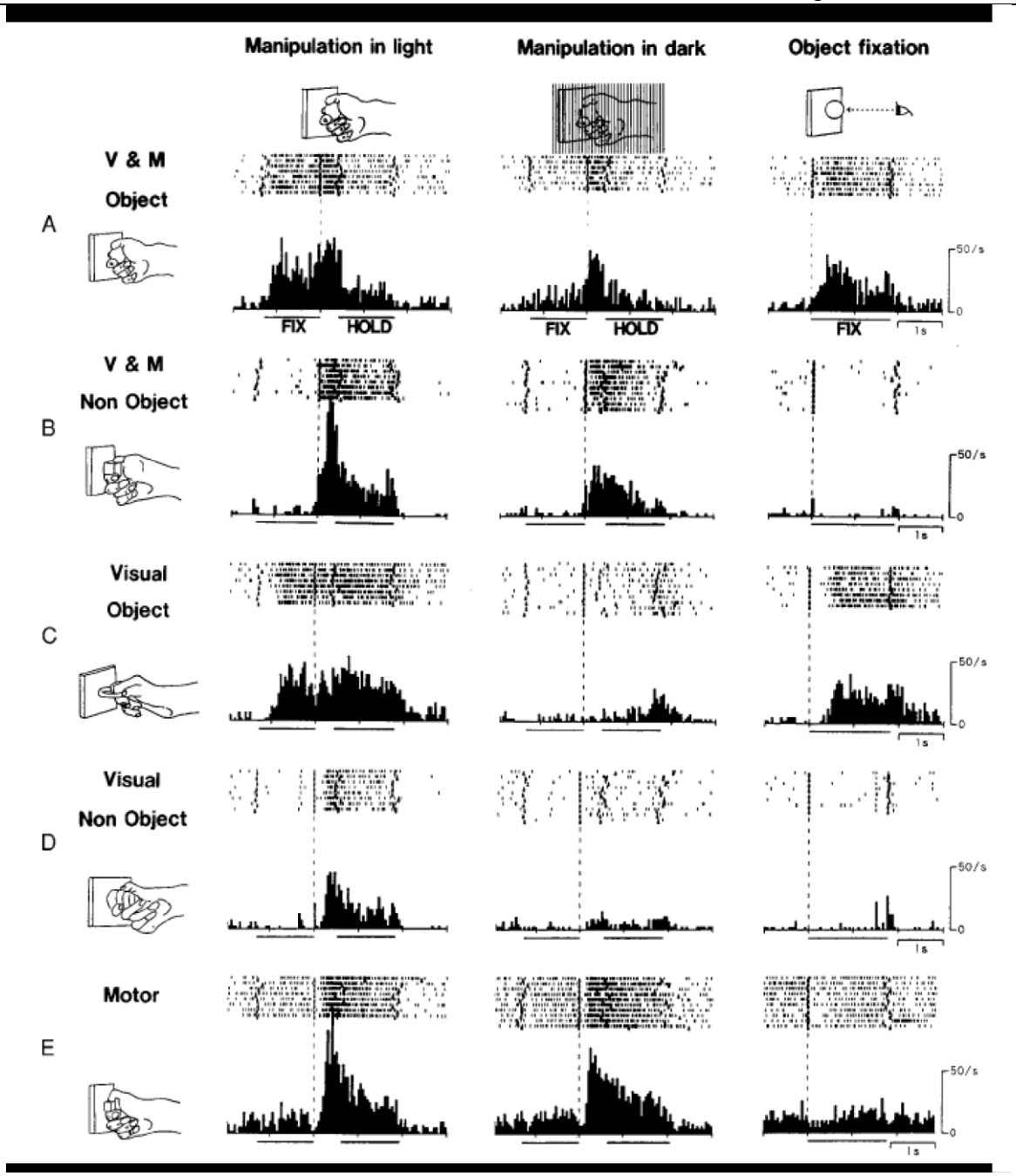

Figure 1-1 : Classification of AIP cells according to Sakata
Five types of hand manipulation related neurons under three task conditions. **A.** Example of object-type visual-motor neuron, **B.** nonobject-type visual-motor neuron, **C.** object-type visual dominant cell, **D.** nonobject-type visual dominant cell, **E.** motor dominant cell. The lines below the histograms show the mean duration of the fixation period and the holding period. Modified from Murata et al., 2000.

1.2.2 Frontal area F5

In macaques, the posterior bank of the arcuate sulcus with the adjacent cortex on the convexity is termed area F5. In their seminal paper, Rizzolatti and colleagues reported that neurons in F5 showed activity that was not specific to movements of individual joints of fingers like in the case with M1 neurons but to more abstract motor actions like grasping in different shapes or holding (Rizzolatti et al., 1988). Many of the neurons analyzed in this region were selectively tuned for a particular grip type, like precision grip, side grip or power grip. Also, many of these neurons could be activated by the visual presentation of the graspable objects. In later studies, F5 neurons were recorded while monkeys grasped repetitively one of six different objects either in the light or in the dark (Murata et al., 1997; Raos et al., 2006). The results showed that activity in F5 is very similar for objects with different geometric shapes when they are grasped with the same grip type, suggesting that the activity of F5 neurons is mainly determined by the type of grip that the animals use and not by the object shape itself. Moreover, some subset of neurons was activated also during mere object fixation in the absence of grasping. Fogassi and colleagues (Fogassi et al., 2001) tested the functional relevance of F5 for visuomotor transformation through an inactivation study. They showed that usual pre-shaping of the hand during the reaching was significantly impaired and kinematic differences compared to the normal case emerged. On the other hand, the general hand movements were still executable, especially after a series of corrections made with tactile feedback. Thus the authors concluded that F5 lesions affected specifically the visuomotor component of grasping movements. These symptoms after inactivation were quite similar to those after inactivation of AIP. This is consistent with the hypothesis that AIP and F5 together form a parieto-frontal circuit for sensorimotor transformations specific for hand grasping.

Furthermore, in a recent study Umilta and colleagues, by simultaneously recording in F5 and M1, showed that only F5 neurons were tuned for specific grasps before the movement start (Umilta et al., 2007). M1 neurons on the other hand lacked this early pre-movement specificity but where strongly involved during movement execution. This particular characteristic of F5 neurons makes them especially a good candidate for real-time BMIs, as it can serve as an additional signal source to more classical BMIs that utilize signals from M1.

1.3 Hand-grasping BMIs with neuro-prosthetic applications

The development of neural prostheses to restore voluntary movements in paralyzed patients is enjoying an increasing pace in attention lately. Such devices harness neural signals from intact brain areas to manipulate artificial devices, and ultimately could control the patient's own limbs (Hatsopoulos and Donoghue, 2009). Since our hands play a central role for interacting with the world (Lemon, 1993), improvement of hand function remains a high priority for patients with motor deficits, e.g., amputees, spinal cord injury patients, stroke victims, and others (Snoek et al., 2004; Anderson, 2009). Neural prostheses for grasping could greatly improve their quality of life.

Recent years have seen a multitude of studies on BMIs for movement control (Schwartz et al., 2006; Scherberger, 2009; Hatsopoulos and Donoghue, 2009). Besides EEG- and electrocorticographic-based systems in humans (Leuthardt et al., 2004; Wolpaw and McFarland, 2004; Bai et al., 2008), invasive BMIs in non-human primates have been developed using neural population activity in primary motor cortex (M1) to reconstruct continuous 2D and 3D arm and hand position (Wessberg et al., 2000; Serruya et al., 2002; Taylor et al., 2002; Carmena et al., 2003), and monkeys have learned to use these signals to control a gripper-equipped robotic arm to feed themselves (Velliste et al., 2008). This approach has generally not yet been extended to decode sophisticated grasping patterns, which is attributable to the complex nature of dexterous finger movements and the large number of degrees of freedom of the hand (Schieber and Santello, 2004), however see Vargas-Irwin et al. (2010) for a first example of such an approach. Furthermore, the exact mechanisms by which grasping movements are learned and retrieved are quite unclear, making effective decoding hard.

Alternatively, cognitive control signals related to intended actions can be extracted by tapping into "higher order" planning signals in premotor and parietal cortex (Musallam et al., 2004; Santhanam et al., 2006; Mulliken et al., 2008; Andersen et al., 2010). Key areas for such high-level control of grasping are ventral premotor cortex (area F5) and anterior intraparietal cortex (AIP), which are strongly and reciprocally connected (Luppino et al., 1999), establishing a fronto-parietal network dedicated to transforming visual signals into hand grasping instructions (Jeannerod et al., 1984; Kakei et al., 1999; Brochier and Umilta, 2007). Unlike M1, these areas represent upcoming hand movements at a conceptual or categorical level well before their execution (Musallam et al., 2004; Baumann et al., 2009; Fluet et al., 2010). Targeting these areas could therefore considerably simplify the decoding

of complex movements. The proof of the concept of a successful real-time BMI using only these types of abstract signals was not available in the literature to our knowledge, in this work we aim to contribute to the field in this respect.

1.4 Main objectives of the present thesis

A 1995 study by U.S. Department of Health Human Services suggests that 1.7 million people, only in USA, are suffering from some form of paralysis. This number is a lot higher, over 5.5 million, according to a more recent study sponsored by the Christopher and Dana Reeve Foundation (Paddock, 2009). Paralysis can result from spinal cord lesions and other traumatic accidents, peripheral neuropathies, amyotrophic lateral sclerosis, multiple sclerosis and stroke (Andersen, 2009). Another 1.4 million patients have motor disabilities due to limb amputation according to the same U.S. Department of Health Human Services survey. Many of these patients still have sufficiently intact cortex activation to plan movements, but they are unable to communicate these signals to their limbs and execute them. If we add other patients from all around the world, where we lack reliable statistics, it becomes clear that any assisting technology for restoring some upper-limb functionality holds significant promise for direct improvement in quality of life for millions of people.

In the last decade, we have witnessed considerable progress in this multidisciplinary research area, mainly due to availability of better recording technology and easier access to powerful computation. However, still considerable number of problems need to be tackled before fully functional neuroprosthetic brain-machine interfaces that can be utilized clinically in a broader sense. Among the existing major problems are; improving the quality of neuronal recordings, to achieve robust and long-term performance, extending the brain-machine interface approach to sensory functions and arguably most importantly having a better understanding about the operations of underlying brain regions, so we can attack the diseases more effectively. In this work, we aim to contribute to the field mainly on the last point by showing the feasibility of utilization of two higher order motor-cortical areas for a neural-prosthetic application. We will also report our attempts to improve the utilization of the signals coming from these brain regions via different learning algorithms.

To that extent, we will show the development of a simple decoder for hand grasping in macaques that can distinguish various grip types and wrist orientations in real time. Using predominantly multi-unit activity recorded simultaneously from AIP and F5, we will demonstrate real time decoding of grasp type by maximum likelihood estimation and off-line

decoding with various learning algorithms as well as decoding of grasp timing. Overall, these results represent a first step towards the development of a motor prosthesis for dexterous grasping movements in paralyzed patients utilizing data from AIP and F5.

The main hypothesis and specific goals:

The decoding analysis of neural signals typically involves two layers: the encoding and the decoding stages. In the encoding stage, neural signals are characterized as a function of the biological signal. In the decoding stage, the relation is inverted, and the signal is estimated from the spiking activity of the neurons. Our main hypothesis is built upon the assumption that abstract hand postures are held in two areas we are recording from, AIP and F5, and that this abstract information can be extracted using specialized machine learning and signal processing techniques, which was trained with the neural data captured by ~100 electrodes from these regions. During decoding, the system makes predictions based on the underlying probabilistic model and captured stochastic properties of the model during training. We have three specific goals which will guide us throughout the process and we will present those in 3 separate chapters after the methods chapter following this one.

i. Online decoding of power and precision grip :

As the first goal of the project, in chapter 3, we will decode in real-time a few specific hand movements (power and precision grips in 5 different orientations) from the neural population activity in AIP and F5 in the delayed grasping task (details in methods). To do this, we will simply use the rate information from each firing neuron for 10 different task conditions and use likelihood methods. The decoded movement intention will be presented in a static-image form back to the animal and we will report a thorough investigation of the characteristics of this closed-loop feedback on-line decoding experiment where we will be able to study the effects of visual feedback during sensorimotor transformation for grasping. With this real-time decoding based on multiunit signals, we will find the chance to validate some observations and conclusions drawn in previous single-unit recording studies from our lab (Baumann, 2009; Fluet, 2010). Furthermore, we will investigate the contribution of different regions to decoding performance for grip type and orientation. In a subsequent analysis, we will also study an optimized spike-sorting method. Overall, these results will highlight quantitative differences in the functional representation of grasp movements in AIP and F5

and represent a first step toward using these signals for developing functional neural interfaces for hand grasping and will be the main contribution of our work to the literature.

ii. Decoding of movement start :

For any practical usage of a brain-machine interface (BMI) for hand grasping, it is important to decode not only the intended movement accurately, but also the time when it should happen. Thus, in chapter 4, we will explore a time decoding task, as our second goal in this thesis. We will simply perform the grasp decoding task as in chapter 3. However, instead of using the Go command of the task control computer as the signals to move, we will decode the start of the movement from the neural activity in AIP and F5. For this, we will implement a decoding algorithm that will continuously interpret the spiking activity of a most recent fixed window of data and from that, we will estimate the behavioral state of the animal (baseline, planning, or movement execution). Based on previous research in our lab on the prediction of behavioral states from LFP activity, we know that there is informative data regarding brain states in AIP and F5 (Baumann, 2009). However, this work will utilize only spiking data to do such state decoding and not only improve our understanding about two brain regions but also provide a proxy about the feasibility of the usage of AIP and F5 for a realistic neural-prosthetic device. Overall, the online decoding of different neuronal states will be an important milestone, which will enable future BMIs, to differentiate between action planning and execution directly from the neural activity.

iii. Improved decoding :

Finally, in chapter 5, we will report our findings on the off-line analysis of decoding the grasp type and orientation once again, but this time we will utilize a data-mining approach and search for an improved learner for decoding the spatial components of our task. Using the decoder from chapter 3 as our benchmark, we will compare 24 learners, which include many standard machine learning algorithms and ensemble methods, to investigate the effectiveness of the state of art learners and to find a more comprehensive representation of the underlying signals. Finally, we will propose a learning model for obtaining more robust and accurate decoding for the data recorded. These experiments, all together, should bring necessary and critical further steps toward future prosthetic BMIs that can employ signals from higher order cortical areas for hand grasping neural-prosthetic devices.

2 General Methods

2.1 Basic Procedures

Hand grasping movements were decoded in real time using neural activity recorded simultaneously from area F5 and area AIP in two female rhesus macaque monkeys (animals Z and S, weights 6.5kg and 8.0kg respectively). All procedures and animal care were in accordance with guidelines set by the Veterinary Office of the Canton of Zurich and the Guidelines for the care and use of mammals in neuroscience and behavioral research (National Research Council, 2003).

The experimental paradigm we have used in this work is named "delayed grasping task". This behavioral task has been developed and employed previously in earlier studies in our lab (Bauman 2009, Fluet 2010). For this work we have extended the original task where necessary. In this task, the animals were seated in a primate chair and trained to grasp a handle with their right hand (Fig. 2.1). This handle was placed in front of the monkey at chest level, at a distance of approximately 30cm, and could be grasped either with a power grip (opposition of fingers and palm)(Fig 2.2B) or precision grip (opposition of index finger and thumb) (Fig 2.2C). Two clearly visible recessions on either side of the handle contained touch sensors which were used to detect contact of thumb and forefinger during

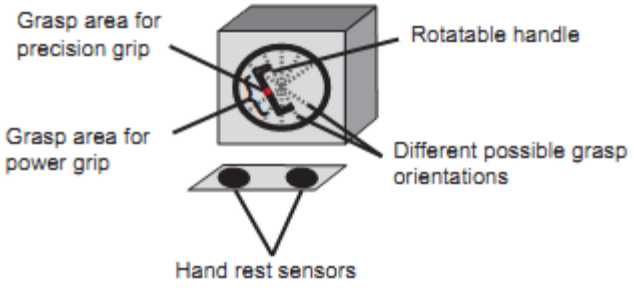

Figure 2-1: Grasping Box
The handle which can be oriented in 5 different angles, contained touch sensors which were used to detect contact of thumb and forefinger during precision grips, while power grips were detected using an infrared light barrier inside the handle aperture. Two capacitive touch sensors were functioned as hand-rest buttons.

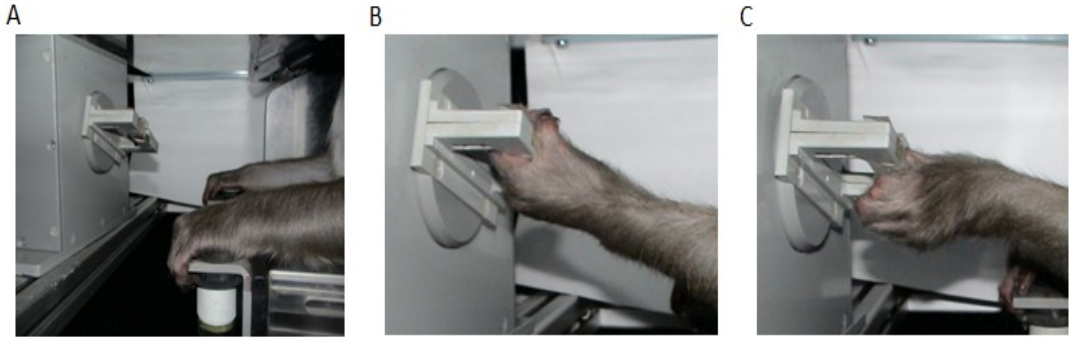

Figure 2-2: Hand-rest position and two grasping types animals are habituated to execute during the experiment
Animals were trained to sit still in setup in hand-rest position and to perform two grasping types when signaled. **A.** Hand-rest position. **B.** Animal performing power grip. **C.** Animal performing precision grip.

precision grips, while power grips were detected using an infrared light barrier inside the handle aperture. The monkey was instructed which grip type to make by means of two colored LED-like patterns projected from a LCD screen onto the centre of the handle via a half-mirror positioned between the animal's eyes and the target. While previous studies from the same laboratory used standard LED components, the current study required positioning of a digital display to present visual feedback during real time decoding, making LED placement impractical. Instead, cues were

presented to the animal by means of small colored dots shown on the screen. Hereafter these will be referred to as "light dots". The handle could be rotated into one of 5 discrete orientations (upright, 25° and 50° to the left and right), and was illuminated by two spotlights placed on either side. Apart from these light sources, the experimental room was completely dark. In addition, two capacitive touch sensors (Model EC3016NPAPL, Carlo Gavazzi, Italy) were placed at the level of the animals' waist, and functioned as hand-rest buttons (Fig. 2.2A). The behavioral task was controlled by means of custom-written software implemented in LabView Realtime (National Instruments, Austin, TX, USA). Also, an infrared camera was used to monitor the monkeys' behavior continuously throughout the entire experiment.

The analog neural signals extracted from two cortical regions are first amplified (for signal preservation) in the proximity of the animal and then sent via UDP protocol to a signal processor for further distribution to recording and decoding PCs.

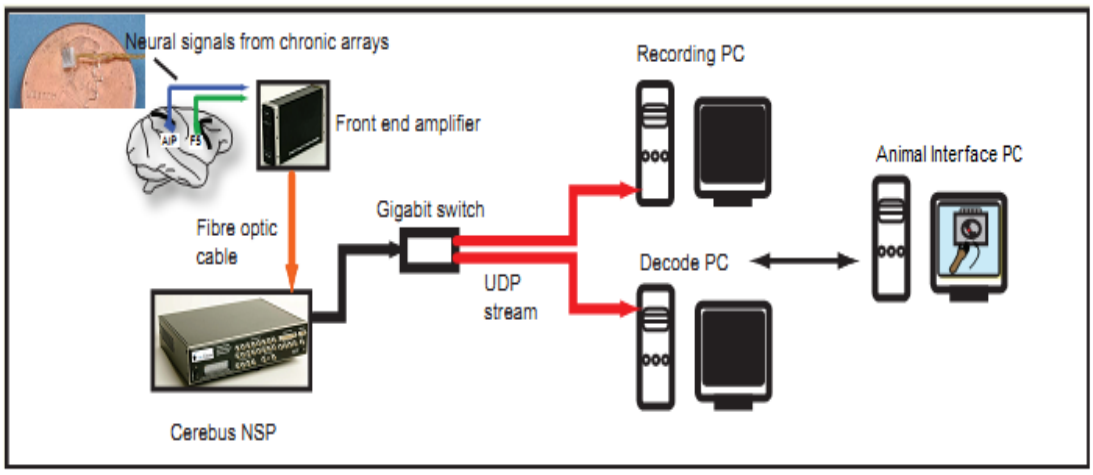

Figure 2-3: Recording setup network diagram
Neural signals from floating micro-electrode arrays are sampled using Cyberkinetics Neural Signal Processor and streamed to recording and decoding PCs. Animal interface PC is controlled via LabView software and decoding PC.

2.2 Behavioral Paradigms

In the *delayed grasping task*, the monkey was required to grasp the handle in one of 5 orientations with either a power grip or a precision grip. This gave a total of 10 different grasp conditions that were presented on a trial-by-trial basis in pseudo-random order. The animal began a trial by placing each hand on a hand-rest button while sitting in darkness. In the *baseline period,* a red dot was illuminated and the handle positioned in one of the five orientations. From this point on the animal had to keep both hands at rest for a variable period of time (700-1100 ms, mean: 900 ms). In the following *cue period (duration: 600ms),* the object was illuminated to reveal its orientation and an additional dot was presented adjacent to the red dot,

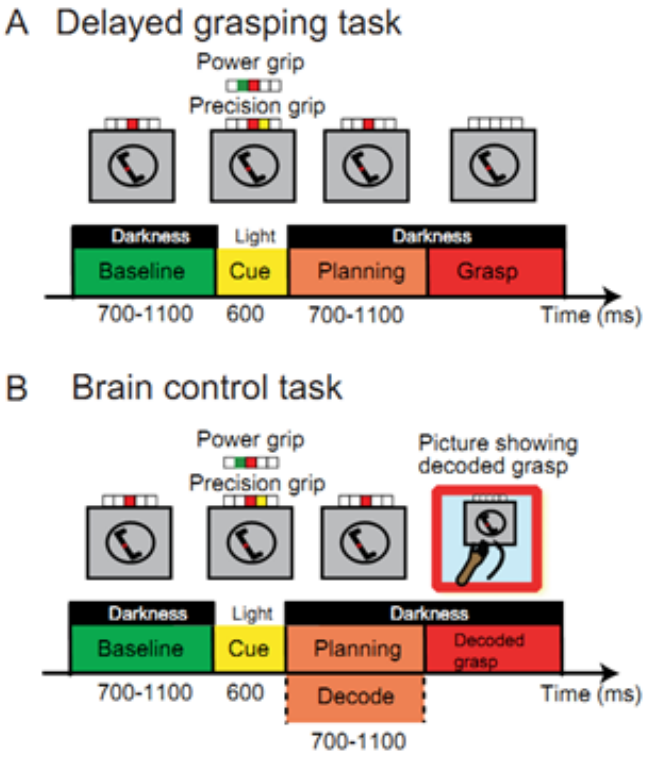

Figure 2-4: Task paradigm
Animals were trained to perform two tasks. **A.** Delayed grasping task, consisting of four epochs: baseline, cue, planning, and movement. The task was performed in the dark, except for the cue period when the handle was visible together with an instruction dot for grasp type. **B.** Brain control task. This task proceeded as in A, except at the end of the planning epoch, where the planned grasp was decoded and visually fed back to the monkey (picture of grasp) without requiring the animal to actually execute the movement.

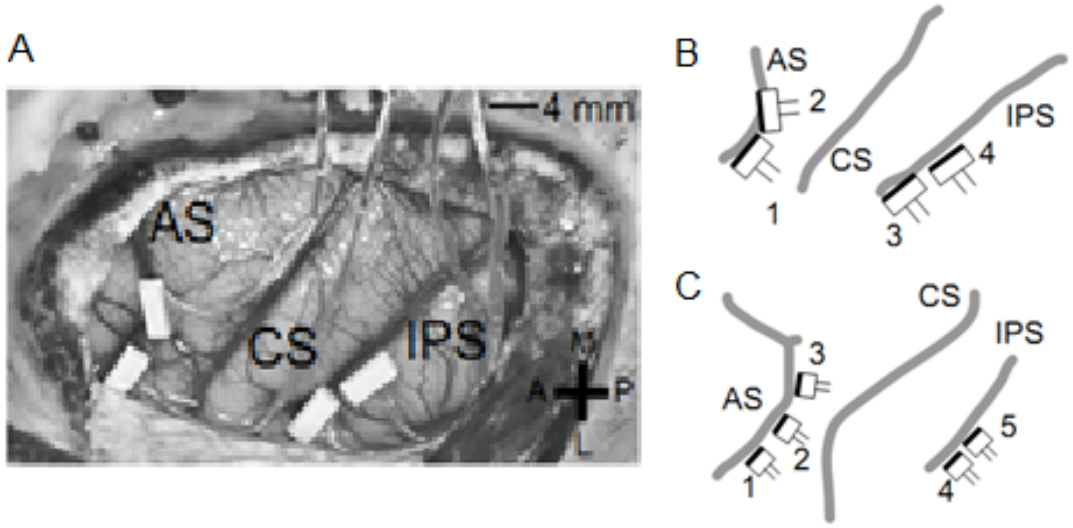

Figure 2-5: FMA implantation details
A. Placement of FMAs in animal S. Two arrays were placed in F5 on the lateral bank of the arcuate sulcus (AS). Two further arrays were placed in AIP towards the lateral end of the intraparietal sulcus (IPS). CS, central sulcus. Cross: medial, lateral, anterior, and posterior direction. **B.** Schematic of FMA placement in animal S including FMA numbering. Dark edge on each FMA indicates row of electrodes with the greatest lengths. Annotations the same as in A. **C.** Schematic of FMA placement in animal Z.

which instructed the type of grip to be performed: for power grip the dot was green while for precision grip it was white. Then, the spotlights and the cue dot were extinguished while the red dot remained illuminated for a variable time period (700-1100 ms, mean 900 ms) during which the monkey was required to remember the grasping instructions (*planning period).* The red dot was then switched off, instructing the animal to reach and grasp the handle in the dark (*movement period)*. Upon activation of the handle sensors, the handle was then illuminated again to allow visual feedback of the executed grasping movement. If the animal performed the correct grasp, this feedback was given together with a fixed amount of fluid (water or juice) as a reward, and the animal could initiate another trial by placing both hands at the hand rest buttons. Execution of the wrong grasp resulted in handle illumination together with presentation of the red dot, but in this case no reward was given. Failure to activate the handle sensors (e.g., when no movement was initiated) led to

trial abortion without visual feedback. Animals were considered fully trained once task performance exceeded 80%.

For *real-time decoding*, each session began by sampling spike data from F5 and AIP during the planning phase in the standard *delayed grasping task*. These first 100-150 trials were used to train the classifier (see below) by calculating the average firing rates during the planning epoch separately for each of the 10 grasp conditions and each unit.

Once this process was completed, the *brain control task* was started (Fig. 2.4B). In this real time decoding task, baseline and cue epochs were identical to the delayed grasping task. However, during the planning epoch, spiking activity was sampled and used to make a prediction at the end of this period, about which grasp condition (grip type and object orientation) the monkey was intending to execute. If the instructed and decoded conditions matched, the monkey was rewarded without being required to execute the movement. Instead, a static picture of the animal's hand executing the decoded grasp was presented on the LCD screen from a perspective of the animal, i.e. as if the animal was actually performing the grasp movement.
Alternatively, if the decoded condition failed to match the instructed condition, the trial was either aborted or the red dot was extinguished, as in the delayed grasping task, which instructed the animal to grasp the target with its own hand. The latter was intended to maintain interest and motivation in the task, in particular when the overall decoding performance was low (e.g., animal Z, see: Chapter 3).

2.3 Surgical procedures and imaging

Upon completion of behavioral training, each animal received an MRI scan to locate anatomical landmarks, for subsequent chronic implantation of microelectrode arrays. The monkey was sedated (ketamine 10mg/kg i.m. and xylazine 0.5mg/kg i.m.), and placed in the scanner (GE Healthcare 1.5T) in a prone position. During the scan the animal was supplemented with O2 (1 l/min), and its heart rate, O2-saturation, and end-tidal CO2-level was continuously monitored. T1-weighted volumetric images of the brain and skull were obtained as described previously

(Baumann et al., 2009). We measured the stereotaxic location of the arcuate and intraparietal sulci to guide placement of the electrode arrays.

2.4 Chronic electrode implantation

An initial surgery was performed to implant a head post (titanium cylinder, diameter 18mm). After recovery from this procedure, and subsequent training of the task in the head-fixed condition, each animal was implanted with floating microelectrode arrays (FMAs, Microprobe Inc, Gaithersburg, MD, USA) in a separate procedure. We used different types and numbers of arrays in each animal. Animal S was implanted with 32 electrode FMAs and received 2 arrays in each area (Fig 2.5A, B). Animal Z was implanted with 5 electrode arrays, each with 16 electrodes. Three such arrays were implanted in area F5, and two in area AIP (Fig 2.5C). Both types of FMA consisted of non-moveable monopolar platinum-iridium electrodes with initial impedances ranging between 300 kΩ to 600 kΩ at 1 kHz measured before implantation. Lengths of electrodes in the 16-electrode FMA were between 1.0 mm to 4.5 mm, and between 1.5 mm to 7.1 mm in the 32-electrode arrays. Finally, for one of the 32-electrode FMAs implanted in F5 of animal S, 16 of the electrodes were carbon nanotube coated (Plexon Inc., Dallas, TX, USA), which resulted in substantially lowered pre-implantation impedances (~5-10 kΩ). The influence of this coating on the long-term recording capabilities will be reported elsewhere.

All surgical procedures were performed under sterile conditions and general anesthesia (induction with ketamine 10 mg/kg, i.m., atropine 0.05 mg/kg, s.c., followed by intubation, isofluorane 1–2%, and analgesia with 0.01 mg/kg buprenorphene, s.c.). Heart and respiration rate, electrocardiogram, oxygen saturation, and body temperature were continuously monitored and systemic antibiotics and analgesics were administered for several days after each surgery. To prevent brain swelling while the dura was open, the animal was mildly hyperventilated (endtidal CO_2: ~30 mmHg) and mannitol kept at hand. Animals were

allowed to recover for at least two weeks before behavioral training or recording experiments recommenced.

2.5 Neural recordings

From permanently implanted FMAs, we recorded spiking activity from multiple neurons simultaneously in area F5 and area AIP while the monkey performed the delayed grasping task. Neural signals were amplified (x300) and digitized with 16 bit resolution (0.25 μV/bit) at 30 kS/s using a Cerebus Neural Signal processor (Blackrock, Salt Lake City, UT, USA) and stored to disc together with the behavioral data. At the same time, we streamed spike and task data via a gigabit Ethernet connection to a separate decoding computer for the real time decoding described below (Fig. 2.3). Spike sorting was conducted online by manually setting time-amplitude discrimination windows for Animal-Z, and using the proprietary automated spike sorting feature of the Cerebus system for Animal-S.

2.6 Real time decoding algorithm

Since our goal with decoding is to predict the most likely hand posture out of 10 possible configurations (2 hand shapes x 5 hand orientations) given the neural ensemble signal, we formalized our objective as a classification problem. We chose spike rate (action potentials firing rate during planning period) as our input signal, following the most widely used approach in current neural-prosthetics literature (Taylor et al., 2002; Brown et al., 2004; Musallam et al., 2004; Hochberg et al., 2006; Achtman et al., 2007; Velliste et al., 2008). Since real time spike sorting algorithms are prone to error, it is very likely that some of the inputs to our classifier will be pure artifacts of the spike sorting instead of being actual single unit spikes. Therefore, we have used a simple feature selection layer first and only those units which were significantly tuned to the parameters of the task (1-way ANOVA with factor grasp condition 1-10, $p<0.05$) in the training data, were further fed to the decoder.

For classification, we utilized a parametric supervised learning scheme, the Naïve Bayesian (NB) classifier. It is an easy to implement, very robust learner, fast on training and classification (such that computation times will be not a problem for real-time setting), and in addition it also performs well in many complex real-world problems. In fact, in a comparison to many stronger learning algorithms in our setup, NB classifiers have shown to be one of the best performers (see Chapter 4, Subasi et. al., 2010).

Besides pragmatic reasons stated above, one can also conclude using Naïve Bayesian classifiers is close to optimal from an encoding perspective as follows.

2.6.1 Motivation for using Naïve Bayes classifiers: A bottom/up view

One main difficulty in understanding population coding arises from the fact that neurons are noisy in nature, thus encoding should be necessarily stochastic. Thus, we must compute some estimate of the stimulus or a probability distribution over stimuli, using a set of responses. Using an elementary probability equation, which is also known as Bayes' formula, one can combine the information from the ensemble of cells, giving rise to a posterior distribution over stimuli, $P(s|\boldsymbol{r})$, as in Eq. 1.

$$P(s|\boldsymbol{r}) = \frac{P(\boldsymbol{r}|s)P(s)}{P(\boldsymbol{r})} \quad (1)$$

Here s represents hand posture we are trying to decode (stimulus for the animal) and $\boldsymbol{r}$ the vector of neural firing rates. For the classification purposes we can iterate over all the postures and simply select the one with the highest probability density, which maximizes $P(s|\boldsymbol{r})$. This estimate is also known as the maximum a posteriori (MAP) estimate. Furthermore, since $P(s)$ is homogenously distributed in our experiment, ie. the probability of each posture to be signaled to the animal is equal, and the $P(\boldsymbol{r})$ is not dependent on the stimulus s, maximizing $P(s|\boldsymbol{r})$ will be equal to maximizing the likelihood function $P(\boldsymbol{r}|s)$. Thus the MAP estimate for s will be equal to Maximum

Likelihood (ML) Estimation (Eq. 2) of it, which is one of the most widely employed methods for estimation problems in classical statistics.

$$\hat{s} = \underset{s \in S}{\operatorname{argmax}} P(r_1, \ldots, r_n | s) \tag{2}$$

ML estimator is not only an asymptotically unbiased estimator (consistent), it also has the minimum mean squared error among all unbiased estimators (efficient). In other words, given enough data, ML estimator is guaranteed to converge to the real probability value while having the minimum variance (Cramér–Rao lower bound) compared to all the other unbiased estimators. It is after these desirable mathematical properties, that ML estimator is also very popular by the practitioners of neural-signal decoding community and used in many setups yielding to state of art results (Brown et. al., 2004; Achtman et. al., 2007; Ma et. al., 2006).

It is also worth to note that, unlike in many real-life ML estimation problems where finding the global maximum of a high-dimension, continuous likelihood function is challenging, it is trivial in our setup since we are working on a finite and discrete stimulus space. Thus, utilizing ML estimate was very straight forward; simply iterating over likelihood values of 10 different conditions and picking the one with maximum value.

One final challenge left for the classification task arises from the fact that the proper estimation of the conditional joint probability $P(r_1, \ldots, r_n | s)$ may require some significant amount of data. The data need scales exponentially with the number of neurons and since we have in the order of hundred neurons, the curse of dimensionality -as often called in the literature- will soon render the problem intractable for this animal experiment.

Here comes the naïve assumption from NB classifier into play, such that we assume that each neuron fires conditionally independently of one another. After this assumption the estimation problem will require manageable amounts of data and we can factorize the previous conditional joint probability function as in Eq. 3;

$$P(r_1, \ldots, r_n | s) = \prod_{i=1}^{n} P(r_i | s) \tag{3}$$

Now we only need to choose a distribution family for the independent probability distribution, $P(r_i|s)$. The neuronal firing variance in primate motor cortex is known to be in the same order with the average firing rate (Shadlen et. al., 1998; Garstein et. al., 1964; Ma et. al., 2006). A natural candidate is therefore Poisson distribution.

$$P(r_i|s) = f(r_i, \lambda_s) = \frac{{\lambda_s}^{r_i} e^{-\lambda_s}}{r_i!} \quad (4)$$

In Eq. 4, r_i is the number of spikes observed from neuron-i in a particular trial and λ_s is the expected firing rate for the same neuron for the condition s. Thus, for making classifications in real-time we need to first estimate the parameters λ_ss for each neuron. Using ML estimation one can show that the parameter, expected firing rate for a stimulus s, is nothing but the arithmetic average of firing rates. And for a proper estimation of λ_s, our experiments showed that only around 10 samples per condition is enough. Each real-time decoding session began with sampling of spike data from F5 and AIP during the planning phase while the animal performed the standard *delayed grasping task*. These first 100-150 trials are used to train the classifier where we calculated average firing rates during the planning period, for each of the 10 grasp conditions for each neuron. This is indeed the only training a NB classifier needs; therefore we can conclude training was efficient and fast.

2.6.2 Real time decoding trials

Once the training process was complete, the monkey began *real time decoding trials* (Fig. 2.1B). Between 80-200 trials per session were used to test the decoder as follows. Fixation and cue phases were presented as for normal movement trials. Then, during the planning phase, spike data were sampled and used to make a prediction (at the end of this period) about which grasp condition was intended by the monkey, by choosing the condition which maximizes log-likelihood function, as in Eq 5.

$$\hat{s} = \underset{s \in S}{\operatorname{argmax}} \sum_{i=1}^{n} \log\big(P(r_i|s)\big) \quad (5)$$

The actual form of the likelihood requires multiplication of many very small numbers, which is not a desirable operation using double precision arithmetic in modern CPUs. By introducing a monotone transformation like log, we deal with these numerical instabilities while not altering the maxima location.

If the decoded and instructed conditions matched, the monkey then received a small juice reward, while being presented simultaneously with a static image of its own hand executing the corresponding grasp condition. This was presented by means of the LCD screen and half-mirror, with the display controlled via a separate visual feedback computer. Each image was presented such that it overlapped closely with the corresponding image the monkey would have seen if it had actually performed the instructed grasp with its own hand.

Alternatively, if the decoded condition failed to match the instructed condition, the trial was either aborted, or the fixation LED was extinguished as during the delayed grasping task, instructing the animal to make a reach and grasp the target with its own hand. The latter was intended to maintain the animal's interest and motivation in the task during real time decoding, by enabling the animal to periodically execute grasps in return for reward when real time decoding trials could not be completed owing to potential inaccuracy in the decoder's output. Such a step was considered necessary since somewhat low decoding performance, especially in animal Z (See Chapter 3), could otherwise have led to the inability of the monkey to complete successive real time decoding trials despite it having planned the correct grasp.

Decoder performance in each session was evaluated via the total percentage of correctly decoded trials achieved by the end of the session. In addition, we tested changes in decoder accuracy within each session using a sliding window analysis that monitored the performance of the last ten trials.

2.7 Offline data analysis

All data recorded during real time decoding sessions were also stored to disk for offline analysis.

2.7.1 Spike sorting

Raw signals were band-pass filtered (pass band: 300-3000 Hz), and single and multi-units were isolated using superparamagnetic clustering techniques (Waveclus software running in MATLAB) (Quiroga et al., 2004). The quality of single unit isolation was evaluated using three criteria: first, the absence of short (1-2ms) intervals in the interspike interval histogram; second, the degree of homogeneity of the detected spike waveforms, and third, the separation of waveform clusters in the projection of the first 10 wavelet coefficients with largest deviation from normality (Quiroga et al., 2004). In the majority of cases it was not possible to isolate single units due to indistinguishable shapes of waveforms, especially with low amplitude. Waveforms were thus pooled into a larger "multi-unit" which comprised recordings from several individual neurons simultaneously. However, care should be taken to distinguish this point-process signal from continuous "multi-unit activity" (MUA) data generated from envelope functions applied to the low-pass filtered voltage trace (Super and Roelfsema, 2005; Stark and Abeles, 2007; Choi et al., 2010). Finally, the predominance of multi-unit recordings in our data set was in part due to the fixed (non-movable) nature of the electrodes, which did not allow optimization of unit isolation during recordings.

2.7.2 Visualization

To visualize neural activity during the task, peri-stimulus time histograms (PSTHs) were generated by replacing each spike time t_s with a kernel function and averaging all such functions across all spikes and trials (Kass et al., 2003). We used a gamma distribution function as a kernel:

$$R(t) = \begin{cases} (t-t_s)^{\alpha-1} * \beta^{\alpha} * \exp(-\beta(t-t_s))/\Gamma(\alpha) & \text{if } t \geq t_s \\ 0 & \text{if } t < t_s \end{cases}$$

The shape ($\alpha = 1.5$) and rate parameter ($\beta = 30$) were chosen to achieve a small amount of delay (kernel peak at 1.6 ms) and a standard deviation of approximately 40 ms. This procedure ensured that the resulting PSTH curve was smooth, continuous, and causal, i.e. the value at any time point was only influenced by spikes

that had occurred prior to that moment in time, but not afterward (Baumann et al., 2009). However, note that all quantitative analysis and statistical tests were based on the exact spike times without any smoothing.

2.7.3 Tuning

We also quantitatively analyzed the underlying tuning properties of each unit by examining spiking activity during the planning period in the *delayed grasping task*. Firstly, we parameterized each cell's tuning to the task parameters in terms of its preferred and non-preferred grip type and orientation. These were determined for each cell from the mean firing rate during the planning period, which was averaged across all trials of the same grip type or orientation. The preferred grip type was determined as the grip with the highest mean firing rate, leaving the other grip as the non-preferred type. The preferred orientation was defined as the orientation with the highest mean firing rate, while the non-preferred orientation was given by the orientation located at 75° angular distance from the preferred one. This definition was chosen so that the non-preferred orientation was not taken exclusively from the extreme orientations (±50°). If the preferred condition was 0°, we then randomly selected either -50° or +50° as the non-preferred condition (Baumann et al 2009).

Next, we tested the statistical significance of each cell's tuning by means of a two way analysis of variance (ANOVA) with factors grip type and orientation and significance $p<0.01$, with the additional requirement that the cell fired at least 5 spikes/s in the preferred condition.

As a further measure of the tuning strength, we performed a receiver-operating characteristic (ROC) analysis (Dayan and Abbott, 2001). This tested how well one can discriminate, based on the spiking activity of a given cell, between trials with the preferred grip type (or orientation) and trials with the non-preferred grip type (or orientation). We used the area under the curve (ROC score) as a measure of discriminatory power ranging from 0.5 (for chance performance) to 1 (for perfect discrimination). For grip type tuning, we computed the ROC score separately for each orientation and then averaged across all orientations for each grip type. For

hand orientation, we averaged the ROC score across all trials for the preferred and non-preferred orientation irrespective of grip type. To assess the significance of ROC scores, we used a Monte Carlo procedure, in which 1000 repetitions of the same ROC analysis were performed with random shuffling of the labels 'preferred grip type' and 'non-preferred grip type' (or preferred/non-preferred orientation), in order to determine the null distribution of our hypothesis.

2.7.4 Offline decoding simulation

In order to test the performance of the NB classifier under optimal conditions (optimized spike sorting) and as a comparison to the real time decoding results, we decoded the grasp condition also offline and using the offline spike-sorted data. A naive Bayesian classifier was implemented in Matlab, and trained and tested on the same data used to train and test the original classifier. However, instead of using spike data streamed from the Cerebus NSP, this offline classifier could also operate on spike data extracted from the offline spike sorter Waveclus. In addition, simulated decoding was done on restricted data sets, such as AIP or F5 data only. When a one-way ANOVA was carried out to compare decoding performance using each area independently versus both areas combined, additional post-hoc testing was done to locate significant differences via multiple comparisons at the $p<0.05$ level (Tukey-Kramer correction).

2.8 Software implementation for online decoding

Important design constraints

One of the most critical points in the initial design stage for the software development in this project was to guarantee that everything is indeed working when data collection started with the animal. And this was important due to multiple reasons. First, the signal quality from chronically implanted cortical electrodes is known to deteriorate over time. The underlying mechanisms of this are not fully

understood and can show great variability from case to case. However, it is almost certain that it will happen in time with currently available electrode arrays. The only question is when? In extreme cases the signal quality was reported to be unusable after only a few months. In contrast, training of animals and performing all surgical operations is very time-consuming and demanding in multiple resources. Therefore, once the animal is ready to perform, we simply cannot risk losing these efforts due to malfunctioning software. Especially, we cannot risk losing valuable experiment time right after electrode implantation when we know that signal quality is high. Another reason why having a working system right at the beginning of the experiments arises from the fact that the animals can get frustrated easily if the system is not behaving consistently. So, if the system has a lot of initial bugs, the animal could loose its motivation to work, and it will take precious time to win it back. A final point is the fact that the number of recording trials per day was modestly limited. We could expect that the animal will work for about 400 trials per day on average, so we had to be very economical with the recordings, and simply speaking, needed a system that worked right from the beginning. On top of that, we needed a system which utilized system resources efficiently; listening with 100Hz events flooding through hundreds of channels in parallel, keeping the order and timings right, making statistical estimations and predictions on the fly, and visualizing the whole process was obviously open to multiple error sources.

Utilizing unit-tests for the critical calculations in the decoder was an important approach we followed. However, in order to go through a real stress-test the decoder needed to be tested with data flooding the system in real-time. To this extent, we came up with the idea of an additional software component, a neural signal simulator, which will push artificially generated or previously recorded data to the input stream of the neural decoder (Fig 2.6). By targeting the same network ports of the actual signal processor, the system replicates the effect of having an actual animal connected to the setup and it is virtually impossible for the decoder to distinguish between the signal sources. This approach has proven to be very useful and we were able to start data collection and real-time decoding of the data on the very first day the animal was in the experiment.

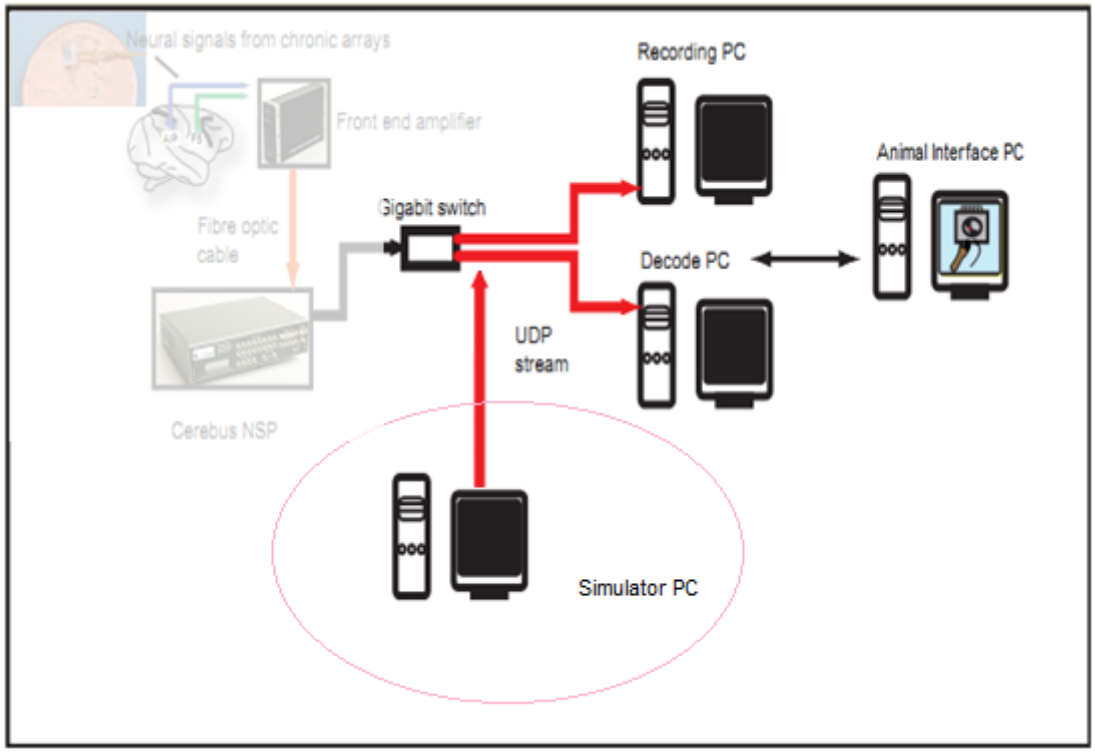

Figure 2-6: The generic system design for simulation environment.
Simulator software is seamlessly integrated to the system network architecture. This helped us to test the system thoroughly before the animal is ready to perform the experiments.

Implementation details

Neural signals from floating micro-electrode arrays are sampled using Cyberkinetics Neural Signal Processor and streamed to recording and decoding PCs. Animal interface PC is controlled via LabView software and decoding PC.

We have used LabView for controlling the experimental setup and used Matlab, C++, Python, Scala and Java for various parts of the off-line data analysis. For real-time decoding and animal interface systems, we have used our in-house developed software suit. On top of that, due to various benefits for testing and debugging the real-time decoding software we have also developed a spike simulator tool capable of creating artificial Poisson-distributed spike trains as well as loading and replaying previous neuronal recordings. This simulator software is used for benchmarking

decoder performance offline before commencing BMI experiments with the animal. All these 3 components were implemented in C++, heavily utilizing Trolltech's (accuired by Nokia lately) Qt Framework (http://qt.nokia.com/). The choice of Qt framework was for its desirable features of providing a coherent framework with components for graphical user interface, networking and commonly used data structure implementations. Besides we have also utilized Neuroshare library for reading Cerebus record files, the Cerebus UDP Network Protocol and a scientific library, GSL (GNU Scientific Library - http://www.gnu.org/software/gsl).

2.8.1 Experiment Data Simulator Software

Compared to off-line decoding, the realization of real-time decoding systems is challenging, in part because the number of available neuronal units is limited and the time window in which the data needs to be analyzed is relatively short (<100ms). If one adds these to the complications of working with an animal where only a limited amount of work-time per day is available and the fact that a buggy system will negatively affect the animal's motivation to collaborate, it is clear that it is crucial to validate decoding algorithms for their online suitability and to fine-tune their performance before being used in neuro-physiological experiments. Also one should note that real data is often not optimal for benchmarking purposes due to its inherent noise and stochasticity.

Therefore, we have developed a spike simulator for evaluating our online decoding software. This software creates and streams data in the exact format as Cyberkinetics' neural signal processor, thus for the decoder software it is indistinguishable from which source (either real or simulation data) the data is coming. The simulator comprises three main features:

1) It can create artificial data with regularly spaced or Poisson distributed spike trains. This mode helps validate the decoder's data capturing limits and its principle performance.

2) The simulator can load and replay previous neuronal recordings. This is useful for benchmarking and cross-validation of different decoding algorithms and to compare the performance of online and offline decoding methods.
3) The tool can also load multiple previous recordings and combine them as if they were recorded simultaneously in real time. This capability is useful to test decoding algorithms with an in principle unlimited number of neurons of natural response characteristics, which, e.g., have been recorded sequentially with movable electrodes. It is also useful for obtaining statistical robustness about our decoder's performance by the mixture of existing data in a similar fashion to bootstrapping methods.

We have used the simulator software during the development of the decoder extensively and confirmed the suitability of it for the development of any on-line decoding software in a similar fashion. In its latest version the simulator software's source code is around 26000 lines (including the graphical user interface components, excluding comments and empty lines). Not necessarily the best statistic capturing the quality of software development effort we still wanted to provide this LOC figure in order to provide some measure of complexity and size of the software. A generic diagram of the Simulator software is provided below in Figure 2.7. For a more detailed class diagram (including class attributes and method names) see Appendix-B. There are three main clusters of classes for the simulator: application logic, visualization, and data representation and wrapping. All of these components are decoupled seamlessly and multiple threads are employed for different components providing a robust and reliable system. A schematic representation of the main building blocks of the simulator software is presented in Figure 2.7. The following Figures 2.8 –2.14 then show screenshots from the different operation modes.

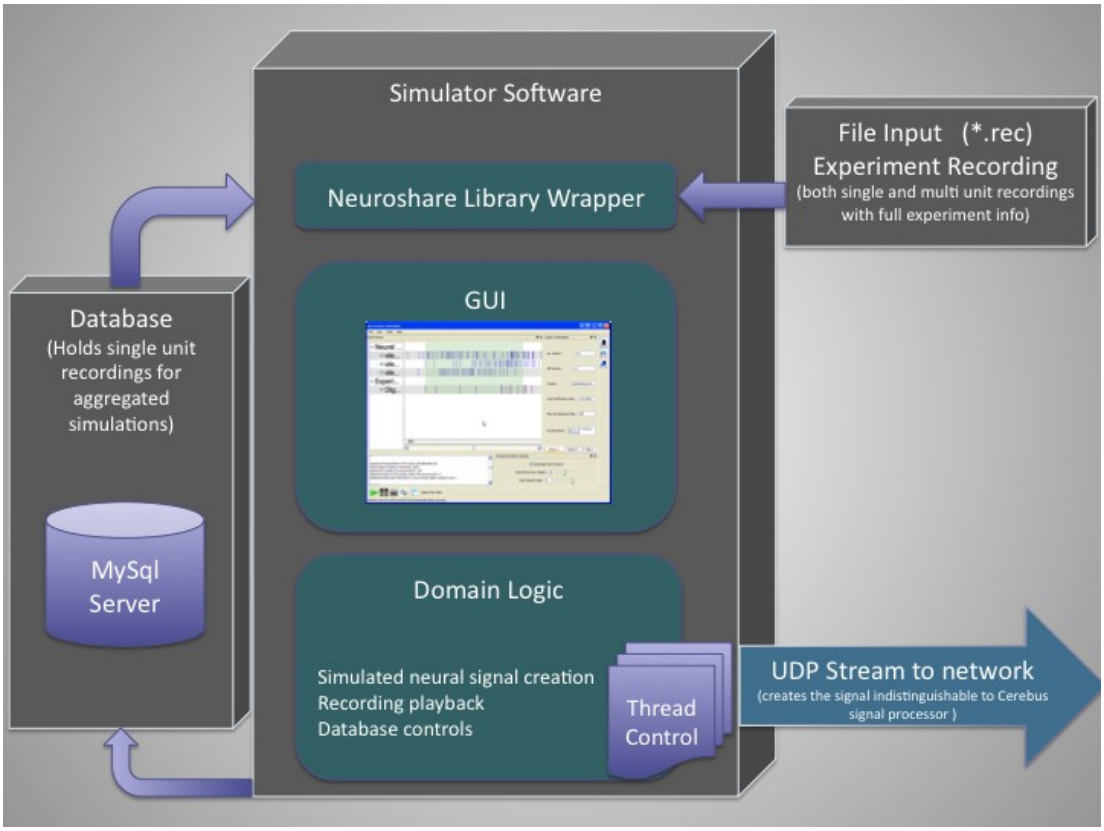

Figure 2-7: Generic diagram of the Simulator Software design.
Schematic representation of simulator software environment is shown. A C++ wrapper converts the standard file format "Neuroshare" recordings from other software vendors to the internal data format. It is coupled to a database server used for the deployment and management of previous recordings. Signal queries from this database are used to create artificial multi-unit signals. The GUI provides means to control the data flow and to display both actual and simulated recordings in a convenient way. For pushing the data to the network in the same format as the actual Neural Signal Processor, the simulator software makes the necessary transformations and then streams the output to the network via UDP protocol. GUI components also run in a separate thread to guarantee the flawless operation of the mission critical components.

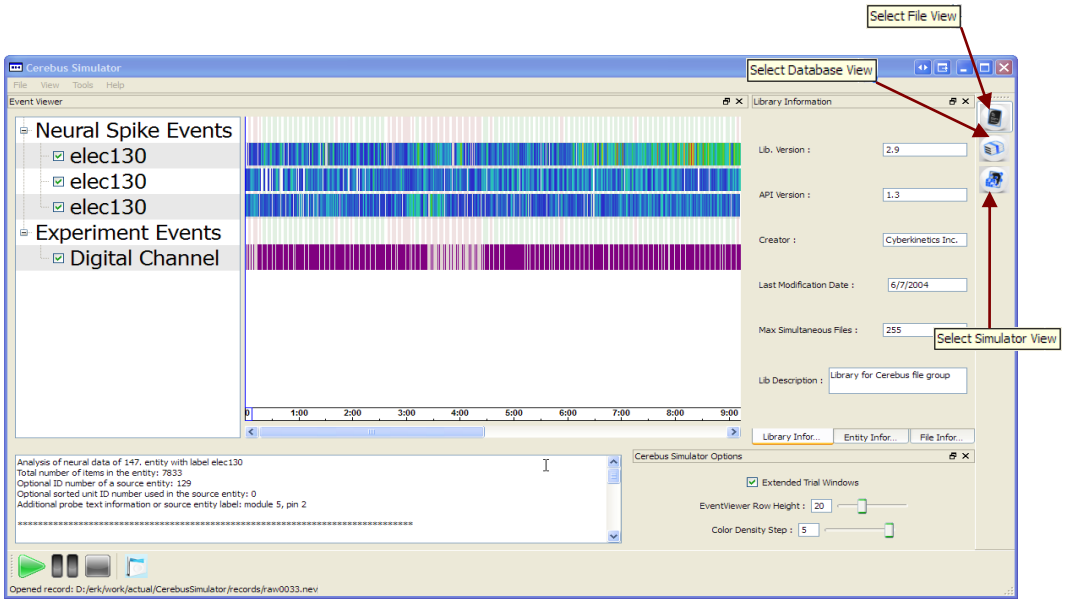

Figure 2-8: An example view of simulator software

Simulator software's main view consists of the dockable widgets shown above. The central "event viewer" widget is showing each units individual neuronal firings as well as experiment timings. On right hand side three buttons switch to necessary views regarding the three operating modes of the software: File View, Database View, and Simulator View. File view essentially lets the user load a previous recording and replay it. Database view visualizes and lets the user interact with the database holding single-electrode recordings from previous studies in our lab. This database is used to simulate multi-electore recordings that are used for testing and development before actual multi-electrode recordings were available. Finally, simulator view creates artificial data for benchmarking purposes. Other display and control information is also visible on the right and bottom of the central Even Viewer widget. The GUI is composed of independent widgets that control different software components and share information via common data structures. All these modules can be rearranged visually and activated-deactivated according to user needs. At the bottom of the GUI canvas, the stream control toolbox mimics a video player's controls. For example, pressing the green play button will push neural data to the network (either loaded or artificially created). Similarly, the other buttons will "pause" and "stop" the neural data streaming.

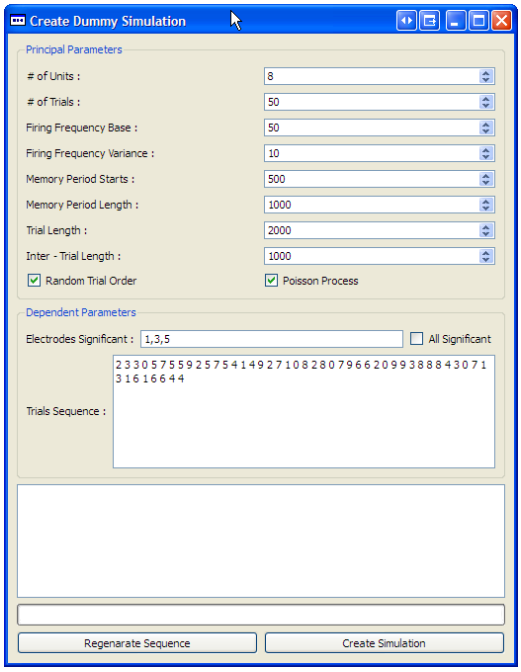

Figure 2-9: Artificial data dialog window for simulator software
After selection of artificial data creation mode the above shown dialog window pops up to collect user inputs on number of units, trials, firing rate statistics and experiment state lengths.

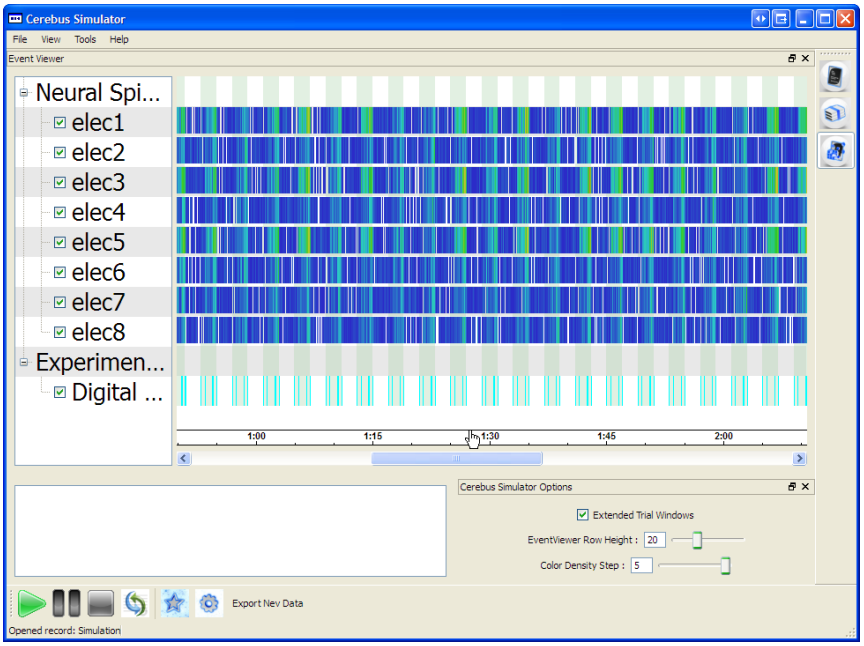

Figure 2-10: Artificial data created with given parameters in Fig. 2.9
50 trials with desired output characteristics are ready to stream to network with pressing the play button. Each green column represents a single successful trial.

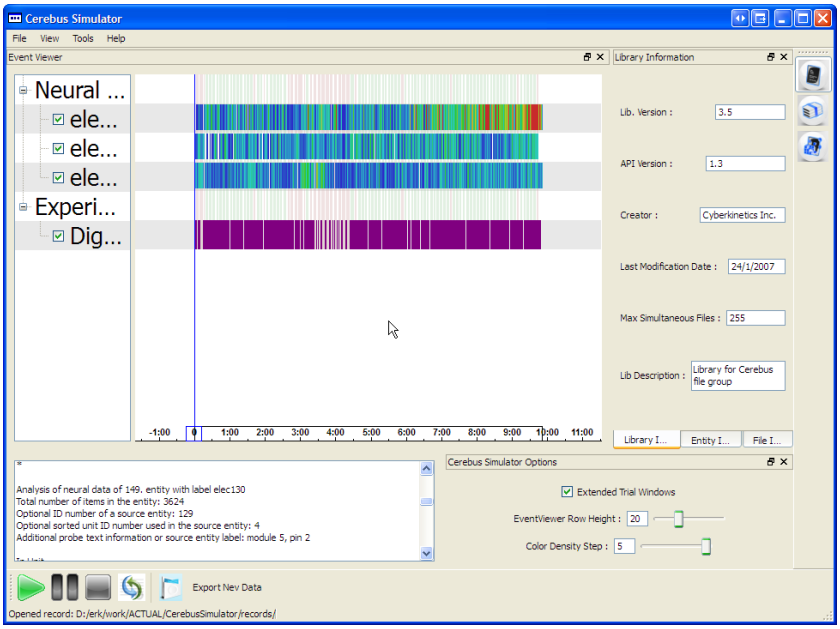

Figure 2-11: Zoom-out view of an actual 3 unit record in event viewer widget
The vertical blue line shows the last point streamed similar to a video player. By clicking to time-line in the bottom pane it is possible to jump to a different time in simulation. During playing it slides in real-time. The ability to zoom in and out using the mouse wheel in this widget lets the user visually analyze the broader characteristics of firing rate statistics in a form similar to a peri-stimulus time histogram with color codes. Red shows highest firing rate regions, whereas blue lowest.

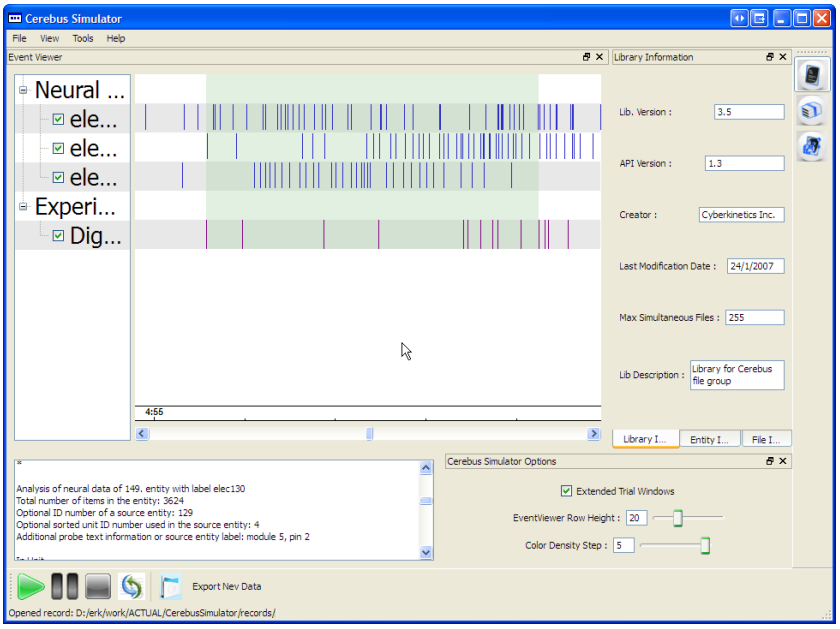

Figure 2-12: Zoom-in view the same data on a single trial
Zoom-in let the user observe individual spike timings in a single experiment trial.

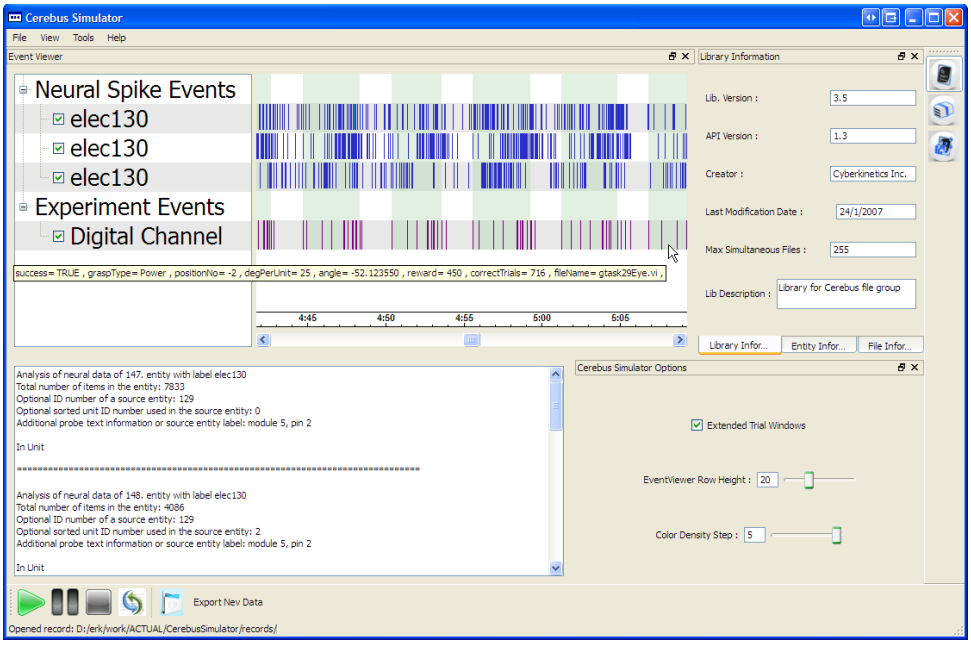

Figure 2-13: Trial information available interactively
By moving the mouse cursor to an unoccupied region in a trial one can see the digital channel information encoded as a string.

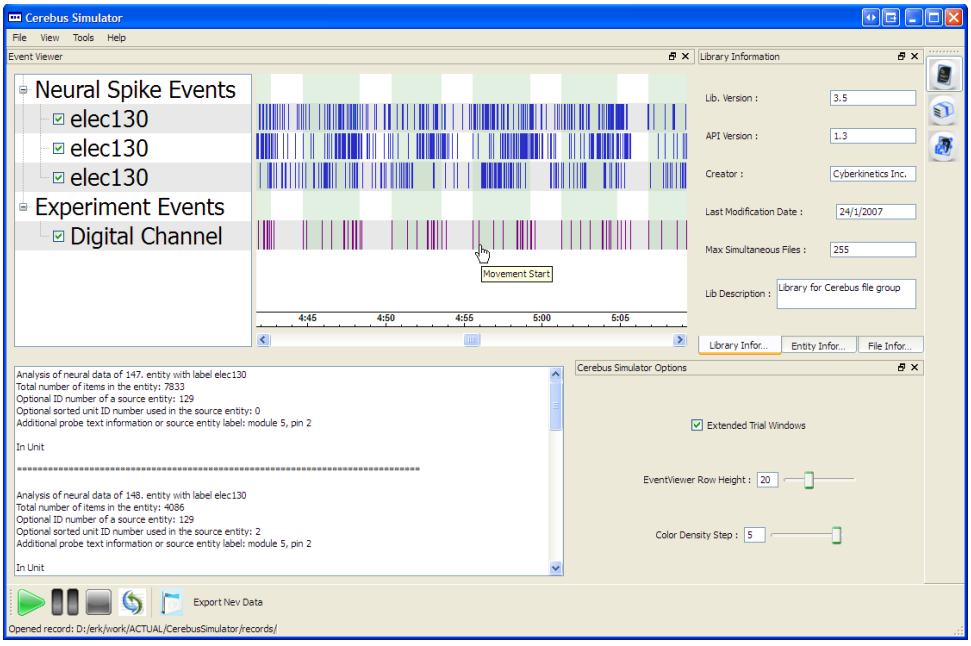

Figure 2-14: Experiment events timing are observable through a digital channel
It is also possible to analyze which digital channel spike is corresponding to which information by simply moving the cursor over it. The actual timing of movement start signal in trial is shown in the figure.

2.8.2 Decoder Software

One of the most critical parts of the software development efforts during this project was the realization of an efficient and robust implementation of the decoder. That software needs to reliably collect information coming from network; from digital channel (meta-info on experiment) and from about 200 neurons firing with maximal 1 kHz events each. It needs to identify different units and calculate their statistical properties during the training phase and then later during the decoding phase make predictions about the intended hand movement given the data using the Naïve Bayesian learner implemented internally. Furthermore, it communicates its output to the recording system on a different machine for logging purposes and to the behavior control system. Finally, it visualizes the data collected in different forms and displays the calculated statistics on decoding performance numerically and graphically. During the course of the project some other features, like decoding of experimental states, calculation and visualization of power spectrum were also added. The source code length for this software is around 12000 lines. We provide a schematic representation of the software design and structure in Figure 2.15. Additionally, a detailed class-diagramm can be found in Appendix-B.

One of the critical design decisions was to use a modular architecture for this software. The main software essentially provides a canvas for different widgets to be plugged-in where they can be visualized and exchange data seamlessly. Utilizing Qt's widgets as our base classes, we developed an event driven architecture that launches new threads when necessary. And this design provided a reactive user interface while collecting the data robustly. Separating the threads for data collection and user interface was critical. Due to heavy real-time visualizations, the computational need of the user interface components was not negligible and a possible bottleneck arising from these components could have easily affected the more mission-critical parts of the software, namely data collection and decoding analysis.

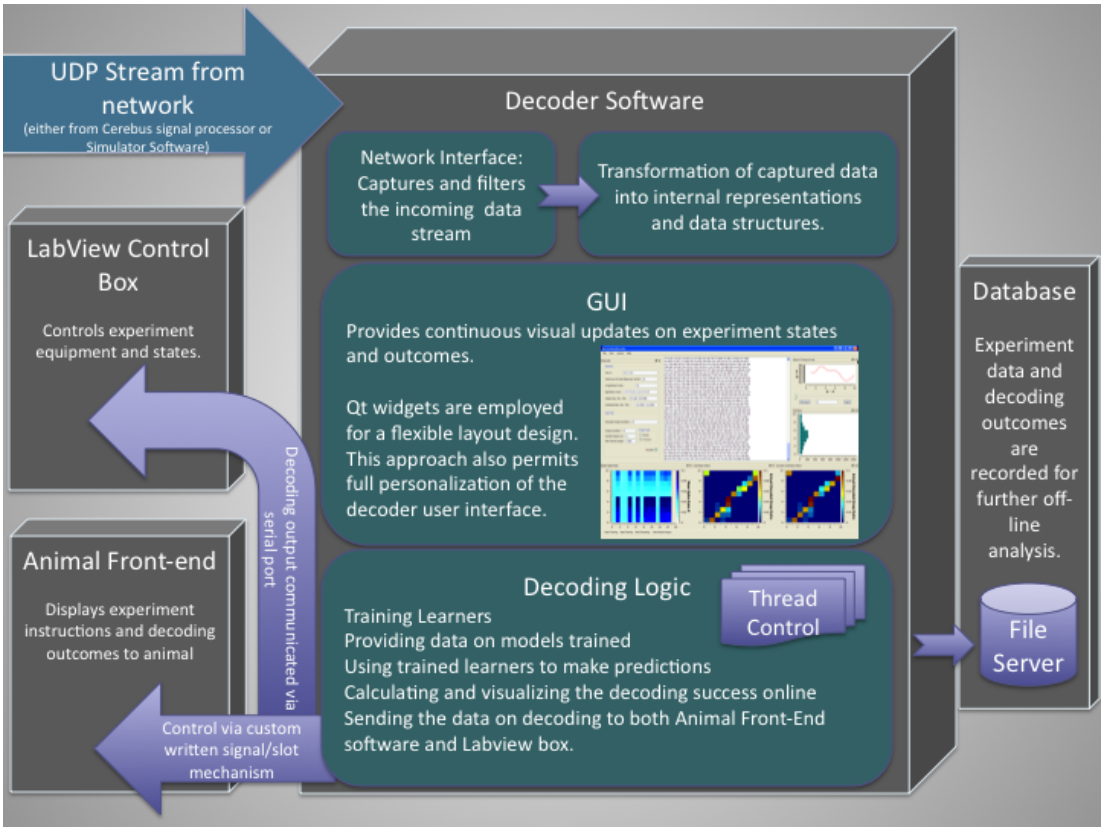

Figure 2-15**: Schematic representation of the Decoder Software components.** The decoder software is fully developed in C++ by heavily employing the QT Framework. The streaming data from the network is captured by continuous listening to specific ports and transformed into internal data structures for further analysis by other software modules. GUI employs some major control elements for starting data collection and decoding phases. Its modular widget based design provides the user with rich personalization capabilities. A user can switch on and off different widgets and keep her settings for the next run conveniently. In initial data-collection phase the user is informed about the progress via both a mean firing rates matrix visual and different data fields. Once the user selects to switch to decoding phase the decoder first calculates some statistics about the collected data, creates tuning curve graphs and train its internal learners (Naïve Bayesian decoders). It also provides some statistics about in-sample decoding results to the user. Then it automatically switches to decoding mode and all the new data flowing through network interface is used for making predictions. Decoder software listen not only neural signals but also experiment state signals provided in the network stream and conditioned on that data it decides when to make its predictions. After each succesfull experiment trial the GUI is updated correspondingly via confusion matrix visuals and some statistics provided in text form. It is also responsible from sending the decoding outputs to animal front-end control software and LabView control box. Similar to simulator software, decoder software also employs separate threads for GUI and mission critical components.

Using the *decoder* software in conjunction with the *simulator* turned out to be beneficial to test against such cases. Simulator's controlled outputs has proven to be very useful and crucial for benchmarking purposes. We could demonstrate that the *decoder* software is able to collect and analyze data reliably simulataneously from around 200 – 300 units even under non-biological high firing levels (>500 Hz). The decoder has essentially 2 modes of operation; training and decoding modes. At first, when software started the decoder opens up in training mode and upon pressing to "start training" button, it actively starts to listen network ports provided in its configuration files. The decoder can also load previous training sessions and start decoding with that initial setup if desired. On the normal real-time operation however, whenever data in the pre-configured format starts to streaming, it starts to collect multi-unit firing rate information for different neurons, experiment states and trials to build an internal database for those. Meta-information regarding the experiment states are generated in Lab-View and interpreted by our decoder in real-time. Once the experimenter decides enough data is collected for reliable estimation of firing rates per hand posture per neuron, he presses "stop training" button, to initiate calculation of the various statistics for collected data and training of Naïve Bayesian learners. An initial estimate of decoding confusion matrix on training data and neural tuning curve visualizations are provided after successful completion of this analysis and after switching automatically to a new display setup. We have implemented different template based visual arrangements of our widgets in this software, which allows an advanced calibration of different visual scenarios. In other words, one can arrange visually the widgets available in a view and then record this arrangement and recall this after any time he needs it. At this point, initial data collection and the training of our learners are successfully finished, thus the researcher press "start decoding" button and the learners start making their classification at the end planning period and feed their predictions through network to Lab-View, Logger and animal interface software. The animal interface software's event capturing mechanism wakes up upon successful arrival of a signal and shows the decoded hand posture in a static image form to the animal. This operation of the decoder continues until either the researcher decides that enough data is collected, or

more typically, the animal loses its motivation to work. During the course of decoding, the confusion matrices for decoding are calculated and displayed for assessing the performance of the decoder in real-time.

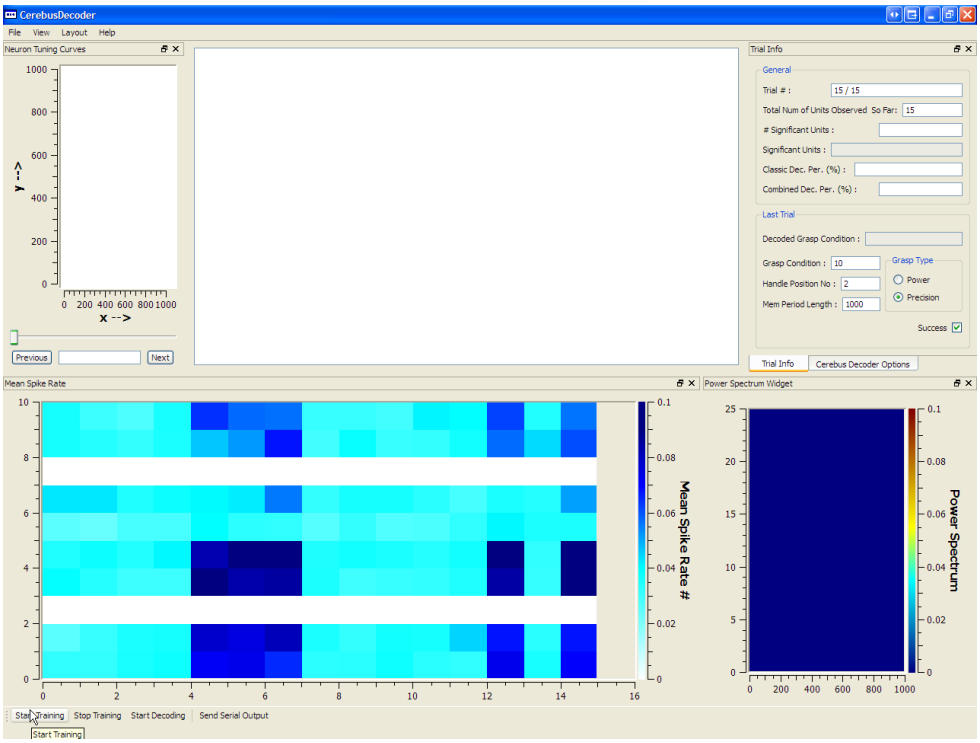

Figure 2-16: A sample view from decoder software during training period

Once the user presses the "Start Training" button in the bottom toolbox, the decoder software starts listening to the relevant ports and captures and processes the incoming data. The screenshot shows data collection of 15 artificially created neurons. One can track the average firing rate estimations via the visualization widget on the left bottom corner. In this matrix, each box is colored according to the estimated mean firing rate for a specific neuron and experiment condition (high firing rates are visualized in dark blue). Here, neurons 5,6,7,13 and 15 show higher activity and also employ a better neural signal tuning for the experiment task in hand. (no data present for experimental conditions 3 and 8: corresponding rows are white). Such a visualization during data collection greatly facilitaes the user to spot possible problems in real time and to have a broad feeling about the neural tuning. Early detection of data recording problems minimizes costly loss of experimental time and data.

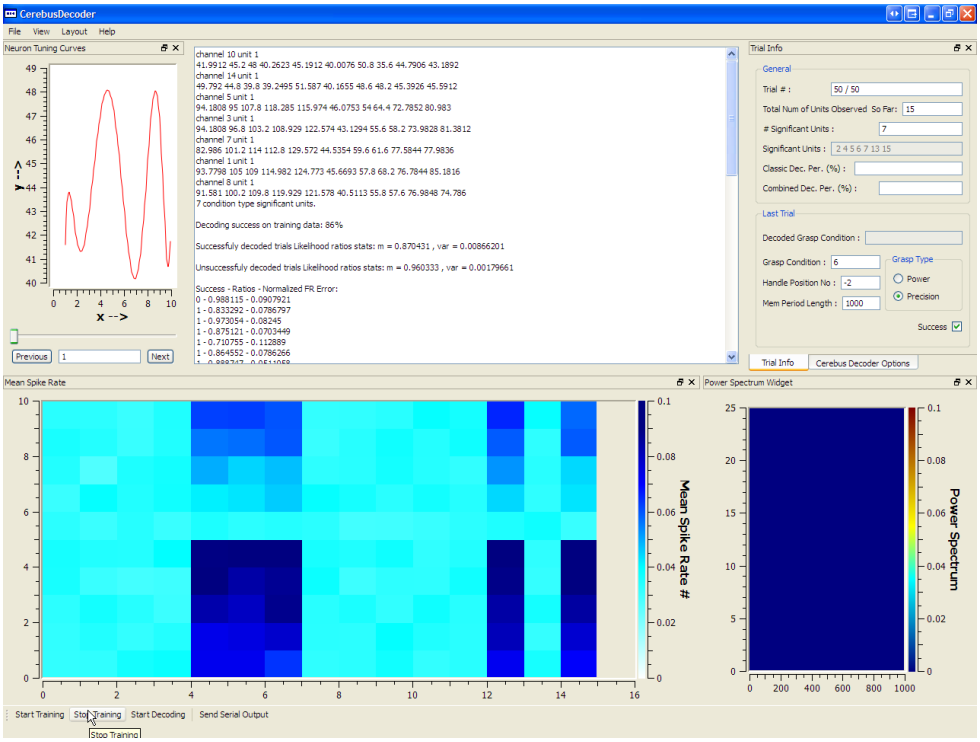

Figure 2-17: Firing rate statistics are calculated after training is finished.
Once the user decides that enough data is collected for training of the internal Naïve Bayes learners he/she presses stop training and the decoder software automatically starts training the learners and calculating other statistics. It also selects significantly tuned neurons (according to an ANOVA criterion; $p<0.01$) and the graph on the left upper corner shows a polynomial fit of the average firing rate per condition. Other statistical details are shown in the top middle console window. After training, the decoder automatically runs the learners on the training data to provide a first feedback about the in-sample performance of the decoding model and then simply waits until the user presses "Start decoding" button.

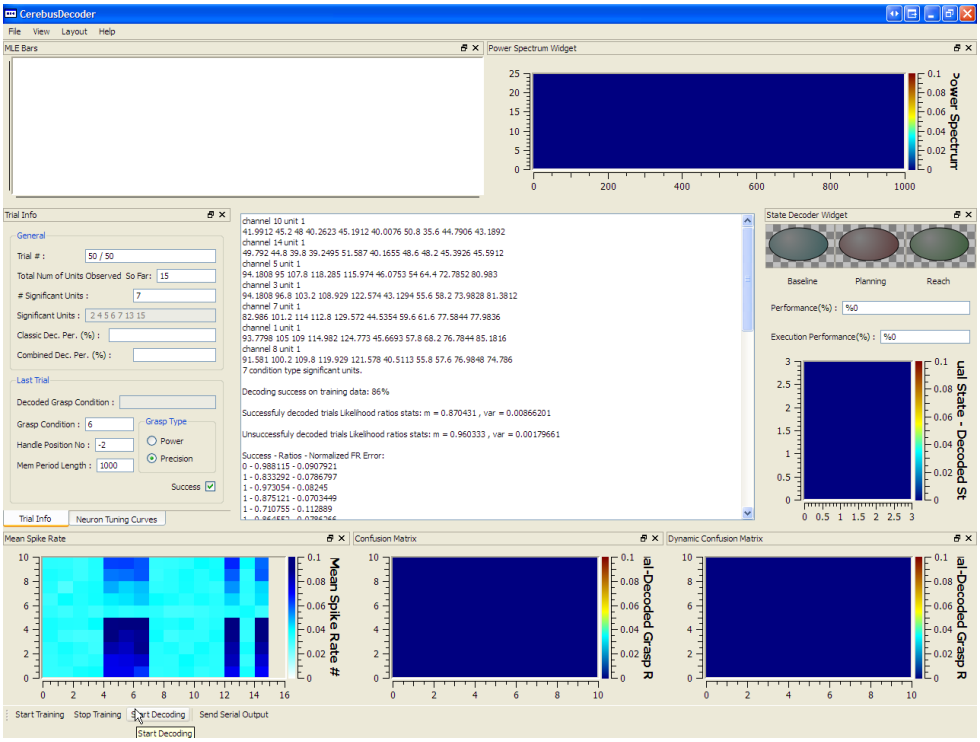

Figure 2-18: Decoding view right after training

Once the user presses the "Start decoding" button, new widgets appear in the GUI to inform the user about the real-time decoding statistics. Average firing rates are still visualized on the left bottom corner, but they should be stable (otherwise data collected for training was not sufficient). Two new visualizations panels show confusion matrices at the center and right bottom position. They provide an intuitive visualization about the decoding performance ,one by a model that continuously updates its average firing rate estimates and the other one with a constant training set as determined by the end of the training period. This allows tracking of the difference between these two different approaches. On the left hand side of the central row, statistics on data collection and decoding performance are provided in text form. The right hand side widget of the same row shows the predicted experiment-state (temporal decoding of the experiment state) and also visualizes the performance of this state-space decoding via a separate confusion matrix. The top row contains two additional widgets for visualizing the likelihoods of each condition at the end of each trial and for providing information of the LFP power spectrum of the raw signal (not used here but implemented for future application).

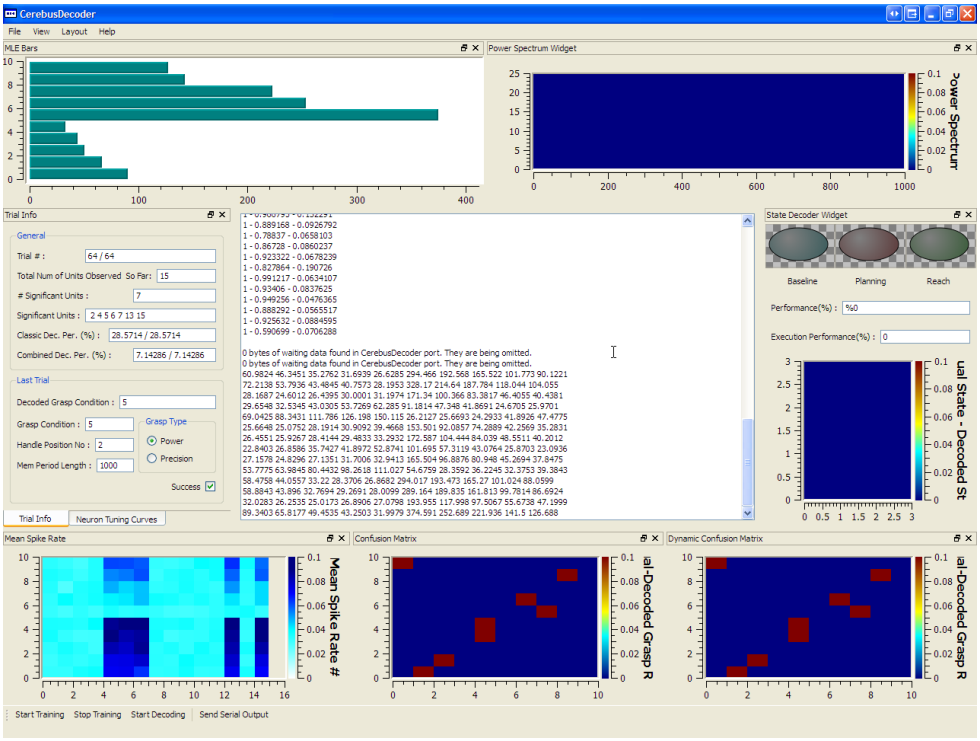

Figure 2-19: Decoding view during online brain control trials

This figure shows a random moment during the decoding experiment. Decoding performance is visualized in confusion matrices and in the (central) trial info widget. Likelihood bars (top left) provide some intuition on the decoding of each trial. The decoder sends the information about the decoded condition immediately (i.e., in less than 100 ms) to the animal-intercace software (LabView control box), which essentially closes the information control loop. Real-time visualization of the decoding performance allows a user to spot possible errors (e.g., recording problems) quickly and to understand how the recorded neurons are encoding the information. The decoding mode essentially collects data, makes predictions, and updates its statistics until the user stops the task.

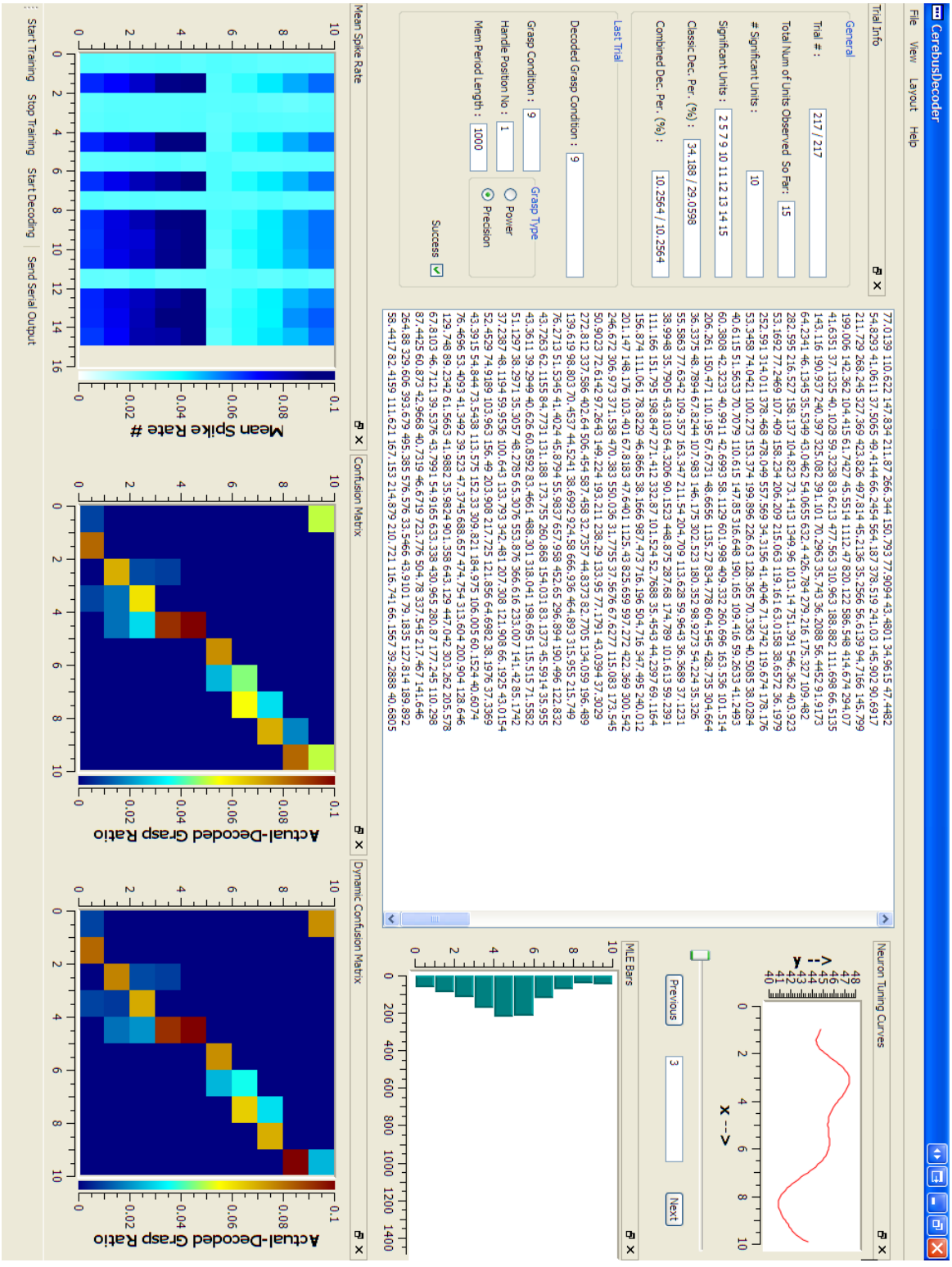

Figure 2-20: A view from decoder towards the end of experiment
Confusion matrices and trial info widget inform on final performance of the session.

3 Real-Time decoding of hand grasping signals

3.1 Introduction

This chapter presents major findings on the main objective of this thesis work; namely, assessing and analyzing the implementation of a real-time hand-grasping brain machine interface and inspecting the characteristics of the signals emerging from the areas under interest. The content of this chapter is published in the Journal of Neuroscience with a title "Grasp movement decoding from premotor and parietal cortex". The data presented here is based on a total of 26 real-time decoding sessions conducted in two animals (monkey S, 12 sessions; monkey Z, 14 sessions) that were chronically implanted in AIP and F5 with floating microelectrode arrays (FMAs; monkey S, 128 channels; monkey Z, 80 channels). Across these sessions, we recorded in monkey S 827 units in F5 and 899 units in AIP; in monkey Z, we recorded a total of 491 units in F5 and 133 units in AIP. Of these, the vast majority was classified as multi-units (80-85%). Previous studies have already examined in detail the tuning properties of single units in both AIP (Sakata et al., 1995; Murata et al., 2000; Baumann et al., 2009) and F5 (Rizzolatti et al., 1988; Fluet et al., 2010). Therefore the current analyses will focus on the large population of multi unit data that we sampled with chronically implanted FMAs, with a view to comparing and contrasting these results with respect to single unit data recorded previously, and to utilize these signals to maximize the amount of information available for real time decoding.

3.2 Results

3.2.1 Distribution of tuned activity

The distribution of yielded multi-units varied in both monkeys and areas was as follows.

In monkey S, the implanted FMAs yielded an average of 57 (standard deviation 7) and 64 (SD 7) multi-units recorded per session from area F5 and AIP, respectively. These were located on 45 (SD 4) and 46 (SD 3) electrodes in F5 and AIP, respectively, out of a total of 64 implanted in each area. In both areas, approximately 20% of these multi-units showed significant modulation of their firing rate to the factors grip type or orientation during the planning period (two way ANOVA, $p<0.01$).

In monkey Z, we found in F5 on average 29 (SD 8) single- or multi-unit signals per session located on 27 (SD 8) out of 48 implanted electrodes. Of these multi-units, 26% were significantly tuned. In area AIP, fewer implanted electrodes were able to sample neural signals (8 out of 32 electrodes, SD 4; 25%) resulting in a low number of multi-units per session (8, SD 4). Despite the relatively small sample size, the proportion of significantly tuned multi-units was similar to F5 (29%). Overall, the chronically implanted electrodes in both monkeys were able to record samples of tuned multi-units in both areas, indicating that information about grip type and orientation was present in the neural data, which we then used for the real-time decoding experiment.

Figure 3.1 summarizes the spatial distribution of tuned multi unit activity across FMAs, and within each array. For each electrode, we measured the mean number of tuned multi-units per session, and averaged that yield across all sessions. For both monkeys, we found that some implanted arrays sampled more tuned activity than others. For example, in monkey S, FMAs 1 and 4 yielded the most tuned activity on average (yield of 0.29 and 0.37 tuned units per electrode and session) compared to

FMAs 2 and 3 (yield of 0.07 and 0.11, respectively) (Fig. 3.1A).

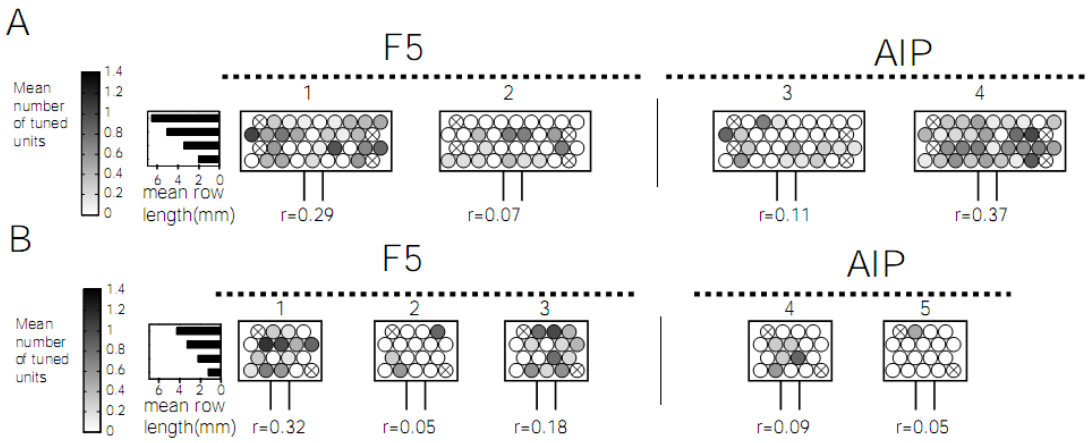

Figure 3-1: FMA electrode layout, dimensions and distribution of tuned multi units across and within FMAs
Hexagonal arrangement of electrodes within each array corresponds to manufacturer specifications. View is "top down" showing approximate locations of electrodes when looking down onto the cortical surface. Electrodes within each row of the array had a variety of different lengths, with the mean length of each row displayed as a bar chart to the left. Electrode color in each plot represents the mean number multi units tuned to either to grip type or orientation (ANOVA, $p<0.01$) obtained across all recording sessions analysed. Crossed-out electrodes were reference or ground with no recording capability. Dashed line represents approximate location of sulcus (arcuate for F5, intraparietal for AIP) relative to FMA. **A**, tuning yield for animal S. **B**, tuning yield for animal Z.

Similarly in monkey Z, the average yield of tuned units was 0.32 and 0.18 in arrays 1 and 3 in F5, respectively, whereas it was only 0.05 in array 2 (Fig 3.1B). Furthermore, FMAs 4 and 5, implanted in area AIP of this monkey, had a relatively low yield of tuned units (average of 0.09 and 0.05, respectively). These results may suggest that we did not achieve optimum placement of all implanted FMAs; rather a subset of the arrays recorded tuned activity regularly from session to session, while others yielded relatively low numbers of tuned units.

Within the arrays that sampled more tuned units, we found that sampling of tuned activity was concentrated on certain individual electrodes or groups of electrodes. For example, in FMA 1 in monkey S, most tuned activity was picked up by two groups of electrodes located at the medial and lateral sides of the array, and relatively

little tuned data was recorded by the electrodes in the middle (Fig. 3.1A). In FMA 4 of the same animal, a "hot spot" for tuned activity was located in the mid-to-lateral section of the array. Thus, there was no consistent spatial pattern in the distribution of high-yield electrodes within each FMA, instead we found a rather heterogeneous distribution of tuned activity across the arrays. This makes sense given their fixed and permanent implantation.

3.2.2 Grasp properties of multi-unit activity

3.2.2.1 Example Units

Figure 3.2 illustrates two example multi-units from F5 and AIP that were modulated by grip type and orientation. Both units were sampled during the standard delayed grasping task as well as during real time decoding. The F5 unit showed a transient increase in firing rate during the planning period relative to baseline activity that was much stronger for power grips than for precision grips (Fig. 3.2 A). This unit was therefore strongly tuned for grip type during the planning period. Consistent with this tuning, the unit also showed a clear burst of firing around the time of grasping for power grip trials. However, modulation by orientation was absent. The observed strong modulation by grip type has been well characterized in previous single-unit studies of F5 (Murata et al., 1997; Raos et al., 2006; Umilta et al., 2007; Fluet et al., 2010).

The example unit from AIP (Fig. 3.2 B, D) had a different activity profile compared to the F5 unit, with a much stronger modulation by handle orientation. It showed an increase of its firing rate immediately after presentation of the instructed grasp (cue period), with increased firing for right-ward handle positions (+50° and +25°). This tuning for orientation was sustained throughout much of the planning period, and was somewhat clearer for precision grip trials than power grip trials.

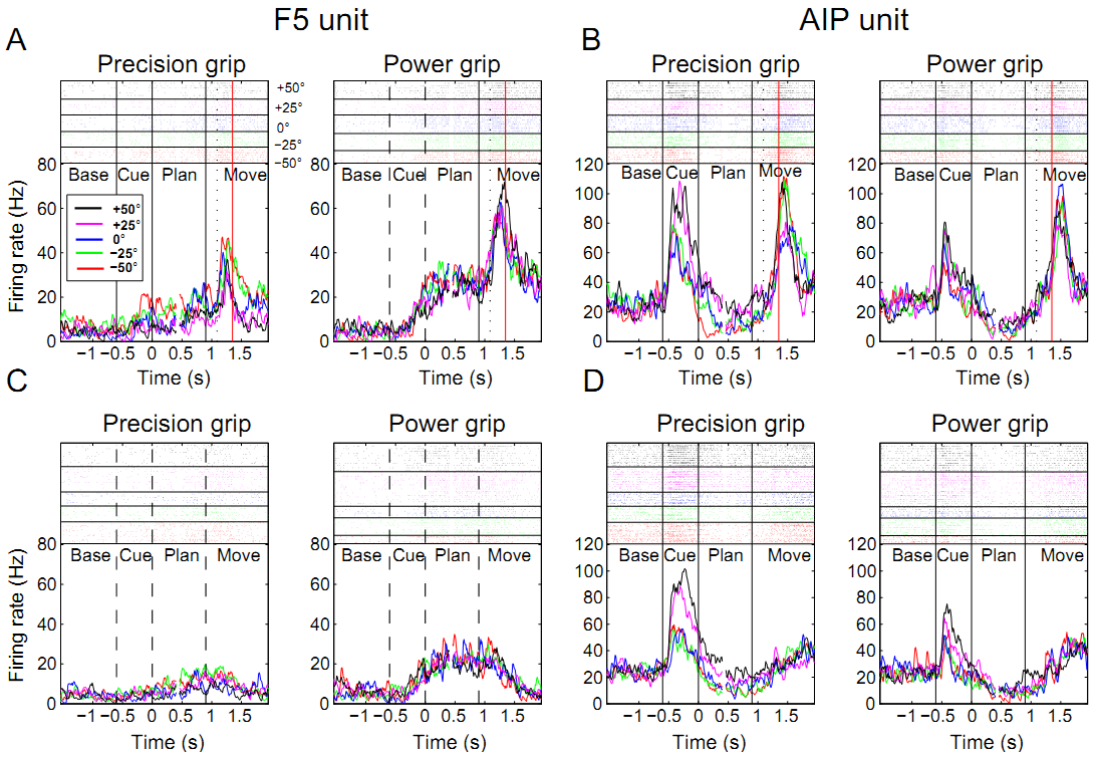

Figure 3-2: Firing rate histograms and raster plots of an example multi unit from F5 and AIP, during the delayed grasping task and during the brain control task

Firing rate histograms and raster plots of two example multi-units from F5 and AIP, during the delayed grasping task and the brain control task. Each panel A-D shows precision grip trials (left) and power grip trials (right) separately. Each color represents a particular handle orientation in the spike rasters (on top) and for the averaged firing rates (at bottom). Dashed line within movement epoch represents mean time of hand rest release, solid red line indicates mean time of handle contact. All trials are doubly aligned to the end of the cue epoch (at 0 s) and the start of movement (at 0.9 s), gaps in the curves and rasters (at ~0.4 s) indicate realignment. **A.** Multi-unit recorded in F5 in the delayed grasping task showing tuning for grip type during the planning period, with greater firing rates for power grip trials than precision grips. **B.** Multi-unit recorded in AIP in the delayed grasping task showing orientation tuning during cue, planning and movement. **C.** Activity of the same F5 unit as in A, but during brain control trials. Note the similar activity modulation for grip type during the planning phase, but a lack of movement-related activity. **D.** Activity of the same AIP unit as in B during brain control trials. Note the separation of average firing rate profiles according to handle orientation during the cue and planning epoch and the absence of movement-related activity.

Finally, the unit showed a burst of activity around the time of movement execution with a similar modulation by object orientation and grip type. This rather complex firing pattern, i.e., simultaneous modulation by orientation and grip type, is consistent with previous observations in AIP (Baumann et al., 2009).

Figures 3C and 3D show the activity of the same F5 and AIP multi-units during real time decoding trials. For the F5 unit, we observed the same grip type modulation during the planning period. Note however, the absence of an activity "peak" during movement execution, since no grasp was actually performed. The AIP multi-unit also showed broadly similar activity during decoding trials, maintaining its preference for right-ward handle positions especially during precision grips. As in F5, movement-related activity was clearly absent. Overall, these data demonstrate that information about grip type and orientation were present at the multi-unit level in F5 and AIP during the planning period, including during decoding trials, when no movement was made.

3.2.2.2 Population Activity

Similar findings were observed at the population level. Figure 3.3 shows the population firing rates across all 244 tuned multi units from area F5 and 210 multi units from AIP of both monkeys, separately for each unit's preferred and non-preferred grip type and orientation. In both areas, the mean firing rate of the population was modulated by both grip type and orientation, starting shortly after the beginning of cue presentation and lasting until the end of movement execution. As was observed for individual units, important differences could be seen between the two areas. In F5, cue-related activity was relatively small, while there was a large peak of activity during movement execution. Furthermore, the key parameter that modulated the firing rate curves during the planning period was the preferred grip type; note that in particular the mean firing rate for preferred grip type. Note that in particular the mean firing rate for preferred type / non-preferred orientation condition (blue curve) was higher than for the non-preferred grip type and preferred orientation (green curve). This effect became even more pronounced during movement

execution, where the four curves clearly separated into a preferred grip type (blue and red) and a non-preferred grip type group (green and black).

In the population of 210 multi units from AIP, cue-related activity in AIP was more pronounced, while movement-related firing (although still present) was weaker (Fig. 3.3 B). In contrast to F5, both the preferred orientation and the preferred grip type modulated the mean activity in AIP. This separation of the curves began early in the cue period, and persisted throughout the planning phase and movement execution.

In summary, F5 showed moderate tuning to orientation early in the task and strong grip type tuning with a peak during movement execution, while AIP showed a clear tuning for grip-type and orientation throughout the task. These data indicate that distinct representations of both task parameters exist in the multi-unit activity of both areas during movement planning, which we used for real time decoding.

Figure 3.3 C-D shows the population firing rate of the same multi units during real time decoding. As for individual multi units (Fig. 3.2), the population activity showed similar tuning to grip type and orientation in both areas during the cue and planning epochs in both tasks, even though the monkey did not actually perform a grip (note the clear absence of movement-related activity). This clear modulation of firing rate by the task in the absence of movement execution provided the means by which we were able to decode grasp in real time using only activity from the planning epoch.

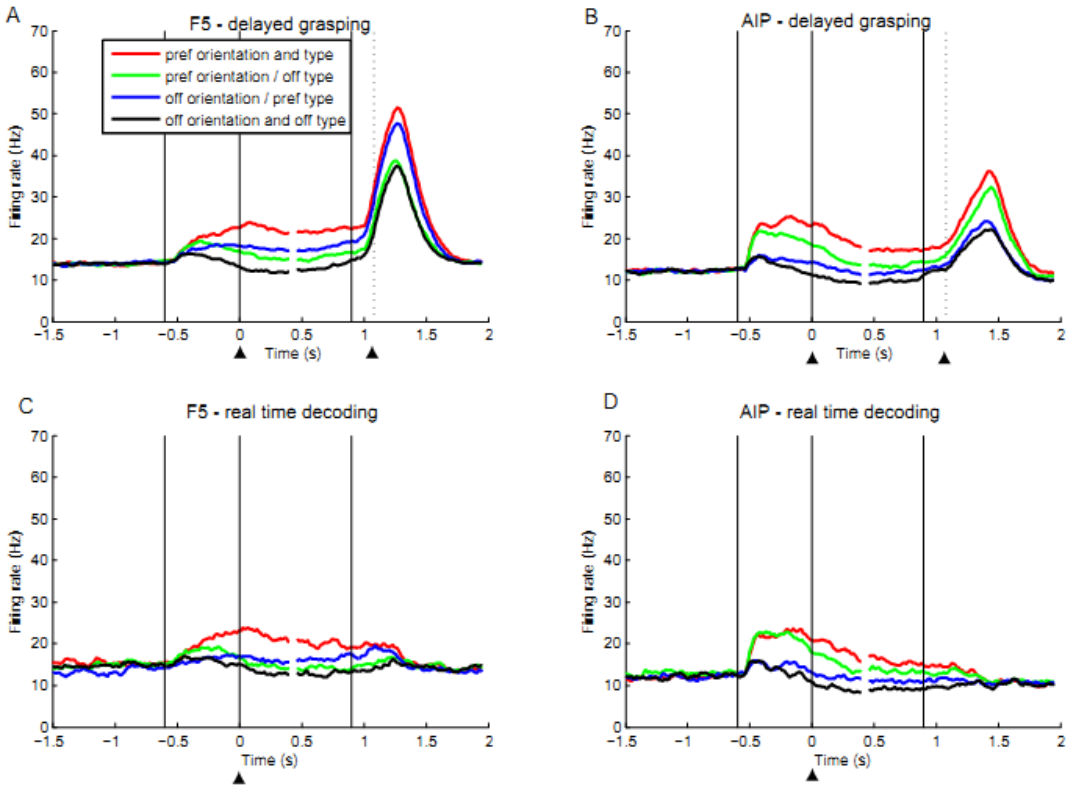

Figure 3-3: Population firing rate activity
Average activity of multi-units from F5 (left column, n=244) and AIP (right column, n=210) in the delayed grasping task (top row) and the brain control task (bottom row) is presented for each combination of the unit's preferred and non-preferred grip type and handle orientation. Epoch definitions as in Fig. 3.2. Dashed line within movement epoch represents mean time of hand rest release.

3.2.2.3 Multi-unit coding properties

To further investigate the tuning properties of F5 and AIP multi-units, we examined the distribution of preferred grip types and orientations across all recorded multi units during the planning period (Fig. 3.4).

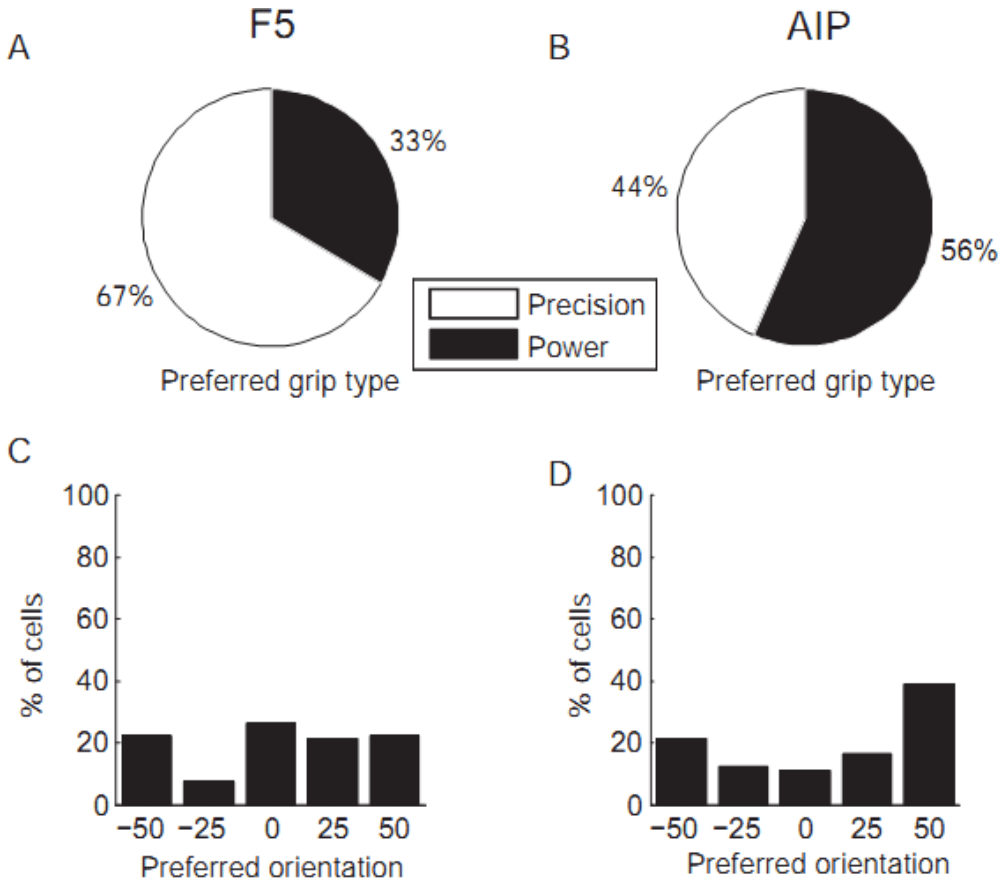

Figure 3-4: Distribution of preferred grip type and orientation during the planning epoch for F5 and AIP multi units
A, ratio of F5 multi units preferring precision grip (white) vs power grip (black). A large majority of cells preferred precision grip in comparison to power grip. **B**, ratio of AIP multi units preferring precision vs power grip; here the proportions of units preferring each grip type were more equal. **C**, distribution of F5 multi units preferring each of the five orientations. **D**, distribution of AIP multi units preferring each orientation. In comparison to F5, AIP units showed a preference for the extreme handle orientations ($\pm 50^{o}$).

In area F5 the majority of cells (67%) had a preference for precision grip (Fig 3.4 A), which is consistent with the finding that more complex grip types tend to be over-represented in motor areas (Muir and Lemon 1983, Umilta et al 2007).

However, in contrast to our previous single-unit work in F5 (Fluet et al 2010), we did not observe a clear preference for extreme handle orientations in our data during the planning epoch. Instead, the distribution of preferred orientations across the population was relatively uniform (Fig. 3.4 C). This difference could be due to the averaged nature of multi-unit data in the present study.

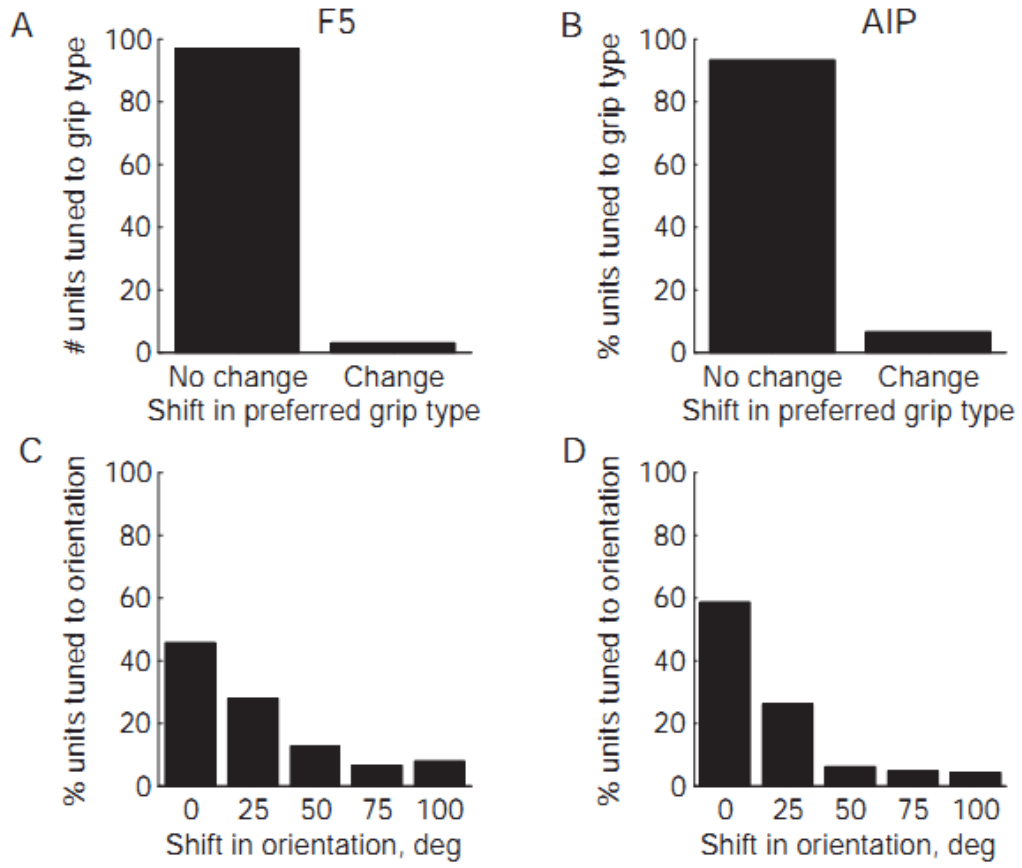

Figure 3-5: Consistency of tuning preferences during the planning epoch across the delayed grasping and brain control tasks
A, Consistency of grip type tuning in F5. Histogram indicates the proportion of multi units that maintained the same preference for grip type during brain control that they displayed during delayed grasping, versus those that changed preference to the opposite grip type. **B**, consistency of grip type tuning in AIP. In both areas, the vast majority of units did not change their preferred grip. **C**, change of orientation tuning in F5 between delayed grasping and brain control. Histogram shows the proportion of units for which the preferred orientations in the two tasks were the same (0^o shift), were highly similar (25^o shift) or were further apart ($50\text{-}100^o$ shift). **C**, change of orientation tuning in AIP. Shifts of preferred orientation >25 in either area were less frequent. Overall, planning epoch tuning properties of F5 and AIP multi units were similar in the two tasks.

In AIP the distributions of preferred grips and orientations were somewhat different to those in F5. Firstly, there was no clear preference for either grip type in the population during the planning period (Fig. 3.4 B). Secondly, the majority of AIP multi units (60%) coded predominantly for the extreme handle orientations during planning (Fig. 3.4 D), which was in strong contrast to the F5 data. Together, these

population data confirm that both grip type and orientation were well represented in the multi unit activity during the planning period of the delayed grasping task.

This tuning pattern during movement planning remained essentially constant while the monkey performed the real time decoding task (Fig. 3.5). For all tuned units of the delayed grasping task, the preferred grip type and the preferred orientation were also measured from the mean firing rate during movement planning in the real time decoding task. In both areas F5 and AIP, only a small minority of multi units showed a change in preferred grip type (8% of cells) (Fig. 3.5 A,B), and the preferred orientation stayed either the same or shifted by a single position at most; few cells showed larger shifts in their preferred handle position (Fig. 3.5 C,D). Thus the patterns of spiking activity across the population of cells in the two areas remained similar during real grasping and grasp decoding, and could be reliably accessed for decoding in real time.

3.2.3 Online grasp decoding

Representative performances during a single real-time decoding session are shown in Figure 3.6 for monkeys S and Z, respectively.

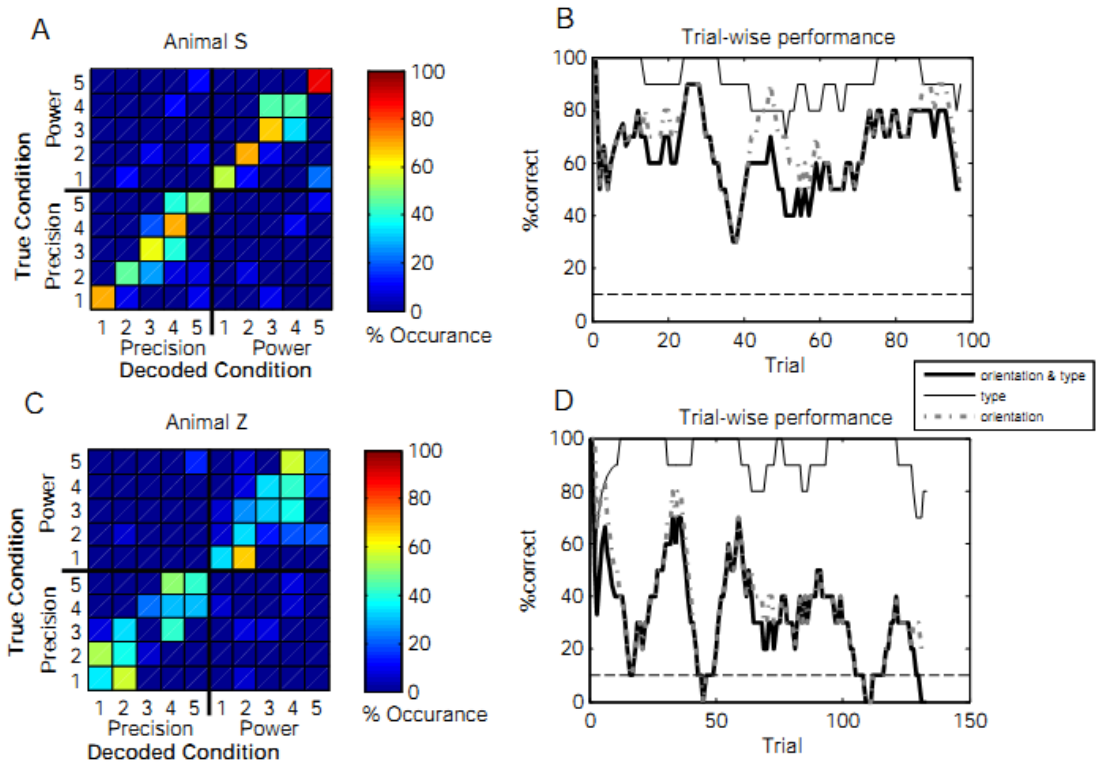

Figure 3-6: Example real time decoding performance measured during a single session

Data are shown separately for animal S (**A-B**) and animal Z (**C-D**). **A**, decoding predictability of grip type (precision vs. power) and handle orientations (here labeled 1-5), i.e. 10 grasp conditions in total. Color code of confusion matrix indicates percentage of occurrences during which a given condition was classified as one of the 10 grasp conditions by the Bayesian decoder. Correct classifications therefore line up along the diagonal. Note that classification errors were mainly made between neighboring orientations but rarely between grip types. **B,** trial-wise classification performance for the same session analyzed in A. Classification accuracy was measured using a sliding window which averaged the performance of the previous ten trials. Post-hoc, performance was measured separately for all ten conditions (thick black line), grip type only (thin black line) and orientation only (dash-dot line). Note however that the monkey always performed the full 10 condition decoding task. Dashed line represents chance level for 10 condition decoding. Temporary performance decreases were mainly due to errors in orientation classification rather than grip type. **C,D**: same analysis for a single session recorded in animal Z.

The output of the decoder is compared to the instructed condition for all trials during this session using a confusion matrix (Fig 3.6 A, C; see *Methods* for details). Correctly classified trials appear along the diagonal of this matrix. Using multi-unit spiking activity from F5 and AIP during the planning period, grip type and orientation were decoded with an overall mean accuracy of 62.0% in monkey S and with an accuracy of 30.8% in monkey Z. For the latter animal, it can be seen from the confusion matrix that the main type of error responsible for this somewhat low performance was confusion of neighboring orientations, while grip type was rarely misclassified. Figure 3.6B shows the trial-by-trial success rate of the decoder during the same session, measured with a sliding window (see: *Methods*). Although there was some variability in performance, trials were generally decoded above chance level (10%). Decoder performance was significantly better in monkey S; again, there was almost no confusion about the grip type, but in contrast to monkey Z, a much better classification of orientation was observed (Fig. 3.6C). The trial-wise performance of the decoder was also more stable in comparison to the previous animal, and remained well above chance throughout the session (Fig. 3.6D).

Figure 3.7 summarizes the real-time decoding performance of both animals across all sessions. Average confusion matrices for monkey Z and S are shown in Fig. 3.7A-B. The same qualitative features of decoding performance exhibited within the example sessions were also observed in the average across all recordings: classification of grip type was highly accurate in both monkeys, while decoding of orientation was less so, and was particularly poor in monkey Z. Decoding accuracy across sessions is given in Fig. 3.7C for each animal. It remained consistently above chance level for both animals with a mean performance of 50.4% (±7.6%) in monkey S and of 33.5% (± 5.9%) in monkey Z. This difference was significant ($p<10^{-5}$, 2-tailed t-test).

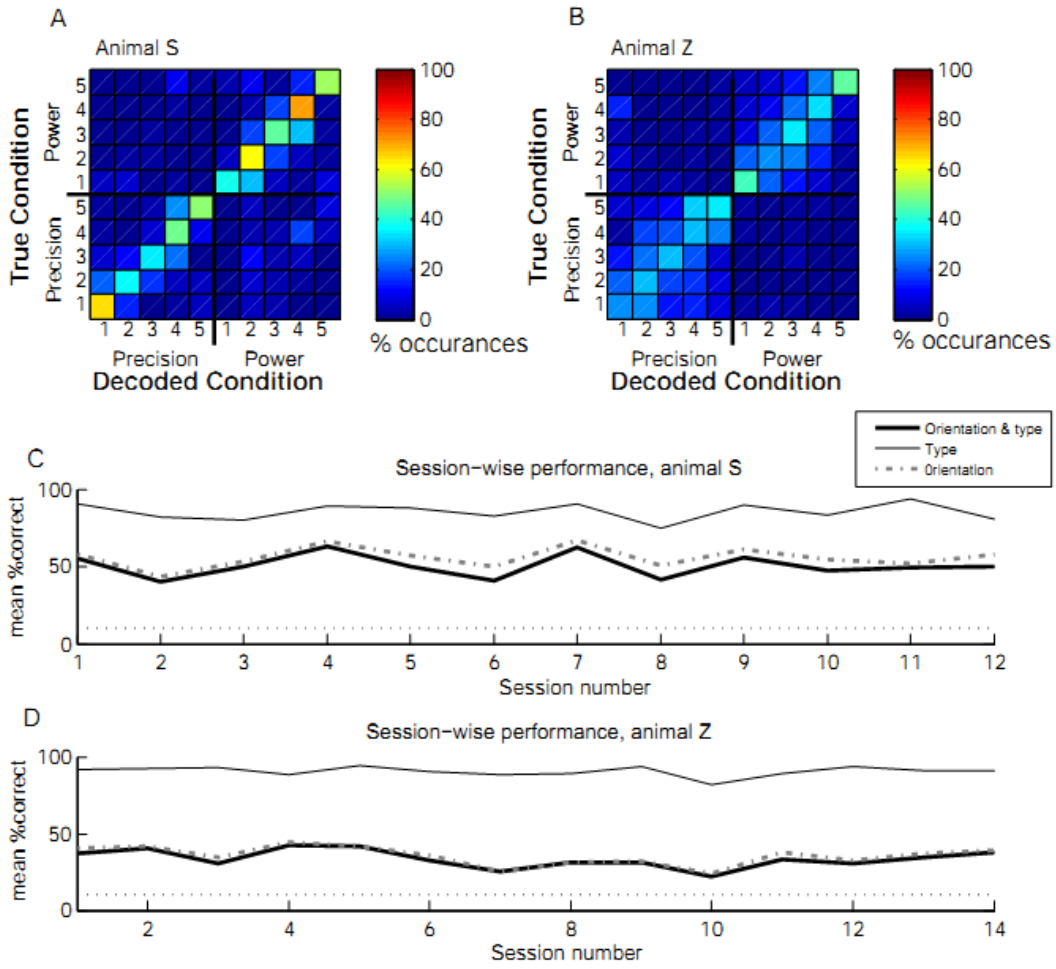

Figure 3-7: Real time decoding performance for each animal, population results **A.** average decoding predictability across 12 sessions for animal S, computed by averaging individual confusion matrices across sessions. Conventions same as in fig 3.6A. **B.** average decoding predictability across 14 sessions in animal Z. Note that in comparison to animal S, decoding in this animal showed low predictability of the 5 handle orientations, but high predictability of grip type. **C.** session-wise classification performance for animal S across 12 sessions, computed by averaging across the diagonal of each session's confusion matrix. Each point represents the mean percentage of correctly classified trials. As in figure 7, classification performance was measured separately post-hoc, for all ten conditions, for grip type decoding only, and for orientation decoding only. As evident in the confusion matrix, decoding performance was limited by errors in orientation decoding, while grip type classification was highly accurate. **D.** similar analysis for 14 sessions recorded in animal Z. Horizontal dashed line in C and D represents chance level for 10 condition decoding (10%).

To gain more detailed information about decoder performance, we quantified the decoding accuracy separately for grip type and orientation classification (Fig. 3.7C,

D, solid black and dotted curves). As expected, classification of the grip type was always highly accurate in both monkeys (monkey S mean: 85.5%, monkey Z mean: 90.6%). Accuracy was in fact slightly, yet significantly higher for monkey Z than for monkey S (two-tailed t-test, $p<0.01$). In contrast, decoding of orientation was performed with less accuracy in both animals, which is not surprising, given that there were more orientation (5) than grip type conditions (2) to classify. However, there was a considerable performance difference between the two animals; while orientation could be classified with an accuracy of 56% in monkey S; it was only 35.5% in monkey Z, very similar to the overall observed performance of 33.5%. This suggests that in monkey S, information about both grip type and orientation was available to the decoder, while the decoding of orientation information was rather poor in monkey Z.

3.2.4 Offline decoding

To investigate how these differences in classification of grasp parameters were related to the information available within each cortical area, we evaluated the decoding performance in an offline analysis separately for AIP and F5 and in combination with improved (offline) spike sorting.

As a first step, we investigated by how much the decoding performance improved if we replaced the real-time spike sorting procedure (NEV) with an offline, optimized spike sorting method (super-parametric clustering, SPC, see *Methods*). SPC can provide high-quality spike sorting but is currently not implemented to run fast enough and unsupervised in real-time mode. Figure 3.8 directly compares the decoding performance based on online (NEV) and offline (SPC) spike sorting.

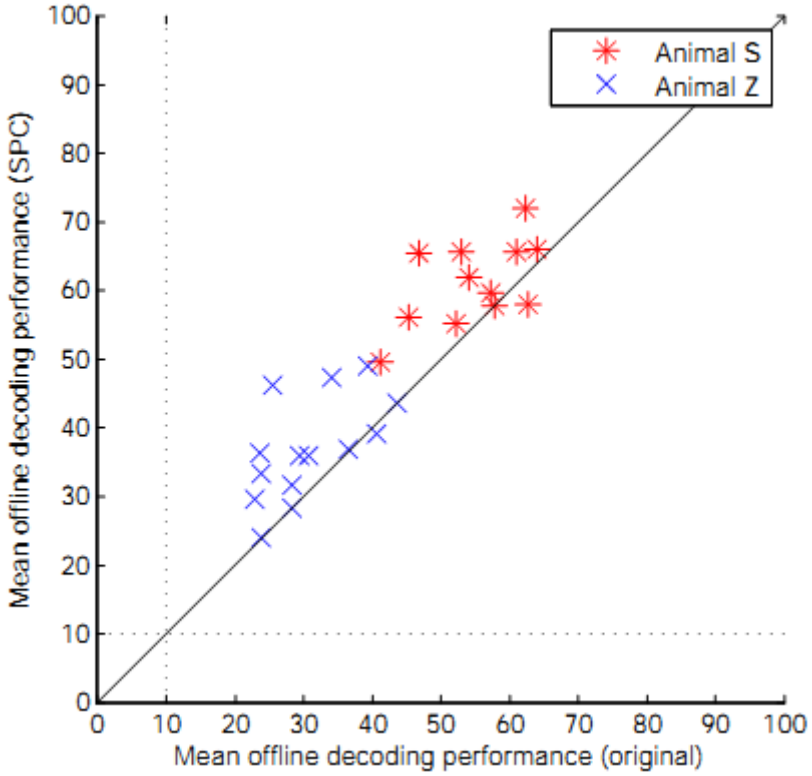

Figure 3-8: Scatter plot comparing mean simulated Naïve Bayesian decoding performance achieved using spike data sorted online, versus performance attained offline with spike data sorted offline in Waveclus using superparamagnetic clustering (SPC)
Data from both animals is shown together. Diagonal line indicates unity. Most points fall above this line, but close to it, indicating that decoding performance was marginally improved by performing spike sorting offline. Mean increase = 6.25%.

In both animals, decoding performance for SPC vs. real-time sorting increased in 24 out of 26 decoding sessions with a mean increase of 6.2 % (monkey Z) and 6.3 % (monkey S). Although these were modest performance gains, the difference was significant (two-tailed matched-samples t-test, $p<1x10^{-4}$).

Secondly, we decoded grip type, orientation, or all 10 grasp conditions separately from either area or from both areas combined using the Matlab implementation of the same Naïve Bayesian classifier that we have used online. Figure 3.9 summarizes the results of this analysis and compares them to the performance of the real time experiments. Certain key differences between F5 and AIP became readily apparent. For the decoding of grip-type alone, performance was better when using only spiking data from F5 than from AIP only (Fig. 3.9A-B); in monkey Z, grip type classification

was 90.9% accurate (±6.1%) for NEV data, compared to 63.6% (SD 8.2%) accuracy in AIP. In monkey S, AIP performed slightly better with 74.7% (SD 4.4%) but was still worse than F5 with 87.9% (SD 3.6%). We observed that performance was significantly higher using F5 alone compared to AIP alone, and when using both areas combined versus AIP alone. The same effect was observed using data sorted in Waveclus ('SPC', white bars, $p<0.01$). Furthermore, decoding with information from both areas together did not lead to significantly better performance than using F5 alone, neither for NEV nor SPC data. Finally, we did not see a significant performance increase when using SPC data, with the exception of area AIP in monkey S (Fig. 3.9B); here, grip type decoding accuracy increased significantly to 81.9% (SD 6.5%) for SPC data (paired t-test, 2 tailed, $p<0.01$), corrected for multiple comparisons. To summarize, these results suggest that maximum classification of grip type could be achieved using data from F5 alone, and it was not possible to extract further information via offline spike-sorting routines or by the inclusion of AIP activity.

In contrast to grip-type decoding, results obtained during orientation-only decoding were more accurate when using data from AIP than F5 (Fig. 3.9C-D). This effect was strongest in monkey S, with decoding performance averaging 53.9% (SD 7.7%) using AIP NEV data, as compared to only 42.7% (SD 8.0%) using F5 NEV data. Here, combining signals from both areas gave the most accurate orientation classification (57.9%, SD 7.7%), although this difference was not significant compared to performance with AIP alone (2-tailed t-test, $p>0.05$). The same effect of area was observed using SPC data. In addition, there was a marked increase in decoding accuracy using the offline-spike-sorted data in monkey S, compared to online-spike-sorted data (NEV). This was true for both AIP, and for AIP and F5 combined; using SPC data increased decoding performance by 8.6% (AIP) and 8.7% (AIP+F5) from what was observed using NEV data. (For AIP+F5 a similar increase was seen in comparison to real-time decoding performance). In contrast, there was no performance increase for F5 (Fig. 3.9D).

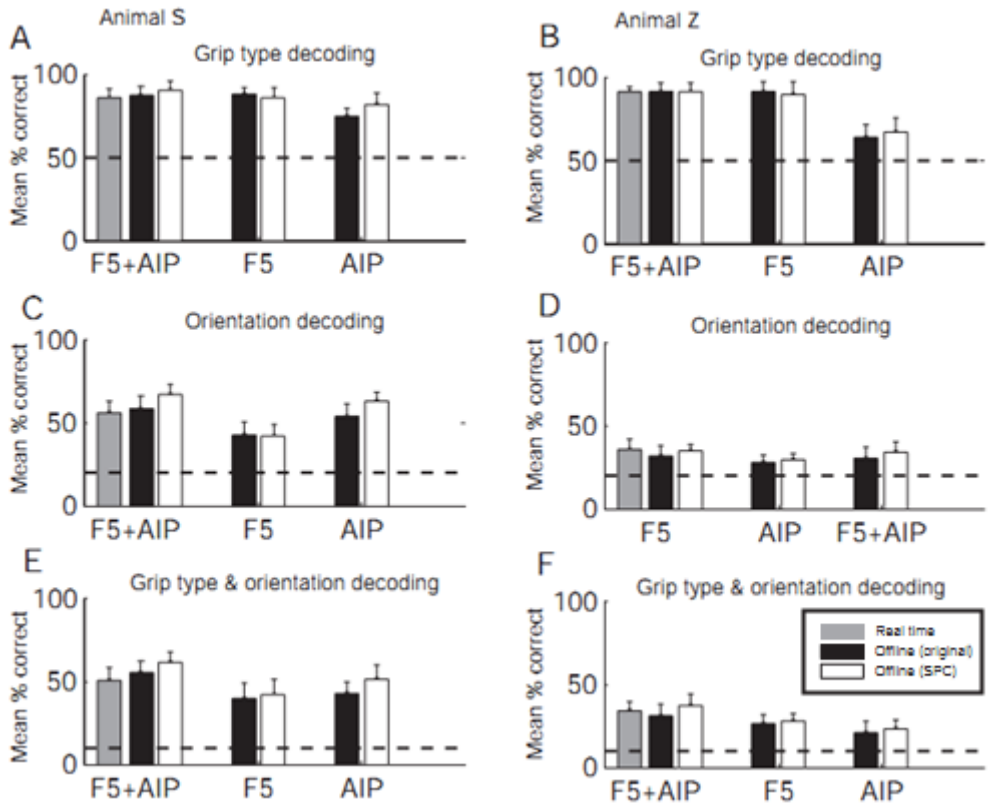

Figure 3-9: Summary of decoding performance results
Each bar chart shows the mean simulated classification accuracy across all sessions, analysed separately for different spike sorting methods (online vs SPC), decoding types (grip type vs orientation), and cortical areas (AIP vs F5). Actual observed real time decoding results are included for comparison (gray bars). Respective chance levels are indicated with dashed lines. Error bars represent standard error. **A,** grip type decoding performance in animal S. Classification of grip type was performed at high accuracy, particularly when using data from F5. Offline spike sorting failed to significantly increase decoding performance. **C**, orientation decoding performance in animal S. AIP performed significantly better than F5. **E**, decoding performance for all 10 grasp conditions. In both C and E, performance was greater using data from both AIP and F5 than using either area alone. In addition, application of SPC spike sorting resulted in clearer performance gains. **B,D, F**, same analysis for monkey Z.

In monkey Z, accuracy of orientation decoding from AIP alone was somewhat lower (30.3%±6.8%), but remained greater than from F5 (27.5% ±5.2%). No significant difference was found between accuracies for the different areas, or for both areas combined. Finally, in contrast to monkey S, little improvement was observed for decoding using SPC data (Fig. 3.9C).

Simulated decoding of the full 10 conditions produced similar effects to the orientation-only decoding, with the addition of an overall reduction in accuracy due to the more complex nature of the classification required. In monkey S, mean simulated performance using F5 alone was relatively poor (39.5%, SD 9%), which could be increased slightly to 42.1% using SPC data (Fig. 3.9F). AIP accuracy using NEV data alone was greater than F5 (42.4%, SD 7.2%). However, incorporating offline spike sorting in this area resulted in a much larger performance increase of 8.5% (to 51.0%, SD 8.4%). For monkey Z, the reverse was true: the decoder performed better using data from F5 alone in comparison to AIP (25.9%, SD 5.9% versus 21.0%, SD6.9% respectively). Performance levels using either area alone were low, and only minimal improvements were observed using offline spike sorting.

In both animals, we found the best decoding performance, whether real-time or offline, by combining data from both areas. In monkey S, the mean real-time decoding performance measured across sessions (in which we used data from both areas by experimental design) was 50.4% with SD 7.6%. Offline performance using NEV data was slightly higher (54.8%, SD 7.4%), and the best observed performance was for SPC data (61.1%, SD 6.2%). Unlike the previous simulated decodes, where performance using the combined information from AIP and F5 was always statistically indistinguishable from the performance achieved with the best area alone (F5 for grip type, AIP for orientation), we found that performance with AIP and F5 combined was significantly better than performances achieved with either area alone. This was true for both NEV data and SPC data (1-way ANOVA, $p<0.001$).

A similar trend was observed in monkey Z, albeit with lower overall performance levels. Mean real time decoding accuracy in this animal was 33.7% (SD 5.9%). Simulated decoding performance using combined data and online spike sorting (NEV) was slightly lower than the real time results with 30.7% (SD 7.0%). Overall best performance was again achieved using combined F5+AIP data that was spike sorted offline (36.9%, SD 7.6%); as in monkey S this was significantly higher than performance using data from either area, although only for offline spike sorting.

Taken together, these results indicate that F5 tended to perform better than AIP at grip type-only decoding, and AIP consistently outperformed F5 during orientation-

only decoding. However, utilization of data from both areas was necessary for optimal decoding accuracy of grip type and orientation in the 10-condition task.

3.2.5 ROC analysis

As mentioned above, clear differences in the decoding accuracy of grip type and orientation were observed between AIP and F5 at the population level. To quantify the extent to which these differences could be observed in the activity of individual units, we carried out an ROC analysis to quantify the classification accuracy of each multi unit for grip type and object orientation (Fig. 3.10). Only multi units which were significantly tuned to either parameter in the 2-way ANOVA were included in this analysis (since non-tuned cells would tend to perform at chance levels). Results were comparable for the two animals and are therefore combined here. Each histogram shows the area under the curve (AUC) values across the population (F5: 244 units, AIP: 210 units) separately for significant and non-significant AUC values (Monte Carlo analysis, see: Methods). For grip type, the majority of cells from F5 and AIP were able to significantly distinguish between power and precision grip based on their spiking activity during the planning period (95 % of F5 units and 83 % of AIP units respectively) (Fig. 3.10A-B). Note however that in F5, a significant portion of the AUC distribution was skewed towards larger AUC values, indicating that F5 units perform the classification with higher accuracy than AIP units. This difference in mean AUC value between F5 and AIP for the significant units was highly significant for grip type classification (2-tailed t-test, $p<1x10^{-6}$).

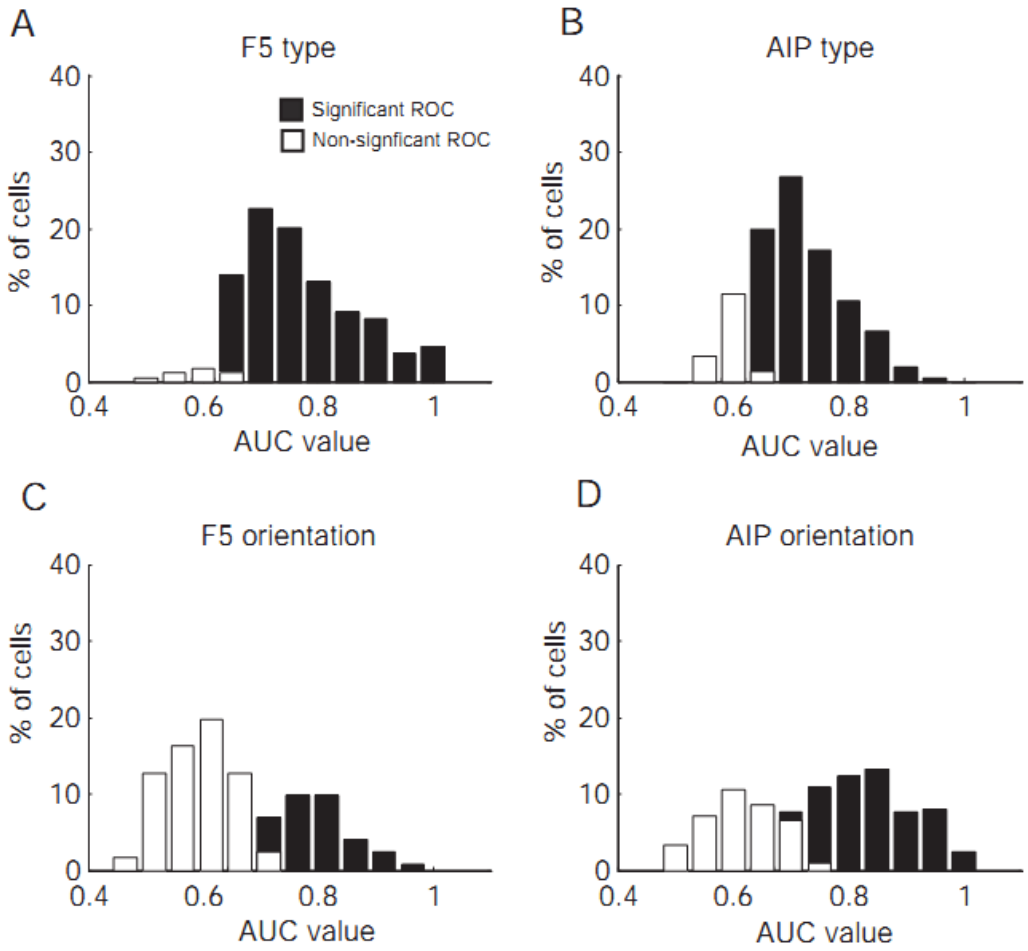

Figure 3-10: ROC analysis of classification accuracy for F5 and AIP multi units
The ability of each significantly tuned multi unit to correctly classify grip type (precision vs. power) and handle orientation (preferred vs non-preferred) was measured as a function of the area under the curve (AUC) value in the ROC analysis. Each histogram shows the distribution of AUC values across the population; significant and non-significant AUC values (Monte Carlo analysis) are represented by white and black bars respectively. **A**, grip type classification in F5. 95% of tuned F5 multi units could classify grip type with significant accuracy. **B**, grip type classification in AIP. AUC distribution was less skewed towards higher AUC values, indicating lower accuracy of grip type classification in AIP compared to F5. **C**, orientation classification in F5. In contrast to grip type, F5 multi units performed relatively poorly at orientation classification; the majority of AUC values in the distribution were non significant. **D**, orientation classification in AIP. Unlike F5, the majority of AIP AUC values were significant and skewed towards the right, indicating high classification accuracy for orientation. Note that in both AIP and F5, AUC distributions for orientation were bimodal.

In contrast, we observed a bimodal distribution of AUC values for orientation classification. On the basis of AUC values, each population of multi units could be divided into significant and non-significantly tuned cells (Fig. 3.10C-D). This separation was much more obvious in F5, where the largest peak comprised non-significant cells; only a minority of cells (34 %) classified object orientation reliably. In AIP, the reverse was true: a clear majority of cells (63%) had ROC values significantly larger than 0.5, and the mean AUC values of significantly tuned cell was significantly larger in AIP (0.83, SD 0.08) as compared to F5 (0.79, SD 0.06; 2-tailed t-test, $p<0.01$).

These findings complement our real-time decoding results and demonstrate that the observed differences between F5 and AIP units for the real-time decoding of grip type and orientation are reflected in the coding properties of individual units and do not originate from a skewed representation of AIP and F5 units in our sampled population.

3.3 Discussion

In this chapter we have demonstrated the real time decoding of grip type and orientation in macaque monkeys using neural activity recorded from F5 and AIP, two "higher order" areas involved in the planning of grasping movements. Previous studies from our laboratory have already investigated the encoding of these parameters by populations of neurons in AIP and F5 during a standard delayed grasping task (Baumann et al 2009; Fluet et al 2010). This study demonstrates the logical converse of these findings: sufficient information about these variables is available during the planning of grasping movements to enable them to be interpreted in real time and fed back to the monkey, circumventing the execution of the actual grasp. The fixed nature of the chronically implanted FMAs precluded optimized isolation of single units during recording, resulting in the main bulk of the sampled activity being multi-unit in nature (Fig. 3.1). However, although decoder performance could be further improved using optimized spike sorting, this effect was

small (Fig. 3.9, 3.10). The multi unit activity that was recorded showed broad similarities in tuning properties to findings from previous work. Both F5 and AIP contained distinct representations of grip type and object orientation during the planning period of the standard delayed grasping task (Fig. 3.2-3.4). However, important differences existed in these representations between the two areas. At the population level, precision grip was over-represented in F5, but not in AIP (Fig. 3.5). On the other hand, AIP cells tended to prefer more extreme orientations whereas F5 showed a more uniform distribution of preferred orientations. These tuning preferences were maintained by the cells during real time decoding, when no movement was executed (Fig. 3.6). A post-hoc analysis of decoder performance found that grip type was consistently decoded with a high level of accuracy (>90%), while performance in orientation was somewhat reduced (figure 3.8). Offline simulations demonstrated an effect of cortical area on decoder performance. On average, signals from F5 resulted in the highest accuracy for grip type decoding, and concomitantly the lowest accuracy for orientation decoding. In contrast, activity recorded from AIP yielded better decoding of orientation together with highly accurate grip type decoding (Fig. 3.9). In line with this, an ROC analysis of tuning sensitivity of F5 and AIP multi units found that the strength of tuning within F5 neurons for grip type was significantly greater than for orientation, while the opposite was true for AIP (Fig. 3.11). Overall, these results demonstrate the real time decoding of intended grasping goals using multi-unit signals from higher-order motor areas obtained during planning of these movements, and underscore the importance of utilizing signals from multiple cortical areas for control of BMIs to restore movement function.

Comparison to previous work

Research in the field of BMIs for motor control has seen rapid expansion in recent years (Scherberger 2009). A key approach has been the closed-loop decoding of 2D and 3D arm and hand trajectories, derived mainly from M1 (Serruya et al 2002; Taylor and Schwartz 2002; Hochberg et al 2006; Wolpaw and McFarland 2004), to control robotic arms for grasping objects (Taylor et al 2003; Carmena et al

2003; Velliste et al 2008), or to decode individual finger movements (Aggarwal et al 2009). A key difference in the present study was our decision to target parietal and frontal areas which are thought to play a role in sensorimotor integration during movement planning (Andersen et al 2002,2004,2010; Scherberger et al 2005; Pesaran et al 2006; Cisek and Kalaska 2005). This was motivated by two factors. Firstly, the use of these cortical structures as a signal source for BMI applications has already proven to be a viable tactic for decoding hand and arm reaching movements, using parietal reach region (PRR) and dorsal premotor cortex (PMd) (Andersen et al 2004; Shenoy et al 2003; Musallam et al 2004; Scherberger et al 2005; Santhanam et al 2006; Mulliken et al 2008). Secondly, these areas contain more abstract representations of intended movement goals that can be extracted directly, circumventing the need to decipher low-level M1 movement control signals (Shenoy et al 2003; Andersen et al 2004, 2010). Our work therefore extends this avenue of research by including real time decoding of hand grasping movements via the selection of the equivalent parieto-frontal circuits for grasping (AIP and area F5).

There is substantial evidence that movement representations in AIP and F5 are also abstract (Scherberger 2009). Neurons in AIP represent grasping movements in terms of the visual properties of target objects (Taira et al 1990; Sakata et al 1995; Raos et al 2006; Baumann et al 2009), and like other parietal areas, also contain context-related information for the appropriate selection of actions (Baumann et al 2009; Gail and Andersen 2006; Scherberger and Andersen 2007). Similarly, area F5 contains neurons which respond to the visual properties of objects to be grasped well in advance of movement execution (Murata et al 1997, Raos et al 2006, Umilta et al 2007, Fluet et al 2010), such that the representation is based more in an extrinsic coordinate frame compared to a more intrinsic (or muscle-like) representation in M1 (Kakei et al 1999, 2003; Morrow et al 2007).

In line with these findings, we observed a clear tuning of neural activity in AIP and F5 to grip type and object orientation, which lasted throughout the planning period of standard delayed grasping trials (Fig. 3.3, 3.5) and was further confirmed by a post-hoc ROC analysis of tuning sensitivity (Fig. 3.11). We trained a simple Bayesian classifier using this planning activity, which was able to reliably predict the

upcoming grasp movement and orientation on a single trial, before the actual grasping movement was executed (Fig. 3.7, 3.8). Thus, our study extends the substantial body of work on decoding of movement intentions to incorporate real time control of hand grasping.

Nature of the decoded information

Decoding was always carried out using spiking activity collected exclusively during the planning period, while the monkey's hands were still positioned at rest; initiation of reach to grasp before decoding was completed automatically resulted in termination of the trial. We frequently observed the monkey perform successive trials using brain control, without initiating reach-to-grasp (data not shown). We therefore suggest that the decoded information explicitly represented the planned or intended grasping movement, rather than a motor output signal (Scherberger and Andersen 2007). Two lines of reasoning support this conclusion.

Firstly, our instructed delay task is a variant of a widely used standard paradigm for investigating planning and working memory, in premotor cortex (Cisek and Kalaska 2004; Messier and Kalaska 2000; Fluet et al 2010) and posterior parietal areas (Snyder et al 1997; Gail and Andersen 2006; Baumann et al 2009). Secondly, the sustained instructed delay period activity elicited by this kind of task has been well studied, and is thought to be related to the preparation or planning of intended movements (Alexander and Crutcher 1990; Riehle and Requin 1993; Snyder et al 1997; Padua-Schuoppa et al 2002; Churchland et al 2006a,b). In particular, preparatory activity in F5 and AIP has been shown to represent intended grasping movements (Raos et al 2006; Umilta et al 2007; Baumann et al 2009; Fluet et al 2010). Our observation of significant tuning for grip type and orientation in AIP and F5 during the delay is fully compatible with these results. The present work extends these findings by demonstrating that this grasp planning activity does not require an actual grasping movement to be executed (Fig. 3.4), in accordance with earlier work on eye and reaching movements in PPC (Snyder et al 1997). It is precisely these characteristics of premotor and parietal cortical areas that make them useful sources of control signals for neural prosthetics (Shenoy et al 2003).

Nonetheless, it is possible that this activity may have reflected the sensory properties of the cue, such as the color of the LED instructing grasp type. Arguing against this interpretation, however, is the fact that spiking within AIP and F5 showed sustained modulation for several hundred milliseconds after the end of the cue period (Fig. 3.3, 3.4). This is well outside the range over which one would be expected to see transient neural activity triggered by visual stimuli (Schmolesky et al 1998; Thorpe et al 1996). We further controlled for this possibility by testing decoder performance offline, while excluding planning period activity from within the first 100-150 ms post-cue; results stayed essentially the same (data not shown). That is, classification of the intended grasp was not affected by excluding potential long-latency, sustained visual activity.

Alternatively, delay period activity might have been more simply associated with anticipatory activity in proximal, and/or intrinsic hand muscles. Despite the fact that sampling of neural activity for decoding was stopped approximately 200 ms before initiation of reach to grasp, it is still possible that the monkey could have activated hand and arm muscles in anticipation of the forthcoming movement, while keeping its hand positioned on the hand rest. The decoded information might then have reflected an efferent command signal and/or afferent feedback. However, even though we did not directly monitor EMGs during decoding, previous work on patterns of EMG activity during reach-to-grasp demonstrated that the majority of hand and arm muscles required for grasping show selectivity for different patterns of grasp at the earliest during reaching-preshaping (Brochier et al 2004). While a minority of muscles *do* show early activation before initiation of the reaching movement (such as EDC or AD), this activity does not vary systematically as a function of grasp type (Brochier et al 2004) but is instead related to simple extension of the fingers at the moment of hand-rest release. In direct contrast to this, F5 neurons show clear tuning to grasp type well in advance of either M1 activity or EMG activation (Umilta et al 2007). Thus, preparatory activity in AIP and F5 which is tuned to grasp movement parameters cannot simply be explained by anticipatory muscle activation. Rather, it is consistent with the hypothesis that AIP and F5 are responsible for the sensorimotor transformation of visual information about object

properties into hand grasping instructions (Taira et al., 1990; Sakata et al., 1995, 1997; Murata et al., 2000, Raos et al 2006, Umilta et al 2007; Baumann et al 2009; Fluet et al 2010).

Discrete decoding

Another important feature of our study was the use of ensemble activity in F5 and AIP to decode grip type in a discrete as opposed to continuous sense. This approach is in the same vein as previous work which used neural ensembles to decode reach movement goals and/or saccade targets (Shenoy et al 2003; Santhanam et al 2006; Musallam et al 2004; Scherberger et al 2005). Discrete decoding is especially suited to applications where a neural prosthetics user selects quickly from multiple individual targets, e.g. during typing (Santhanam et al 2006). However, it is clear that full dexterous grasping behavior is a continuous process, and so an obvious structure to find the relevant control signals is M1 (Velliste et al 2008; Aggarwal et al 2009). Yet progress in this area has so far been limited, presumably because it is still unclear which M1 movement representation is optimal for effective hand/finger movement decoding (Paninski et al 2004; Townsend et al 2006; Morrow et al 2003; Moran and Schwartz 1999; Scott 2000) or how M1 activity is then translated into patterns of activation of relevant muscles (Lemon 2008; Yanai et al 2007; Asher et al 2008). A sensible alternative approach might therefore be to decode the discrete intended grasp posture, which could then be translated into the relevant continuous control signals for a robotic grasping prosthesis via an external control system. In this light, our study represents an important first step for the future development of more advanced grasping prostheses. This needs to be extended to include more grasp shapes and orientations (Scherberger 2009). Ultimately, to enable control of the patient's own hand, functional electrical stimulation based on neural activity will be necessary (Pohlmeyer et al 2009).

Comparison between AIP and F5

A number of invasive BMI studies in non-human primates have targeted multiple cortical areas using chronically implanted electrodes (Wessberg et al 2000; Carmena

et al 2003; Santhanam et al 2006; Doherty et al 2009). Multiple-area BMIs may allow implementation of different control strategies (Hatsopoulos et al 2004), improve decoder accuracy by stimulating somatosensory areas (Doherty et al 2009), or provide measurements of inter-area communication (Pesaran et al 2008; Andersen et al 2010). Our decision to implant both F5 and AIP was motivated by similar considerations. While both areas are related to hand grasping, important differences exist between AIP and F5 in terms of their anatomical and functional connectivity, and their representations of underlying grasp movement parameters. We found that these observed differences in grasp movement encoding manifest themselves during real time grasp decoding.

AIP receives input from parietal visual areas (in particular LIP, CIP, and V6a) and from the inferior temporal cortex (TEa, TEm) (Nakamura et al., 2001; Borra et al., 2008). These areas represent spatial and object orientation information of visible objects (Sakata et al., 1997; Tsutsui et al., 2001, 2002; Galletti et al., 2003). Consistent with this, AIP neurons strongly encoded orientation in the delayed grasping task, in terms of overall numbers of tuned cells and strength of tuning sensitivity (Fig. 3.11). This explains why decoding of orientation was significantly more accurate using AIP ensembles than F5 (figure 3.9 D,F). Our results are therefore compatible with AIP's representation of the grasp target in visual terms (Baumann et al 2009).

In contrast, F5 is strongly anatomically connected with M1, exerting a powerful influence over M1 corticospinal output (Shimazu et al 2004; Cerri et al) as well as containing its own projections to the spinal cord (Maier et al; for review see Lemon 2008). Concordant with these properties, F5 contained a strong representation of the grip type, especially precision grip (Fig. 3.5), F5 neurons were better at discriminating grip type than AIP (Fig. 3.11), and grip type decoding using F5 ensembles gave the highest accuracy (Fig 3.9 A,B). These results are consistent with the more direct role F5 plays in the generation of hand movement instructions (Brochier and Umilta 2007).

Although each area was specialized for one of the two task parameters, we found that the strong representation of grip type in F5 was counteracted by a particularly

weak representation of orientation in line with previous results (Fluet et al 2010), while in AIP the distribution of tuning properties was more balanced (Fig 3.9, B,D,F). Using F5 alone might therefore be a poor choice for a grasping prosthesis, as indicated by the observation that under-sampling of AIP activity in monkey Z led to extremely poor orientation decoding performance. However, using signals from AIP alone also consistently resulted in lower decoding performance than that obtained with both areas simultaneously (Fig. 3.9), suggesting that useful additional information about grip type was available to the decoder from F5. Taken together, these findings suggest that in terms of BMI control, neural activity from F5 complements and adds to AIP grasping related information, and an ideal grasping prosthesis would therefore utilize signals from both areas. This approach is consistent with the above observations on the respective roles of AIP and F5 within the parieto-frontal grasping network.

Multi unit signals and decoding

A key characteristic of our study was the predominance of multi unit neuronal signals recorded through the implanted FMAs. On average, around 85% of units recorded were subsequently identified as "multi-units" upon close inspection of spike sorting output. This was chiefly due to an inability of the sorting procedure to separate multiple spike waveforms and classify them reliably as belonging to individual neurons, especially when such waveforms have low signal/noise ratios (SNRs). Fixed FMA electrodes are particularly affected by this problem since they cannot be repositioned in order to improved isolation of single neurons. Multiunit activity from a given channel therefore comprised an aggregate spike train, pooled from a minimum of two or more neurons recorded through a single electrode (Super and Roelfsema 2006; Ventura 2008; Nikolic et al 2010). Care should be taken to distinguish this point-process signal from continuous "multiunit activity" (MUA) data generated from envelope functions applied to the low-pass filtered voltage trace (Super and Roelfsema 2006; Stark et al 2007; Choi et al 2010); although in any case, both sources reflect information that has been sampled from multiple instead of single neurons.

Of relevance here are results from several studies showing that the tuning properties of multiunit data are largely the same as those observed for single units (Desimone and Gross 1979; Zeitler and Fries 2006; Super and Roelfsema 2006, Super et al 2001; Hatsopoulos et al 1998; Maynard et al 1999; Stark et al 2007). Our data agree with this view. Grasp related activity exhibited by AIP and F5 multi-units, observed through PSTH firing rate curves (Fig. 3.3), analysis of tuning to grip type and orientation (Fig. 3.5) and measures of tuning sensitivity (Fig 3.11), were broadly similar to the equivalent analyses performed using single units (Baumann et al 2009, Fluet et al 2010). Furthermore, ensembles F5 and AIP data can be used to decode movement intentions in real time, when these signals are largely multiunit in nature, as previously predicted by offline simulations using only sequentially recorded, isolated single units (Pesaran et al 2002; Shenoy et al 2003; Scherberger et al 2005). Our study therefore confirms previous experimental and theoretical work demonstrating the use of multiunit activity as a viable source of information for decoding applications (Stark et al 2007; Ventura 2008; Fraser et al 2009).

A notable feature of our multiunit recordings was the relatively uniform distribution of preferred orientations amongst F5 multi-units during the planning period (with the exception of the -25deg handle position), which contrasted with the overall tendency for F5 single units to prefer extreme orientations (Fluet et al 2010). One possibility could be that we recorded mainly from cells belonging to the class of "sensory" F5 neurons described by Fluet et al, which when analyzed separately, showed an orientation distribution that is highly similar to ours (see Fluet et al 2010, their figure 3.8A). Arguing against this hypothesis however is the fact that these same multi-units showed a clear overrepresentation of precision grip, and were tuned throughout the planning period; this would then place them in the previous author's "motor cells" category (Fluet et al 2010).

A more plausible explanation therefore is that features of the orientation representation we measured in F5 were influenced in part by the nature of the recording process. It is known that although well-isolated cells may show clear tuning functions, during their implicit pooling into "multi-units" these may sum in such a way that the resulting net "electrode tuning function" fails to reach

significance (Ventura 2008; Fraser et al 2009). Thus one should expect a concomitantly weaker degree of orientation tuning for multiunit data in comparison to single unit results. Indeed this seemed to be the case for our F5 dataset: for example in monkey S, out of a total of 689 multi-units recorded, only 5% were significantly tuned to orientation. The end result was that we may have effectively underestimated orientation information in F5 due to averaging effect of MUA. This effect would be compounded by the immoveable nature of the FMA electrodes, which prevented precise targeting of dorsal and ventral F5; these are location where one would expect to find more orientation tuned cells (Fluet et al 2010). Our observation that task-modulated cells were distributed heterogeneously both within, and between F5 arrays (Fig. 3.1) adds weight to this hypothesis.

Given these considerations, it is therefore possible that the observed distribution of preferred orientations for F5 multi units was an outcome of the particular combination of cells that were pooled together in our multiunit recordings, i.e. sampling error. This could have more of an effect on F5 data, where the representation of orientation information is already weak in comparison to AIP, as quantified by the ROC analysis of significantly orientation-tuned cells (Fig. 3.11) and observed in previous studies (Baumann et al 2009; Fluet et al 2010). Under-sampling of F5 orientation encoding information would explain the particularly low performance of F5 ensembles during orientation decoding (Fig. 3.8, 3.9). However, despite these shortcomings, we maintain that our multiunit data demonstrates many fundamental similarities to the grasp-related properties of well isolated AIP and F5 neurons.

The use of multiunit data can be seen as a trade-off between ease of signal acquisition on the one hand, and loss of tuning fidelity on the other. Application of more advanced decoding paradigms and statistical encoding models may help to resolve the latter issue (Fraser et al 2009; Ventura 2008). With this in mind, multiunit data may prove to be a favorable source of control signals for neuroprosthetics applications. Extraction of single unit activity requires the use of computationally intensive spike sorting routines (Santhanam et al 2006; Quian-Quiroga et al 2004). However, this presents a significant challenge for the development of small,

implantable invasive BMIs using single unit activity, which would need to perform such routines online without access to additional computing resources (Ventura 2008). Furthermore, changes in signal quality due to tissue reactions (Poliakov et al 2005) and additional (as yet unknown) processes tend to severely single unit recording capabilities over long time periods of implantation (Schwartz et al 2006; Hatsopoulos and Donoghue 2009; Scherberger 2009). Multiunit data may also contain additional information which is not present in single unit activity (Zeitler and Fries, 2006; Stark et al, 2007). In light of this, recent research has focused on development of decoding approaches that utilize multiunit activity as recorded, with little or no prior sorting (Stark et al 2007; Fraser et al 2009; Borghi et al 2007; Ventura 2008). Our finding that decoding performance was only modestly improved by additional offline sorting (figure 3.10) is largely consistent with this approach. Overall, the present study demonstrates that multiunit F5 and AIP signals can be readily applied to decode grasp movement parameters in real time. Future work will be needed to extend this approach to include more varied and continuous dexterous grasping movements, executed without enforced delay periods (Scherberger 2009).

4 Temporal Decoding of Grasp Execution

In the previous chapter, the fundamental research question for this thesis is answered, namely the feasibility and analysis of implementation of a conceptual real-time grasping Brain-Machine-Interface via use of signals from AIP and F5. In this chapter new questions will be posed for possible extensions on the existing system.

In this chapter, mainly the findings on investigation of grasp time decoding will be presented. Via use of the same experimental setup and data collection methods we will investigate a different objective, namely to make classifications on "experiment-state" instead of behavioral output. In other words; decoding when to grasp instead of how to grasp is the fundamental question of this chapter.

4.1 Introduction

The vast body of literature about real-time prosthetic BMIs has concentrated on decoding spatial task components, i.e. reach type/direction/orientation, as we also did in the previous chapter. However, for a prosthetic device / a practical BMI it would be equally important to successfully decode temporal components, i.e. movement execution timing. It should be clear that in a real-life scenario a neuroprosthetic device, however strong in its spatial predictions, would be pretty

useless without also having a reliable temporal inference component. One should also remember that in the previous chapter we assumed that we have full knowledge of the experimental states. Providing such an external input regarding the experiment states is a standard treatment for many of the brain machine interfaces currently in development. In a controlled lab environment this is not a problem, but for a practical fully-autonomous prosthetic device estimating these epochs is also very crucial in order to control the timing of movement execution.

As we showed in the previous chapter we know that cortical neurons in AIP and F5 show strong modulations right before and during the execution of goal directed reaching movements. Therefore, we will make the simplifying assumption that there are 3 distinct behavioral states for hand grasping in our data; a baseline, a planning period and a movement period. We will try to decode these 3 states from the simultaneously recorded neuronal activity. It is outside the scope of this work to answer how many temporally differentiable states actually exists for grasping in the cortical regions of interest and to investigate the interaction and actual mechanisms among them. In this chapter, we mostly approach the problem from an engineering perspective, considering the practical implications of our findings on neural-prosthetic devices.

In the last decade, there were a few attempts to address the problem, decoding behavioral brain states for neural-prostheses. The pioneering work from Shenoy et. al. (2003) showed that utilizing a maximum likelihood estimator on spiking activity from parietal reach region (PRR) of the posterior parietal cortex for epoch classification is possible. Later work of Achtman et. al. (2007), proved the same concept in a more realistic application with a combination of reach decoding from spiking data from PRR & MIP. Their results were promising since they managed to show that a fully autonomous neural-prosthetic device for arm-reaching is working almost as well as a neural decoder with the perfect state timing information. Furthermore, in an attempt to capture the full probabilistic nature of the internal state transitions first Hudson and Burdick (2007) and then later Kemere et. al. (2008) employed hidden Markov models with unsupervised and supervised learning

schemes, respectively. The work of Kemere et al. is the latest publication to date on the subject to our knowledge.

Here, we will build our study starting from a common concept in these studies; using a sliding window to make continuous estimates about the underlying brain state. We will first start using the same decoder we have used in the previous chapter, simple maximum likelihood estimation as our principal classifier. Then we will gradually improve the robustness of our predictions by utilizing Markov models; a finite state machine (FSM) and a hidden Markov model (HMM). Finally, we will redefine the problem as a dichotomy where the experiment state should be classified either as Movement or Non-Movement type. This approach can be particularly useful for showing it is indeed possible to signal the movement start, which has an obvious application for real-life prosthetic devices. To achieve this we will follow a data-mining approach and present the results from different learning algorithms to investigate best performing classifier in such a scenario.

4.2 Methods

As in Chapter 3, neural data was collected via chronically implanted electrodes in the anterior intraparietal cortex (AIP) and ventral premotor cortex (F5) of macaque monkeys performing in the same experimental setup (delayed grasping task). As stated in Chapter 2, in delayed grasping task trials were divided into four epochs: fixation, cue, planning and movement. Monkeys initiated trials by placing both hands on rest sensors and fixating a red LED in the dark. After a variable delay (fixation, 700-1100 ms), the handle was illuminated for 600 ms (cue), revealing its orientation. At the same time, a second colored LED (‘context cue’) was illuminated, which instructed the animal about the required grip type (power or precision). After a variable delay (planning, 700-1100 ms), the dimming of the fixation light served as the go signal to initiate movement execution. It was therefore straight-forward to train our decoder in a supervised learning scheme with well-defined and separate epochs. For this study we concatenate Cue and Planning periods together and treat it as if this is a single state in an attempt to simplify the state structure. We believe that

such a 3-state approach is sufficient to capture a generic scenario for grasp related time course of internal states. We utilize the spiking activity in these epochs and assign them with behavioral states.

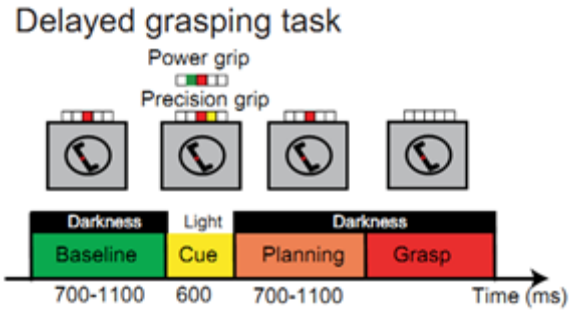

Figure 4-1: Experimental states of original 'delayed grasping task'
In this chapter for state decoding, we make the simplifying assumption that there are only 3 behavioural states exist; Baseline, planning and movement. The transition state "Cue" will be treated as a part of baseline while the signal change in this epoch will help our decoder to predict the next state.

The work from Shenoy et. al. (2003) showed that minimum window size for epoch estimations is between 150-300 ms. Later, Achtman et al. showed that a 200 ms window that started 150 ms after the target presentation was the shortest window that resulted in near asymptotic accuracy (Achtman et al., 2007). We have confirmed these finding with our experiments and here show only results where spike data sampled with a 250 ms moving window and the states were estimated every 50 ms. These values are chosen to wait enough for significant change in firing rate statistics to emerge while state switches still occur fast enough for practical constraints of a usable prosthetic device.

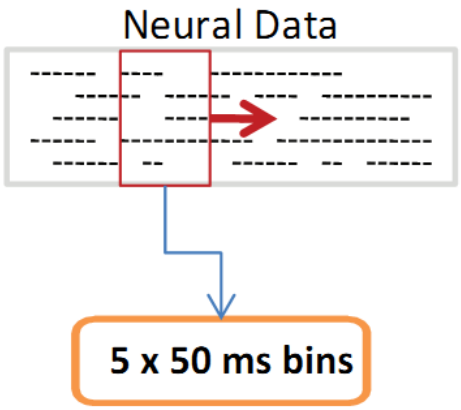

Figure 4-2: Illustration of sliding window data collection
A moving window with the width of 250 ms is updated at 20 Hz. The supervised learner is trained with epoch labelled data based on the actual experiment state.

For every moving 250ms time window a state label (based on the most observed state in this period) is created offline. A naïve Bayesian decoder is then first trained with some initial part of these data, and then used to make predictions on the second part of the data. As in Chapter 3, we make a Poisson assumption for firing rate distributions.

Furthermore, we have also built a state decoder widget into our real-time decoding software, Cerebus Decoder. Below in Fig. 4.4 one can see the graphical interface that implemented. This widget, implemented in C++ for Naïve Bayes decoders, FSMs and HMMs, conveniently decodes and visualizes the predicted state in real-time and can be used in combination with spatial decoding easily for fully autonomous hand grasping decoding.

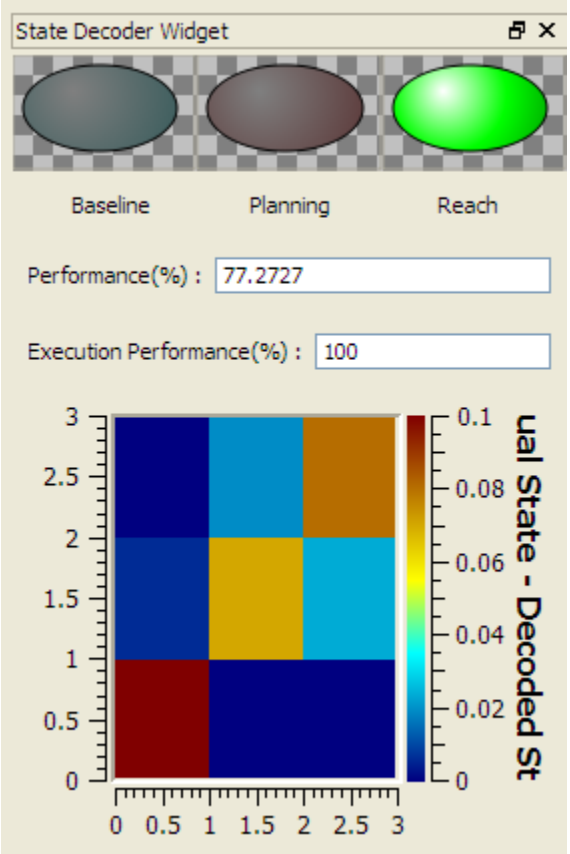

Figure 4-3: State Decoder widget of the real-time decoding software
The Naive Bayesian decoder implemented in our decoding software showed predicted states in real-time to the user via led-like visual elements. It also keeps track of session-wide-decoding-success for state prediction and provides the results both in numeric and confusion matrix forms.

4.3 Results

The data presented here is based on a total of 22 real-time decoding sessions conducted in two animals (monkey S, 11 sessions; monkey Z, 11 sessions) that were chronically implanted in AIP and F5 with floating microelectrode arrays (FMAs; monkey S, 128 channels; monkey Z, 80 channels). Across these sessions, we observed about 65 multi-units in average (SD 14) for animal Z and 105 units for animal S (SD 15). The mean real-time decoding performance with a maximum likelihood estimator was around 65% (SD 15%) for Animal Z and 82% (SD 16%) for Animal S.

4.3.1 Markovian State Machines

While this initial MLE performance is promising, the intrinsic nature of the temporal state estimation problem makes it a good candidate for combining it with finite state decoding techniques of Markovian characteristics. Toward this end, we have implemented a finite state machine of a particular Markovian order. After some initial analysis we decided to stick to an order of 5, i.e. system has a memory of 5 time bins of 250 ms, i.e. a total of 1250ms. The state machine simply assigns transition probabilities for each epoch for observed state histories combined with the MLE predictions of these time bins and then selects the most likely state as its prediction. The system essentially calculates a state transitions probabilities matrix at the end of the training period. Based on the observed historical order of states and MLE predictions, it calculates the most likely next state during testing. Our implementation may be classified as a more stochastic implementation of a constant consecutive observation of a decoding constraint as in Achtman (2007). The Markovian state machine should intrinsically capture such constraints, given that it is observed in training data. Implementation is done in C++ and runs directly inside our decoder either in real-time or in offline mode. Using the same dataset with MLE case, we achieved a state decoding performance of 69% (SD 15%) for animal Z and 88% (SD %15) for animal S.

4.3.2 Hidden Markov Models

As the next step, we introduced a Hidden Markov Model (HMM) decoder to utilize some unobservable states expecting to provide a better handle on the underlying stochastic processes, where the observations to HMM were again fed by the MLE.

HMMs are an extension to finite state machine models. The difference lies in the fact that states are not directly observable anymore but we can only observe a probabilistic function of them. They provide a probabilistic framework for modeling a time series of multivariate observations and were first described in a series of statistics papers by Leonard E. Baum and others in the second half of the 1960s. Starting in the mid-1970s they gained wide acceptance in speech recognition

community and they are still one of the most widely employed techniques in that domain. More recently, researchers used HMMs also in DNA sequence analysis, handwritten character recognition, natural language domains and many others. HMMs are powerful learners which can deal with non-stationary signals robustly. Desirable characteristics of a HMM which makes it a good candidate also for our task are: strong statistical foundation, the ability to handle new data robustly, and computational efficiency (due to the existence of established training algorithms).

Rabiner (1989) explains the basics of HMM and how it can be used for signal prediction. There are in general 3 basics problems to which HMMs are applied to;

- Given an observation sequence and model, efficiently compute the likelihood. This mode of operation might be useful for comparing different models.
- Given an observation sequence and model, find the optimal state sequence. This mode can be seen as a method to explain the data.
- Given observation sequence, estimate the model parameters that maximize likelihood of data. Once the parameters are estimated then we can use this model to make prediction on the unobserved data.

We followed the most standard form of HMMs in our implementations. We utilized the Baum-Welch algorithm to estimate our model parameters and used the Viterbi algorithm to make predictions. Interested reader may refer to Rabiner (1989) for further details on these methods. Implementation was done in C++ and again fully embedded in our decoder. We defined our model as having 3 observable states (our experiment states) and 3 hidden states and created a random initial state transition matrix. With this simple implementation we managed to get decoding success rates of %61 (SD %9) for Animal Z and %85 (SD %10) for Animal S. An important point to note that is, even though the overall average decoding rate was lower compared to State Machines, the robustness both in terms of classifier variance and the in-experiment-switching of states was better with HMMs. Especially the latter is quite an important contribution for real-life prosthetic devices. We want to point out also that the design of HMMs were really simple (3 serially connected states) and we have only used the MLE predictions as the inputs of our HMM. However, one can speculate that with a more carefully designed architecture where also the spiking

statistics of individual neurons are directly utilized in the HMM, one can achieve better performance. Since showing the robustness contribution was enough at this level we left the implementation of the above scenario for later research at the moment.

4.3.3 Data mining for binary decoding of movement time

Attacking to state decoding problem from a different angle, we wanted to quantify our pure *movement period* classification performance and investigate if we can find a better performing algorithm than Naïve Bayesian Learner for this period. This is the most interesting period since if we can estimate this period reliably, we can just use the data immediately before that period (from a buffer) to calculate spatial decoding and directly control an end-effector, in a prosthetic scenario. In order to find a decoder which provides highest-decoding performance we chose to follow a data-mining path and simply tested our dataset with many different learners. Since we needed efficient and correct implementations of all the algorithms we decided to utilize an existing library for this task. To this extent we utilized a Python (http://www.python.org) package called PyMVPA (Hanke et al. 2009). PyMVPA stands for MultiVariate Pattern Analysis in Python and offers an extensible framework with a high-level interface to a broad range of algorithms for classification, regression, feature selection, data import, and export. In Fig. 4.4 and in Table 4.1 the movement period decoding results with different model parameters and initial feature selection methods are provided for 4 different learners: Bayesian Logistic Regression, Naïve Bayes, k-Nearest Neighbor and Support Vector Machine. Fig. 4.4 shows two graphs for each learner; the top one shows a blue dot for every 50 ms window classified as movement epoch for all the 65 sessions decoded. All of these different sessions are aligned for the movement start time and the bottom graph shows the average movement prediction ratio for every window. This graph does not have direct physical implication but was generated to make an assessment on how fast and robustly a particular learner can capture the state changes. For feature selection, which is done on an independent training dataset, we followed simple *best*

first approaches in this section; we either selected the top 5% or top 50 neurons ranked on either 1-way ANOVA statistic or Linear-SVM weights (hyperplane coefficients). We have deliberately skipped a more complicated feature selection mechanism, like recursive feature elimination with more sophisticated learners, since our main objective was testing learners. The green dashed line in Fig. 4.4 is provided for a visual assessment of average goodness of fit of movement period prediction across trials. A brief inspection of this figure shows that all the learners are capable of predicting movement period successfully. Even the worst performing learner (Bayesian Logistic Regression classifier w/o feature selection) has an average success rate around 85% for this binary classification task, and the same value is close to 97% for the best learner (SVM with a linear kernel which uses again SVMs as initial feature selection). Our benchmark learner, Naïve Bayesian classifier, for this task performs well with an average 93.9% prediction accuracy. k-Nearest Neighbor classifiers improve this value slightly while providing a smaller false hit ratio. By comparing the blue and yellow dashed lines in bottom graphs we can see that they also introduce an increased lag to switch to movement classification. Finally, SVMs performed best among all combinations provided here. It is worth noting that we have not done an extensive optimization for parameter values of SVMs. In general, SVMs not only provided the best average accuracy but also showed the fastest response to the onset of the movement start. Final point to note that is the training and execution times of these learners. This is important for real-time neural-prosthetic applications since only after making this epoch classification we will be able to make a spatial decoding and execute it in the correct time. Linear-SVMs showed here again promising results being one of the fastest in testing time. Compared to Naïve Bayes classifiers we obtain a classification speed-up of around 50 times with the fastest SVM implementation (also the best performing learner).

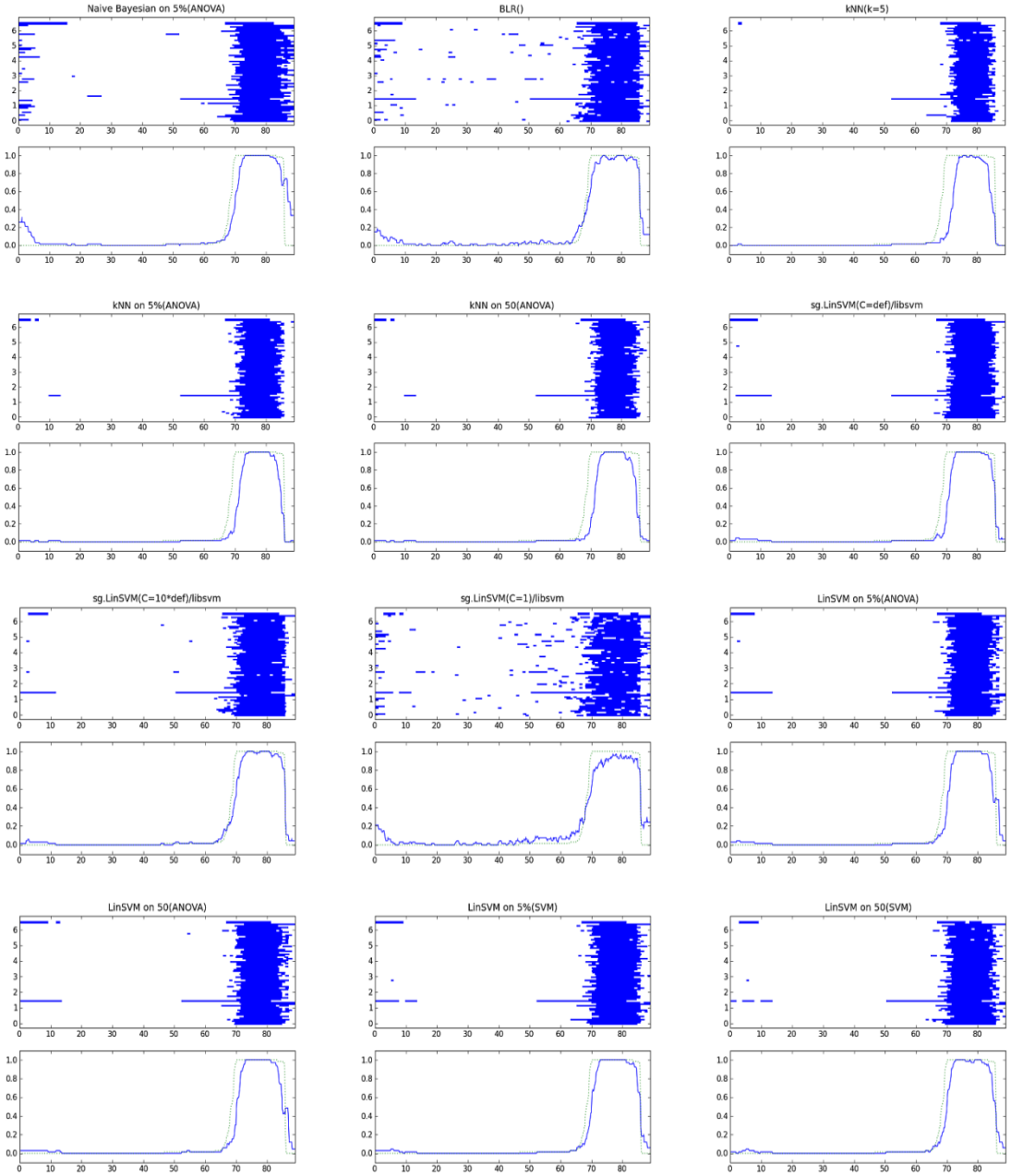

Figure 4-4: Movement Period classification with different learners from a sample experiment

Four different learners and 2 family of feature selection algorithms are combined with different parameter values selected into 12 example model shown above. For all the models 2 charts are provided, top chart shows the predicted movement times for all of the trials in the experiment (movement start times are aligned). The bottom chart shows an average of the top plot against a dotted optimal average decoding.

Table 4.1: Classification results for movement state decoding
Average cross-validated performance results and training and testing times for selected models on Animal-S data. Two different feature selection methods with different thresholds are tested; ANOVA and Support Vector Machines. As classifiers we have tested Bayesian Logistic Regression, Naive Bayes, k Nearest Neighbour and Support Vector Machine with a linear-kernel. For SVMs we have tried different implementations from different libraries and a couple of different parameters for C-value.

Classifier	% Perf.	Training Time	Testing Time	Total Time
Bayesian Logistic Regression	84.50%	34.73s	0.50s	72.29s
Naive Bayesian on 5%(ANOVA)	93.90%	1.49s	32.00s	68.44s
sg.LinSVM(C=1)/libsvm	94.30%	74.08s	12.91s	175.48s
k Nearest NeighbourN(k=5)	94.50%	0.00s	202.29s	406.19s
kNN on 50(ANOVA)	95.50%	1.29s	12.93s	29.94s
kNN on 5%(ANOVA)	95.70%	1.42s	23.62s	51.60s
LinSVM on 50(ANOVA)	95.90%	2.82s	0.81s	8.84s
LinSVM on 5%(ANOVA)	96.10%	2.97s	0.97s	9.35s
sg.LinSVM(C=def)/libsvm	96.60%	23.67s	17.20s	83.23s
LinSVM on 50(SVM)	96.60%	26.11s	0.69s	55.07s
sg.LinSVM(C=10*def)/libsvm	96.80%	31.63s	13.17s	91.08s
LinSVM on 5%(SVM)	96.90%	26.44s	0.86s	56.09s

4.4 Discussion for Temporal Decoding

Brain machine interfaces with prosthetic applications hold considerable promise to contribute to the quality of life of severely disabled patients. Recent developments on neuroscience and prosthetics engineering already proved the possibility of real-life usage scenarios of such approaches for humans (Hochberg et al., 2006). However, critical clinical constraints still need to be addressed. One of the

components for a practical fully-autonomous neuro-prosthetic device is the ability to differentiate among different cognitive states and act upon them accordingly. In this section, we have demonstrated our results with such an application in mind. Previous studies (Shenoy et al. 2003, Achtman et al. 2007, Kemere et al. 2008) have already showed that it is possible to decode those different states with significant success. However, they utilized different brain regions as their signal source; PRR/MIP, PMd/M1. In this work, we have showed that it is also possible to decode brain states for grasping from cortical areas AIP and F5. To do so, we have re-formulated our objective as decoding the experimental states and defined our framework as making predictions on a sliding window of neural spike rate data. The pure signal coming from DSP to our decoding machine was the same online-sorted spiking activity as in Chapter 3. We neglected the analog component of the brain signal as in the previous chapter and completely focused our analysis on the spike data with the assumption that it captures enough characteristics regarding the behavioral state. In an attempt to keep computation requirements manageable while still being reactive enough, we predicted the experiment state at 20 Hz while using the most recent 250ms.

We first used the Naïve Bayesian decoder, the classifier we have applied in the previous chapter for making spatial decoding, and managed to get promising decoding accuracies; 65% (SD 15%) for Animal Z and 82% (SD 16%) for Animal S. In order to utilize the cascaded structure of states, we introduced a Finite State Machine of a fixed Markovian order and achieved a decoding performance of 69% (SD 15%) for animal Z and 88% (SD %15) for animal S, with also improved robustness. Finally, in an attempt to capture the temporal stochastic properties of underlying states more efficiently we utilized a simple HMM decoder to obtain decoding accuracies of 61% (SD 9%) for Animal Z and 85% (SD 10%) for Animal S. The main contribution of HMM was increased robustness which can be enough for the trade-off of a slightly decreased state decoding accuracy in an actual implementation. Our reported average decoding accuracies were slightly less than the results in Achtman (2007), where they obtained decoding accuracies in the range 80%-90% for different methods they applied. Recording from a different brain region may be one reason for this but we believe that the main difference arises from

the finer modeling implemented in Achtman et al. (2007). They created different models for each reach direction in their approach which is a plausible extension. However, this was not feasible for us due to data limitations. Their experimental paradigm allowed them to record more trials per session and they have around 400 trials for training which was enough to estimate parameters for a finer model. Since our experimental setup allowed us to use only around 100 trials for training, we averaged the models for different spatial targets into a single model. And even after these averaging down the data resolution, we believe that the models' prediction accuracies are satisfying enough for most of the real-life scenario. Thus, we can conclude it is indeed possible to use data from AIP and F5 for cognitive state decoding purposes.

In the second part of this chapter we have questioned the optimality of our base decoder, Naïve Bayes, for the task. To do so, we redefined the problem as a dichotomy and tried to estimate the movement period vs. all others. We used different machine learning algorithms and showed that some significant increase in decoding accuracy and robustness is possible with stronger learners and feature selection mechanisms. The best results here were obtained with Support Vector Machines with linear kernels. The faster classification of SVMs also holds promise for real-time implications. However, one should note that computational power requirements and training times with SVMs may be not desirable in comparison to a Naïve Bayesian approach. Also, in an online adaptive learning setup Naïve Bayesian classifiers have advantages against classical SVMs. Finally, one should note that the data from the movement period that we have decoded in this part is coming from a normally behaving animal. The neuronal activity of a tetraplegic patient, on the other hand, may be substantially altered. Thus, we want to emphasize that the results of the last part of this section are obtained mainly for benchmark purposes.

Possible extensions

Results we have presented here and comparison to other works (Achtman et al. 2007) suggest that with finer modeling of different spatial cues one can observe better decoding accuracies for states. Due to data limitations we averaged the neural

responses across targets into a single model per state. However, for example, with enough stationarity in firing statistics among successive days one can try to follow a richer modeling approach. Or a better designed HMM with access to individual firing statistics of neurons may perform better while taking into account correlations and dynamics to exploit underlying temporal coding strategies used possibly by the brain.

One other point we have neglected was the fact that the actual context change signals arriving and being processed in the brain also require some time. The approximate time necessary for visual information to be transduced at the retina, processed by several subcortical regions, and finally reach cortical areas we record from is believed to be in the order of 50 ms (Santhanam et al 2006). Thus, it is not realistic to expect our decoder to switch immediately in the first decoding window to the new state; most likely even the brain is not fully aware of that signal at that moment. This is creating a small negative bias in the performance of our decoders.

Another approach to increase the stability and the performance of our decoder may be to introduce different signal modalities. One very good candidate is local field potentials (LFP). While capturing the lower frequency components of the neural signals, LFP power is higher during the planning period (Shenoy et al. 2003, Scherberger et al. 2005), which might be very useful for our purposes.

In this section, we showed that using Naïve Bayesian decoders, it is possible to reliably predict the current cognitive state. Using Markovian state machines and HMMs and also some other learning algorithms, we have demonstrated possible improvement in prediction accuracies and robustness of state switching. The latency of our algorithm (~300 ms) is in the same order with the actual reaction times of primates (200 – 400 ms), thus we can conclude that this implementation can be realistically useful in a fully-automated hand grasping BMI. We are not claiming optimality for our decoder or for the parameter sets selected. Furthermore, for a clinical implementation one should take into account on the above stated possible extensions to the system.

5 In Search of a More Robust Decoding Algorithm for Hand Grasping

In this chapter we are returning back to our original objective of decoding the spatial component of the task, i.e. predicting the hand grasping posture of the animal given the neural data in the planning period. But this time, similar to what we have done in chapter 4, we are following a data mining perspective to find a better performing decoder. To achieve this we will compare different standard machine learning approaches on the same data set in an offline setting. Similar to the previous chapter, we used here only spike rates that were computed from online-sorted spike counts. The contents of this chapter is published as a peer reviewed conference paper at proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society -EMBC'10 (Subasi et al. 2010).

5.1 Introduction

If we want to decode neural information optimally, a good point to start with is to ask: “How does the brain actually decode this information?” Population coding is one theory which captures the stochastic and distributed characteristics of neurons

well and explains some of the intrinsic properties of motor cortical areas. Population coding has some built-in noise compensation characteristics and is robust. It also gives rise to short-term memory in the system and can instantiate complex and non-linear functions (Pouget et. al., 2000). An early-developed population coding mechanism is "population vector analysis" which is essentially a cosine function fit to the observed activity direction (Georgopoulos et. al., 1989). The population vector algorithm enjoyed substantial success until maximum likelihood estimation (MLE) approaches showed to be superior in capturing the underlying probability distributions (Oram et al. 1998). Being a parametric method, MLE usually assumes Poisson statistics for motor neuron firing rates and for the parameter estimation to be tractable, we made the strong assumption that the individual neurons are firing independently. So, we reach Naïve Bayesian Classifiers, which are widely used in BMI community for discrete goal directed predictions. This was also the main method we followed in chapter 3 for online decoding.

Naïve Bayesian Classifiers can be trained efficiently in a supervised setting and despite their non-realistic independence assumptions they perform very well in cortical signal decoding and are treated to date as state of the art in many settings. But frequently, analysis of multiple simultaneously recorded spike trains with these naive assumptions will raise the legitimate question whether the data is treated adequately. The absence of well-developed statistical methods for analyzing multiple point processes is the main concern for practitioners with classical statistics background (Brown et. al., 2004). Here we address this issue from a data driven perspective, where we define the decoding goal as having the best prediction accuracy, without spending much effort on optimal modeling of the data generating process.

We can define two broad categories of approaches towards generating predictions in general (Breiman, 2001). The first, *"Data modeling approach"* utilizes an underlying model that is constructed a-priori to generate data. This approach relies heavily on parameter estimation techniques and model validation that is achieved usually by goodness-of-fit tests. However, one should be careful in the choice of estimators while keeping in mind that even significant results may be misleading if

the assumptions regarding the initial model are not appropriate. The family of learners in this category commonly referred as likelihood-based classifiers in the machine learning community.

In contrast, the "Algorithmic modeling approach" requires no a-priori model for the underlying data generation. Instead, black-box learners work on all available data and validation is checked by predictive accuracy. One should be particularly careful about over-fitting when working with this set of algorithms. Being strong learners, this family usually provides better prediction performance as compared to model-based procedures, but they suffer from lack of explanatory power. The classifiers in this category are commonly referred as discriminant-based learners.

The main motivation of this section was to bring some of the well known methods from data modeling and algorithmic modeling together, first for the purposes of comparison with our implemented setup and to convince ourselves that we are getting most out of our data. Second, to investigate a possible combined approach which may introduce better performance in the setup and ultimately may shed light on the underlying characteristics of the selected cortical regions.

5.2 Methods

To compare different algorithms on our data set, we created a testing-platform, using the open source, JAVA based software RapidMiner (Mierswa et. al. 2006). This program provides efficient and thoroughly tested implementation of many standard machine learning algorithms in a standard framework and makes the experiment design and the task of comparing learners more robust and easy. We have mostly utilized the algorithms and standard machine learning procedures out of the box coming with the software. Whereas we have extended it in a few occasions (Poisson Naïve Bayes and Anova feature selections to name two examples) where desired algorithms were missing or implemented in a different way compared to how we wanted to test them. The benefits of using open-source software were strongly evident in such occasions. In the end, the ability to design and compare learning experiments either through graphical user interface or XML description or directly

using in JAVA environment through RapidMiner has proven to be a significant time saver in our research. In Figure 5.1, one can see the typical design flow of an experiment in RapidMiner in a tree structure. Also in Appendix A, a sample xml description of one study used in this work is included. This open and clean architecture of RapidMiner made our objective of searching for an improved decoder for our data-set easier.

Trying to find "the" optimal learner is not an easy task (if possible at all), first due to our lack of knowledge and a formal grasp of the underlying data generating process (i.e. primate premotor and parietal cortex) and second, due to learning in general being an ill posed problem. It is possible and very likely that with new evidence (new recordings in our case or recordings from different animals or slightly different regions of the brain) we will realize that our current "optimal" decoder is actually sub-optimal. Thus, we want to make clear that if we use the word optimal in this section we are not claiming a universal optimality and mainly referring "the optimal" among the learners that we have tested for our data set. We can say in general that (Dietterich, 2003), in all learning algorithms that are trained from example data there is a trade-off between 3 factors:

- The complexity of the learner, or the capacity of the hypothesis class.
- The amount of training data.
- The generalization error on new examples.

As the amount of training data increases, the generalization error decreases. As the complexity of the learner increases, the generalization error first decreases but start to increase after a critical point. A common approach to measure the generalization ability of a hypothesis (a learner) is to use data outside of the training set. We can simulate this by dividing training data to two disjunct parts. We use one part for training and the remaining part is used to test for the generalization ability (validation). Then, assuming large enough training and validation sets we can select the most accurate learner on the validation set. This process is commonly referred as cross-validation and used as the main model comparison tool also in the previous chapters.

One other common theme in machine learning is the bias/variance dilemma (Geman et al. 1992). This can be very briefly summarized as the opposite behavior of bias and variance characteristics of a learner in general. For a model, with increasing complexity small changes in the dataset cause greater changes in the outcome, but a complex model on average allows a better fit of the underlying function; thus bias decreases. To decrease bias, the model should be flexible, at the risk of increasing variance. If the variance is too high, the performance on previously unobserved data may suffer significantly. The optimal model should find the best trade-off in between. In this work, we tried deliberately not to over optimize the learners (it is still possible to over-fit to data by trying too many learners even in a cross-validation setup) by extensively searching the parameter spaces. As a rule of thumb, we always tried to stay in a reasonable parameter space and run very simple optimization routines if necessary. Another important point to note that is the initial feature selection mechanism utilized for the learners. Since we were mainly interested in comparing different learners, we did not spend too much time on feature selection (or on extraction) algorithms and utilized the same method (one way ANOVA with a fixed confidence threshold) from Chapter 3 for all the learners here. We have run some simple analysis on the effect of different feature selection methods (not shown here) and have not found any significant results. However, a more detailed analysis may prove otherwise.

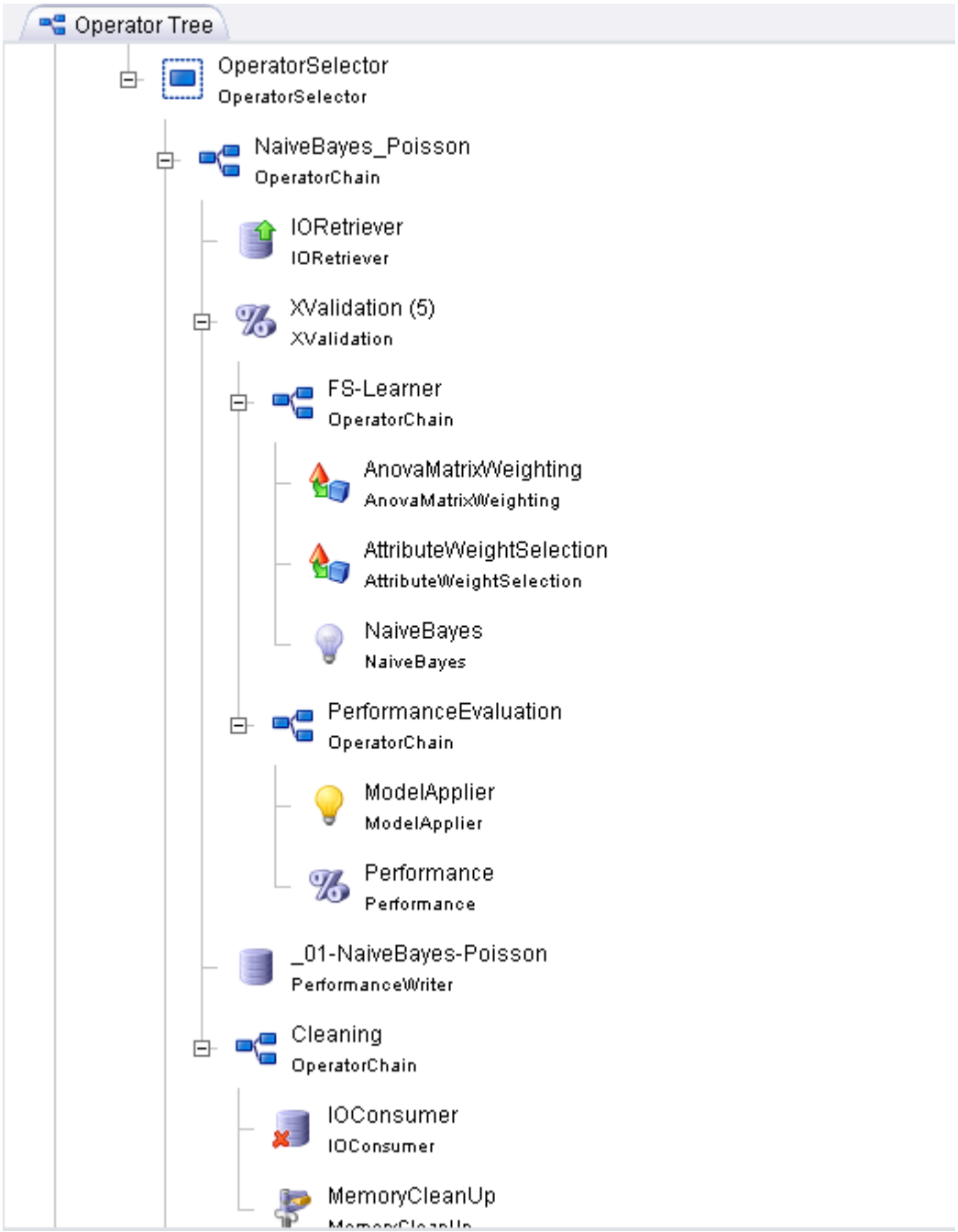

Figure 5-1: A graphical description of experiment design in RapidMiner
The steps of a learning experiment are provided in tree-like structure. This structure is reflected in a DOM tree via XML in RapidMiner and is therefore easy to edit and share experiments outside of the GUI environment. The execution flow of the experiment logic can be read from up to down where sub-trees have higher priority. Above example shows the flow of a cross-validation experiment with ANOVA feature selection for Naive Bayes learners. The details the implementation of the cross validation is hidden from user in this view but still accessible through JAVA source code when necessary.

The reason why we still wanted to use a simple feature selection layer is the fact that there is a trade-off between feature selection/extraction and decision making. If the feature extractor is good, the task of the learner may become trivial. On the other hand, if the learner is good enough then there should be no need for feature extraction, it should combine and select useful features internally. Some of the learners we have tried do this in fact. Thus, if we had provided data without prior feature selection, these classifiers would have an unfair advantage against others. By providing the equal input set, after a simple feature selection layer, to all classifiers, we wanted to bring them into a comparable common ground.

In particular, we tested: Naïve Bayes classifiers with Poisson and Gaussian data assumptions, Perceptrons, Decision Trees, Logistic Regression Classifiers, k-Nearest Neighbor classifiers, Naïve Bayes trees, Multi Layer Perceptrons (classic back-propagation neural networks), and Support Vector Machines (SVM) with Linear and Radial Basis Function (Rbf) kernels. We also tested some widely used ensemble method approaches; Bagging, Adaboost, MultiBoosting, Bayesian Boosting and Stacking. Referring to our discussion on two cultures for statistical modeling we can say that the algorithms we have tried (besides Naïve Bayesian classifiers) were closer to the algorithmic modeling family. Naturally, we have selected Naïve Bayesian classifier with a Poisson distribution assumption as our benchmark, not only because of its frequent use in the literature but also because it served as our principle learner in the previous chapter.

In the following, we provide a brief description of the characteristics of the learners we found particularly interesting for this work.

5.3 Description of Selected Learning Methods

Since our main objective here is not to provide a didactic text on machine learning algorithms, we will concentrate our attention on the methods we have found interesting results with. Naïve Bayesian classifiers, Decision Trees and Support Vector Machines are the main learners we found particularly interesting, due to reasons which will be clear in the Results section. We will also provide some

information about ensemble methods given the significant attention they have drawn in the last decades and possible implications in our research. Since we have already provided a formal discussion of Naïve Bayes classifiers in chapter 2 we skip them here. The interested reader in a more detailed treatment of the algorithms provided should refer to classical text books. Some particularly good examples are; The Nature of Statistical Learning Theory (Vapnik, 1995), The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Hastie et al. 2009), and Introduction to Machine Learning (Alpaydin, 2010).

5.3.1 Decision Trees:

A decision tree is a nonparametric, supervised learner which is composed of a hierarchical decision node structure. Each of these nodes evaluates a function against a threshold to create its branches. This process starts at the root node and is repeated recursively until a terminal leaf node is hit. At leaf nodes, the value written constitutes the output. No assumption is made a priori regarding the class probability distributions and the tree structure. Therefore, during learning the tree grows and branches depending on the complexity of the problem inherent in the data. Each node defines a discriminant in the input space dividing it into smaller regions that are further subdivided as a path from the root down taken. This way, a complex function is broken down into a series of simple decisions. Each leaf node defines a localized region in the input space where instances falling in this region have the same labels in classification. (Alpaydin, 2010)

The two main advantages of a decision tree are; i) the fast localization of the region covering an input due to hierarchical placement of decisions and ii) it's straightforward interpretability by humans. Because of these desirable properties decision trees are sometimes preferred over more accurate but less interpretable methods. Its intrinsic feature selection mechanism is also a desirable property for many tasks.

In the case of a decision tree for classification, the goodness of every split is quantified by an impurity measure. Common methods to do this quantification are

using entropy from information theory, the gini index (Breiman et. al. 1984), or error rates. So, for all attributes the impurity is calculated and the one that has the minimum entropy (or other measure) is chosen. Tree construction continues from that point recursively for all the branches that are not pure. CART (Breiman et. al. 1984), ID3 (Quinlan 1986) and C4.5 (Quinlan 1993) are the most popular implementations of this procedure. A common final step used in decision trees is called pruning. This is simply a way of controlling the variance of the learner in order to achieve higher generalization ability with either pre- or post-processing. We have utilized C4.5 learners with a pre-pruning setup in this work. Decision trees by themselves did not produce superior results compared to other classifiers, however in an ensemble setting in combination with our classifiers we have observed desirable outcomes (see Results).

5.3.2 Support Vector Machines (SVM)

SVM is a very successful discriminant-based method and arises from Vapnik's principle to never solve a more complex problem as a first step before the actual problem (Vapnik 1995). i.e., if we want to simply learn discrimination among given classes, it is not necessary to estimate the class densities $p(\boldsymbol{x}|C_i)$ or the exact posterior probability values $P(C_i|\boldsymbol{x})$; we only need to estimate where the class boundaries lie, that is, **x** where $P(C_i|\boldsymbol{x}) = P(C_j|\boldsymbol{x})$.

For a two class problem with labels −1/ + 1. For the sample $X = \{\boldsymbol{x}^t, r^t\}$ where $r^t = +1$ if $\boldsymbol{x}^t \in C_1$ and $r^t = -1$ if $\boldsymbol{x}^t \in C_2$ we would like to find $\boldsymbol{w}$ and w_0 s.t.,

$$r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \geq +1$$

And we want to maximize margin for better generalization. The task can therefore be defined (see Cortes and Vapnik 1995; Vapnik 1995) as to

$$min\frac{1}{2}\|\boldsymbol{w}\|^2 \; subject\; to\; r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \geq +1, \forall t$$

If we re-formulize the above equation as an unconstrained problem with Lagrange multipliers α^t as;

$$L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{t=1}^{N} \alpha^t r^t (\boldsymbol{w}^T \boldsymbol{x}^t + w_0) + \sum_t \alpha^t$$

This should be minimized with respect to $\boldsymbol{w}$, w_0 and maximized with respect to $\alpha^t \geq 0$. The saddle point gives the solution. This is a convex quadratic optimization problem because the main term is convex and the linear constraints are also convex. Once we solve for α^t, we see that there are N of them, most vanish with $\alpha^t = 0$ and only a small percentage have $\alpha^t > 0$. The set of $\boldsymbol{x}^t$ whose $\alpha^t > 0$ are the support vectors. The instances that are not support vectors carry no information; even if any subset is removed, we would still get the same solution. However, if the data is not linearly separable, the algorithm above will not work. Thus Soft Margin Hyperplane concept is introduced, which essentially introduces some slack variables to the optimization problem and a modifying penalty term as a regularization scheme to limit complexity. After this modification, SVMs can work on arbitrary data, however it is sometimes not the best approach to attack non- linearly separable data this way. Therefore another transformation, commonly referred as Kernel trick, is introduced which maps the input data to another (most of the time higher order) space where it is linearly separable. A nice thing about Kernel trick is due to the way we define our optimization problem, we don't need to explicitly map all the input data set on the new space, instead just calculate a dot product in this space, which makes the computation significantly easier. Or in other words, the use of kernel functions implies a different data representation; we no longer define an instance (object/event) as a vector of attributes by itself, but in terms of how it is similar to other instances.

After this treatment SVMs become a really powerful learner which can deal with non-linear, high-dimensioned data. Among the most popular Kernels are linear, polynomial and radial basis function kernels. These are also the ones we have used in our analysis. The novelty of support vector machines lies in integration of the mapping data to a new space through nonlinear basis functions into a learning scheme whose parameters are defined in terms of a subset of data instances (dual representation). This way, without a need to explicitly evaluate the basis functions one can limit complexity by the size of the training set. Because there is a unique

solution to the optimization problem, we do not need any iterative optimization procedure as we do in neural networks. Support vector machines are currently considered to be one of the best off-the-shelf learners and are widely used in many different domains, especially bioinformatics (Schölkopf, Tsuda, and Vert 2004) and natural language processing applications (Joachims 2002). Also in our setup SVMs turned out to be one of most effective learners (see Results).

5.3.3 Ensemble Methods

Over the last decade, ensemble based systems have enjoyed a growing attention and popularity due to their many desired properties, and the broad spectrum of applications that can benefit from them. The "No Free Lunch" theorem states that there is no single learning algorithm that always induces the most accurate learner in every domain. The typical approach is to train many learners and then choose the one with the best validation set performance. Each learning method contains some intrinsic set of assumptions and this bias leads to error if the assumptions do not hold for the data. Learning is an ill-posed problem in general and with finite amount of data, each algorithm fails under different circumstances. To aim at the highest possible accuracy, the performance of a learner may be fine-tuned on a validation set but still there is no guarantee that we will reach the desired accuracy. The fundamental idea in ensemble learning is there may be another learner that is accurate on the cases where our learner at hand fails. By combining multiple learners in a smart way we can achieve better performance. Recently with computation and memory getting cheaper, such systems composed of multiple learners have become very popular (Kuncheva 2004).

There are two fundamental questions we need to answer in order to work with ensemble systems:

1. How do we generate base-learners that complement each other?
2. How do we combine the output of base-learners for maximum accuracy?

We will give a brief look to possible approaches to attack those problems below.

Finding Complementary Learners: One of the most common approaches for generating complementary base learners is to feed them with different training sets. This can be done randomly by drawing random training sets from the given sample; this is called bagging. Bagging, short for bootstrap aggregating, uses bootstrap to generate L training sets, trains L base-learners using an unstable learning procedure, and then, during testing, takes an average (Breiman 1996). In bagging, generating complementary base-learners is left to chance and to the instability of the learning method. Another common approach to train complementary learners is based on serially training so that instances on which the preceding base-learners are not accurate are given more emphasis in training later base-learners. Boosting and cascading are the most well known approaches in this family. In boosting, we actively try to generate complementary base-learners by training the next learner on the mistakes of the previous learners. Schapire (1990) has shown that this overall system has reduced error rate, and the error rate can be reduced arbitrarily by using such systems recursively. Though quite successful, the disadvantage of the original boosting method is that it requires a very large training sample. Freund and Schapire (1996) proposed a variant, named AdaBoost, short for adaptive boosting, that uses the same training set over and over and thus need not be large, but the classifiers should be simple so that they do not overfit. Once training is done, boosting takes the weighted vote where weights are proportional to the base-learners' accuracies. In this seminal work, Freund und and Schapire (1996) showed improved accuracy in twenty-two benchmark problems, equal accuracy in one problem, and worse accuracy in four problems. Schapire et al. (1998) explain that the success of AdaBoost is due to its property of increasing the margin. This suggests AdaBoost and SVM have similar approaches for the discrimination problem.

Combining Predictions from many Learners: The earliest and most intuitive approach for multi-learner output combination is voting, which is nothing but simple old democracy for base classifiers. In other words, every classifier has one vote and majority decides in the end. In recent decades, more complicated algorithms than

simple voting are suggested and below we provide some of the most popular approaches: Stacked generalization, Bayesian boosting and Multi-boosting.

Stacked generalization: (Wolpert 1992) In Stacking, base-learners are simply combined through another learner, which is again trained with the sample training data set. This is similar in concept to cross-validation, but instead of using a winner take all like solution as in classical cross-validation, stacking blends the results. The combiner learns what the correct output is when the base-learners give a certain output combination. Stacking is a means of estimating and correcting for the biases of the base-learners. In stacked generalization, the base-learners preferred to be as different as possible so that they will complement each other, and selection of different learning algorithms as base learners is therefore suggested. Stacking usually delivers superior results compared to its base classifiers. It is also worth noting that it was extensively used by the two top performers in one of the most challenging and popular machine learning competition of the last years (Netflix competition, Sill et al. 2009). Also in our research, we found a promising implementation of stacking, which delivered superior results compared to the individual learners.

Bayesian boosting: (Scholz et. al., 2005) At each iteration, base models are induced and reweighted continuously, considering the latest batch of examples, only. Unlike other ensemble methods, the proposed strategy adapts very quickly to different kinds of concept drifts. The algorithm has low computational costs. It has empirically been shown to be competitive to, and often to even outperform more sophisticated adaptive window and batch selection strategies. This approach might be especially interesting for non-stationary time series learning.

Multi boosting: (Webb, 2000) MultiBoosting is an extension to the highly successful AdaBoost technique where it is combined with wagging. Wagging (Bauer & Kohavi, 1999) is a variant of bagging, which requires a base learning algorithm that can utilize training cases with differing weights. Rather than using random bootstrap samples to form the successive training sets, wagging assigns random weights to the cases in each training set. It is able to harness both AdaBoost's high bias and variance reduction with wagging's superior variance reduction. Using C4.5 as the base learning algorithm, Multi-boosting has demonstrated to produce decision committees

with lower error than either AdaBoost or wagging on a large representative cross-section of UCI data sets (Frank 2010 - http://archive.ics.uci.edu/ml). It offers the further advantage over AdaBoost of suiting parallel execution.

Combining multiple learners has been a popular topic in machine learning since the early 1990s, and research has been going with an increasing pace since then. The "no-free-lunch" theorem states that there is in fact no best classifier for all classification problems and that the best algorithm depends on the structure of the available data and context. Still, many studies have compared various ensemble generation and combination rules under various scenarios. A non-exhaustive list includes; Dietterich (2003), Breiman (2001), Bauer et al. (1999) and Quinlan (1996). The typical consensus is that boosting usually achieves better generalization performances, but it is also more sensitive to noise and outliers (Polikar, 2006). In this line, AdaBoosted decision trees are considered to be one of the best out-of-the-shelf machine learning algorithms. However, with additional work for a carefully designed setup, one can achieve superior results with methods like stacking.

In the following section, we will present our results with all of these different approaches and finally suggest an ensemble well suited for the problem at hand.

5.4 Results

We present decoding results with 24 different learning methods from 8 recording sessions from each animal (Table 4.1, Table 4.2). There is a first set of 11 algorithms, which is selected from a wide variety of widely used machine learning methods; Naïve Bayes with Poisson (benchmark) and Gaussian distributions, Bayesian Logistic Regression, Bayesian Network, k-NN, Decision Tree, Naïve Bayes Tree, Simple Perceptron, Multi-Layer Perceptron, and finally Support Vector Machines (SVM) with linear and Rbf kernels. Another set of 11 algorithms are then constructed using ensemble methods which utilize some learners from the previous set as their base learners. Finally, after obtaining results with all of the above mentioned methods, we designed two new ensemble systems which are essentially

stacking systems. Based on the results from initial run, we decided to provide 2 Naïve Bayes classifiers (with Poisson and Gaussian distributions) and an SVM-Rbf as our base learners to it and used either a decision-tree or a neural network on the top-layer as the final decision maker. This way we wanted to bring the best of both worlds from model- and discriminant-based approaches together.

Looking at the average decoding accuracies (last columns of Table 5.1 and Table 5.2), the first thing to notice is that there is a substantial difference for both animals (~0.60 vs. ~0.33 for best performing classifiers). This is not surprising after we observed similar results also in Chapter 3. In Fig.5.2 we provide the number of tuned units for both animals in each experiment and one can see that Animal S has significantly more neurons which show task-specifity.

Among single learners, Naïve Bayes (NB) and SVM-Rbf showed best average decoding accuracy (~58%) for both Animal-S and Animal-Z (~32%). On individual recordings of Animal-S, NB-Poisson classifiers ranked 5 times in top 3 classifiers among all the classifiers for that particular recording; the same value was 4 for NB-Gaussian and 3 for SVM-Rbf. Also SVM-Linear and Multilayer perceptrons were ranked among top-3, 2 times each. None of the other single classifiers ranked in top-3 for any of these 8 recordings. The results for Animal-Z was somehow similar, again NB and SVM performing best among single classifiers. But this time SVMs manage to be ranked in top-3 5 times, whereas it was 2 and 3 for NB-Poisson and NB-Gaussian classifiers. Single and Multi-layer perceptrons are ranked in top-3 1 and 3 times, respectively.

Among boosting based ensemble methods, BayesianBoosting with NB-Poisson base learners had the best average accuracy and this learner ranked 4 times in top-3, for Animal-S. Adaboost with NB-Poisson and NB-Gaussian base learners showed second best results both in terms of average performance and top-ranking counts. For Animal-Z, MultiBoosting and Adaboost showed best results with both NB-Poisson base learners. Also the performance of MultiBoosted Decision Trees was close to them. That was the only case in which decision trees as base learners were close to NB classifiers. In general, boosting based ensemble methods improved accuracy in

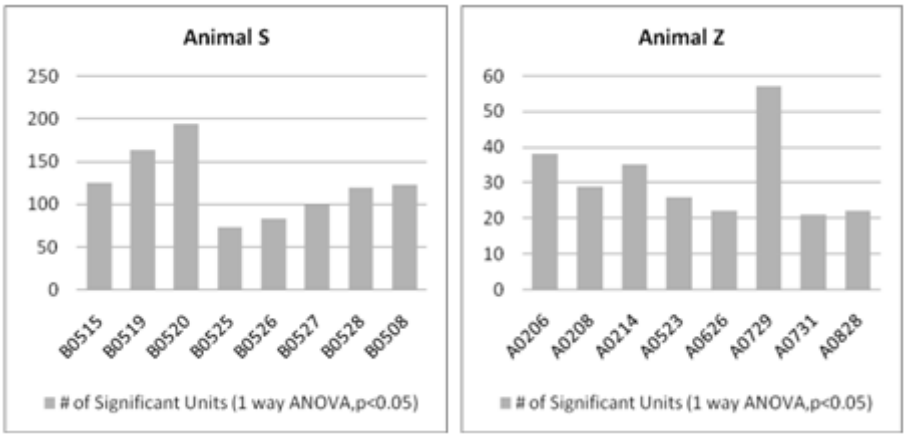

Figure 5-2: Number of significantly tuned units in 8 sessions for each animal
Recordings from Animal-S show clearly more tuned units for every experiment. With better signal quality we are obtaining also better decoding performance for this animal. (Table 4.1-4.2)

some cases for Animal-S, but were beaten in average performance in the end. For Animal-Z, on the other hand, we observed a better performance with a MultiBoosting ensemble compared to single learners' winner SVM-Rbf. Here we can see that, whenever the signal-to-noise ratio is high and the model assumptions are violated heavily, ensemble approaches may outperform state of art single learners.

Finally, once we include our stacked generalization based learners into comparison, we see that they are indeed providing the best average accuracy for both animals (~60% for Animal S, ~33% for Animal Z) and also the best top-3 ranking counts. For Animal-S our proposed method ranked in top-3 in 7 out of the 8 recordings. It was a bit more modest (4 out of 8) for Animal-Z but still best compared to all other single and ensemble learners. Furthermore, it also had the minimum decoding success variance across daily sessions among the best performing approaches, i.e. it is a more robust learner (Table 5.3). We tried using either a decision tree or a multi-layer perceptron neural-network as the top-level learner for this stacking approach. In the end, the decision tree version showed slightly better performance and due to other advantages of decision trees against

neural nets, like easier training and human-interpretability of analysis, we pick the stacking of NB-Poisson, NB-Gaussian, SVM-Rbf with a top level Decision-Tree learner as the best classifier in this setup. The observation yielding to this learner was the different performance characteristics of model-based vs. discriminant-based learners on different days. We hypothesize that, due to increased model-assumption, and model-assumption-breaking characteristics of the collected signal at some days (increased correlation for example) a discriminant-based learner like SVM may outperform NB occasionally. Therefore, a top level learner that can track the different signal characteristics and chose the learner which will provide the best results accordingly will yield a better performance.

Our analysis provided in (Tables 5.1, 5.2, 5.3) confirmed this intuition.

Table 5.1: Decoding performance for Animal S

The best performing 3 learners' results are colored (darkest for the best one) for every experiment.

Record IDs :	A0515	A0519	A0520	A0525	A0526	A0527	A0528	A0508	AVERAGE
Generic Algorithms									
NaiveBayes-Poisson	0.391	0.581	0.514	0.500	0.654	0.582	0.758	0.632	**0.577**
NaiveBayes-Gaussian	0.449	0.602	0.543	0.500	0.605	0.532	0.697	0.658	**0.573**
BayesianLogisticRegression	0.406	0.581	0.476	0.386	0.444	0.468	0.689	0.526	**0.497**
BayesNet-K2	0.449	0.398	0.390	0.341	0.370	0.481	0.409	0.500	**0.417**
DecisionTree	0.304	0.366	0.343	0.295	0.259	0.354	0.220	0.303	**0.306**
NaiveBayesTree	0.246	0.269	0.286	0.386	0.185	0.278	0.303	0.303	**0.282**
kNN	0.435	0.462	0.457	0.386	0.420	0.481	0.538	0.539	**0.465**
Perceptron	0.464	0.548	0.410	0.318	0.346	0.494	0.636	0.368	**0.448**
MultiLayerPerceptron	0.507	0.484	0.524	0.364	0.580	0.443	0.720	0.658	**0.535**
LinearSVM	0.464	0.548	0.667	0.341	0.630	0.506	0.705	0.632	**0.562**
RbfSVM	0.536	0.570	0.648	0.432	0.617	0.468	0.697	0.645	**0.577**
Ensemble Methods									
Adaboost-NaiveBayesPoisson	0.362	0.538	0.543	0.500	0.667	0.544	0.720	0.563	**0.555**
Adaboost-NaiveBayesGaussian	0.406	0.602	0.552	0.500	0.617	0.544	0.697	0.589	**0.563**
Adaboost-DecisionTree	0.304	0.344	0.343	0.273	0.235	0.367	0.205	0.321	**0.299**
Adaboost-kNN	0.391	0.462	0.390	0.318	0.395	0.380	0.530	0.457	**0.416**
Adaboost-Perceptron	0.333	0.505	0.333	0.318	0.370	0.380	0.606	0.458	**0.413**
Adaboost-LinearSVM	0.478	0.581	0.476	0.341	0.519	0.392	0.682	0.502	**0.496**
BayesianBoosting-NaiveBayesPoisson	0.391	0.581	0.514	0.500	0.654	0.582	0.750	0.602	**0.572**
BayesianBoosting-DecisionTree	0.449	0.409	0.362	0.295	0.284	0.342	0.402	0.398	**0.368**
BayesianBoosting-LinearSVM	0.478	0.581	0.476	0.341	0.519	0.392	0.682	0.533	**0.500**
MultiBoosting-NaiveBayeGaussian	0.493	0.602	0.543	0.455	0.580	0.532	0.697	0.578	**0.560**
MultiBoosting-DecisionTree	0.319	0.516	0.486	0.432	0.457	0.405	0.477	0.404	**0.437**
Proposed Methods									
DecisionTree - NB & SVM	0.507	0.634	0.623	0.477	0.605	0.582	0.742	0.645	**0.602**
Neural Network - NB & SVM	0.522	0.624	0.592	0.477	0.593	0.544	0.758	0.645	**0.594**

Table 5.2: Decoding performance for Animal Z
The best performing 3 learners' results are colored (darkest for the best one) for every experiment.

Record IDs :	B0206	B0208	B0214	B0523	B0626	B0729	B0731	B0828	AVERAGE
Generic Algorithms									
NaiveBayes-Poisson	0.317	0.336	0.360	0.244	0.298	0.331	0.261	0.335	0.310
NaiveBayes-Gaussian	0.331	0.353	0.377	0.225	0.316	0.344	0.261	0.341	0.318
BayesianLogisticRegression	0.245	0.387	0.333	0.206	0.237	0.318	0.228	0.293	0.281
BayesNet-K2	0.266	0.395	0.360	0.169	0.202	0.248	0.152	0.263	0.257
DecisionTree	0.129	0.387	0.228	0.225	0.246	0.248	0.272	0.281	0.252
NaiveBayesTree	0.216	0.193	0.158	0.163	0.167	0.242	0.250	0.192	0.197
kNN	0.317	0.294	0.289	0.225	0.228	0.293	0.272	0.311	0.279
Perceptron	0.295	0.429	0.289	0.200	0.289	0.261	0.207	0.269	0.280
MultiLayerPerceptron	0.338	0.328	0.368	0.256	0.246	0.338	0.304	0.317	0.312
LinearSVM	0.302	0.294	0.325	0.250	0.237	0.306	0.272	0.269	0.282
RbfSVM	0.345	0.353	0.368	0.269	0.246	0.331	0.283	0.359	0.319
Ensemble Methods									
Adaboost-NaiveBayesPoisson	0.295	0.420	0.325	0.244	0.298	0.293	0.293	0.335	0.313
Adaboost-NaiveBayesGaussian	0.259	0.403	0.298	0.225	0.316	0.293	0.250	0.341	0.298
Adaboost-DecisionTree	0.151	0.345	0.237	0.213	0.263	0.261	0.272	0.263	0.251
Adaboost-kNN	0.245	0.092	0.281	0.094	0.096	0.248	0.174	0.281	0.189
Adaboost-Perceptron	0.216	0.387	0.289	0.225	0.254	0.287	0.098	0.240	0.249
Adaboost-LinearSVM	0.252	0.403	0.289	0.244	0.263	0.287	0.293	0.311	0.293
BayesianBoosting-NaiveBayesPoisson	0.288	0.353	0.307	0.256	0.316	0.331	0.261	0.257	0.296
BayesianBoosting-DecisionTree	0.288	0.328	0.377	0.238	0.254	0.306	0.261	0.317	0.296
BayesianBoosting-LinearSVM	0.252	0.403	0.289	0.225	0.254	0.287	0.293	0.305	0.289
MultiBoosting-NaiveBayeGaussian	0.353	0.370	0.368	0.250	0.281	0.331	0.293	0.323	0.321
MultiBoosting-DecisionTree	0.259	0.387	0.333	0.294	0.316	0.306	0.304	0.287	0.311
Proposed Methods									
DecisionTree - NB & SVM	0.317	0.378	0.351	0.263	0.325	0.338	0.272	0.377	0.327
Neural Network - NB & SVM	0.295	0.328	0.368	0.238	0.289	0.350	0.283	0.353	0.313

Table 5.3: Variance of selected decoders among recording days
The Ensemble approach has lower variance compared to its base learners.

	Animal Z	Animal S
NaiveBayes-Poisson	0.00160	0.01232
Rbf-SVM	0.00218	0.00861
DecisionTree - NB & SVM	0.00152	0.00685

5.5 Discussion

In Chapter 3 and in Chapter 4, we have shown that intended discrete goals and state timings can be successfully decoded from prefrontal and parietal regions with different classifiers. Speed and the ability to plan ahead are the main advantages of goal directed decoding over more conventional trajectory decoding approaches. Such goal decoding can be ideal for various applications such as typing, selecting among different user interface components or even controlling a robotic hand. In this section, we have compared different learning algorithms (Naïve Bayes with Poisson and Gaussian distributions, Bayesian Logistic Regression, Bayesian Network, k-NN, Decision Tree, Naïve Bayes Tree, Simple Perceptron, Multi-Layer Perceptron, and Support Vector Machines with linear and Rbf kernels) on the same data set for our main task, i.e. decoding among 10 hand postures given the neural data from AIP and F5. This comparison analysis was carried out in an offline setting while searching for an improved decoder for the grasping prosthetic device in mind. We have presented the decoding results with 24 different learning methods from 8 recording sessions from each animal in total (Table 5.1, Table 5.2).

Strictly speaking, a general claim of optimality cannot be set for any of our decoding algorithms used on this problem. However, for any given set of decoders, we can investigate which ones perform particularly well and robustly. While this is not satisfactory from a theoretical point, it is still important and useful for practical reasons, especially for data sets that originate from primate brains, where we lack a

priori information about the underlying data generating process (cortical structure) and only a limited amount of data is available. As we have stated in the previous section, the "no free lunch" theorem suggests that there is no single learning algorithm that in any domain always induces the most accurate learner. Intrinsic set of assumptions in every learner introduces a bias and this leads to error if the assumptions do not hold for the data. In system neuroscience, up to date we still lack a general theory that captures all elements of how the cortical computations arise and decisions are taken robustly based on noisy data. Therefore instead of proposing models which can capture the underlying computation, we utilized here a fully data driven approach to aim at the highest decoding accuracy for our particular problem and data set. To do so, the performance of different learners is quantified by cross validations. We want to emphasize the fact that we don't claim optimality in a strict and universal sense for neural decoding here. Our aim is to point towards a direction of more effective employment of the brain signals for discrete spatial decoding for hand grasping with signals from AIP and F5.

Different decoding algorithms have been applied in brain machine interface experiments till to date. For trajectory decoding, population vector analysis, generalized linear models, Kalman filters or particle filtering are among the common approaches. For discrete decoding, the algorithms that have similarities with working principles of the brain proved to be most successful (Andersen et al. 2010). Naïve Bayesian decoder which calculate maximum likelihood of an intended discrete movement (Gao et al. 2002, Shenoy et al. 2003, Scherberger et al. 2005) is the most commonly used algorithm in this class. Bayesian analysis decodes neural population signals better than, or as well as, all alternative methods (Zhang et al. 1998, Salinas et al. 1994, Sanger et al. 1996). Indeed, Bayesian analysis can produce accuracy that is within a factor of two of the theoretical lower limit of the error given by the Cramer-Rao bound (Oram et al., 1998). Besides the practical success of Bayes classifiers observed in the literature, recent modeling studies have suggested that cortical areas represent probability distributions and may use Bayesian inference for decision making (Beck et al. 2008). One might wonder how such complex computations could be carried out in the cortex. In the last decade, it has been also

shown how optimal ML estimation can be done with a biologically plausible neural network (Deneve et al., 1999) and how recurrent networks of units with bell shaped tuning curves can be wired to implement a close approximation of ML estimators (Pouget et al., 2000).

Also, in our study Naïve Bayes classifiers turned out to be among the best performers (Table 5.1, Table 5.2). In contrast to chapter 3, here we have also utilized NB classifiers with Gaussian firing rate assumption in addition to our usual Poisson assumption. We wanted to simply investigate implications of relaxing the constraint on variance being equal to the mean as in Poisson and observed improved results in some cases. Among other classes of learners, we have obtained improved decoding accuracy in some cases with Support Vector Machines and with some ensemble methods.

SVMs are a family of supervised learning methods which find the optimal hyperplane to maximize the margin between class members. By solving a quadratic optimization problem, they find the global minima and guarantee an upper bound on the generalization error (Burges 1998, Shawe-Taylor et al. 2004). They usually utilize a kernel function to non-linearly map feature vectors to a high-dimension space to achieve linear seperability. After proper treatment they are one of the best out off the shelf learners currently. Being very strong learners, SVMs are also proven to be effective learners in our research.

Among boosting based ensemble methods, Bayesian Boosting algorithm provided the best average accuracy. The idea in ensemble methods is to divide a complex task into simpler tasks that are handled by separately trained base-learners. Each base-learner has its own task. Different algorithms make different assumptions about the data and lead to different classifiers. Boosting based methods utilize different input data to generate complementing learners. Another approach in ensemble learning is to combine different classes of learners, (one base-learner may be parametric and another may be nonparametric) and let a top-level learner make the final decision. When we compare a trained combiner as we have in stacking, with a fixed rule such as in voting, we see that both have their advantages: a trained rule is more flexible and may have less bias, but adds extra parameters, risks introducing

variance, and needs extra time and data for training. Note also that there is no need to normalize classifier outputs before stacking. We have proposed a stacked generalization implementation in this work and obtained best results actually with this learner.

An important point to note is that we have observed that some stronger classifiers did not perform as well as Naïve Bayes classifiers. Especially the inferior results of ensemble methods that utilize Naïve Bayes classifiers as base learners compared to a single Naïve Bayes classifier, brings us to the conclusion that the Poisson firing rate model assumption is indeed close to reality and naïve independence firing rates assumption is not significantly off, at least for the sampled neuron populations. Thus, Naïve Bayesian classifiers are already doing well in capturing population coding characteristics of motor cortical areas. One important observation is that SVMs perform identical to Naïve Bayes in terms of average performance while they both show different characteristics in different recordings. Thus, one can speculate that once population correlations increase or the firing characteristics of individual neurons deviate from Poisson, a strong data driven method like SVM might outperform Naïve Bayes classifiers. This was the main motivation for the proposed stacking ensemble method. In the base level both SVMs and Naïve Bayes classifiers learn the data independently. At the top level, there is a final decision maker that has access to the outputs of both classifiers. We have tried to use decision trees and back-propagation neural networks for this top-level learner and decided to stick to decision trees based on the results and implementation constraints. Decision trees were made popular in statistics in Breiman et al. 1984 and in machine learning in Quinlan 1986 and Quinlan 1993. They learn and respond quickly, and are accurate in many domains (Murthy 1998). Interpretability of its outcomes is another desirable property.

This approach indeed turned out to be the best performing classifier in our analysis. Furthermore, it also had minimum variance among the best performing approaches (across daily sessions), i.e. it is a more robust learner compared to other learners with good prediction accuracies (Table 5.3). Some researchers (Guo et al. 1992), proposed to combine the simplicity of trees with the accuracy of multilayer

perceptrons. Many studies, however, have concluded that the univariate trees are quite accurate and interpretable, and the additional complexity brought by linear (or non- linear) multivariate nodes is hardly justified. A recent survey is given by Rokach and Maimon (2005). It is not hard to reach a similar conclusion also in our work; whether the performance increase by utilization of such a complex approach is significant enough is open to debate. However, increased robustness (having less variance in decoding performance) among different classification sessions is a very important outcome for practical applications. Thus, we believe that by providing an average higher performance more robustly, such an ensemble of classifiers may have important implications for future clinical applications.

Further points to consider for an improved decoding:

Cortical plasticity: Several studies have recently documented the occurrence of cortical plasticity as animals learn to operate a BMI (Carmena 2003, Taylor et al. 2002, Lebedev et al. 2005). This phenomenon is characterized by changes in the tuning properties of individual neurons and physiological adaptations at the level of neural ensembles, which include changes in firing covariance and spike timing (Nicolelis et al., 2009). This learning can be observed in various different time resolutions; over a few minutes to hours of training (Fetz 1969, Jarosiewicz et al. 2008, Moritz et al. 2008), over a period of days (Mulliken et al. 2008), or even over a period of weeks (Varmena et al. 2003, Musallam et al. 2004, Taylor et al. 2002). The brain is arguably the most adaptive learning machine in known existence which can handle virtually unlimited complexity robustly. Thus, once it realizes feedback coming from the decoding system, the neurons that provide input to the BMI can adapt their tuning curves towards a more desirable direction. Please note that, this may actually yield problems for adaptive learning algorithms via feed-forward loops. If the learner is fixed it is easier for brain to capture its characteristics and adapt itself accordingly. On the other hand, a continuously updated algorithm is a moving target for the brain which facilitates further change in the brain; this again can create a

forward feedback for algorithm to adapt to. Our initial experimentation with an adaptive Naïve Bayes implementation in real-time confirmed this (data not shown), i.e. the adaptive version usually provided worse decoding accuracy than the static version.

Correlation structures: We believe that tracking the covariance structure of data can be beneficial for decoding purposes. The recent availability of multiple electrode recordings provides the opportunity to measure covariances among recorded neurons directly and makes tracking them in real-time feasible. Modeling multi-dimensional probability distributions can be extended to include temporal aspects of the neural code this way systematically. Changes in the pattern of correlations between cells may occur in addition to changes in signal with a changing stimulus set, attention state or behavioral task. (Andersen et al. 2010). This could have significant effects on the relevant information content of neural populations and the brain may actually be using such information internally. Thus, analysis methods incorporating such measures can provide superior decoding. In recent literature, we can find evidence suggesting that decoding can be improved by taking into account correlations between spike trains (Abbott et al. 2009, Averbeck et al. 2004, Brown et al. 2004, Nirenberg et al. 2003) and the temporal regularities in responses (Musallam et al. 2004). A similar challenge may be posed while decoding from multiple cortical areas, which allows measures of LFP-LFP and spike-LFP coherences (Parseran et al. 2008). These measures, particularly spike-LFP measures, may indicate changes in the communication between areas and may provide additional insights into cognitive functions and refinement of cognitive decoding algorithms (Andersen et al. 2010). Furthermore, one can also utilize information theoretic approaches if capturing second moments of joint distributions is not believed to be enough. One can speculate that some of the extra performance we have observed with stronger learners in some days may be arising from capturing intrinsic correlations in the data better and with a more structured approach to benefiting from these correlations. Therefore, it may be possible to improve the decoding performance of brain machine interfaces by taking correlation structures into account.

Information theory based approaches for an objective measurement of information content of neuronal population data:

Developed in the 1940's in Bell Laboratories by one of the most influential scientists of the last century, Claude E. Shannon, information theory is a complete mathematical framework for quantifying information transfer on noisy communication channels (Shannon, 1948). It has been applied very successfully not only to many engineering problems related to communication systems and data compression but also yielded to new research on diverse topics; like statistical inference, evolutionary biology, quantum computing and also on system neuroscience.

Information theory can be used to quantify the stimulus–response functions in neural coding and to measure the coding efficiency of developed models. It has complementary characteristics to decoding techniques in general by providing a mathematically rigorous way to quantify how much of the information provided by neuronal populations concerns the prediction of the stimulus, versus how much of this information is about specific aspects of the uncertainty of these predictions. A combination of decoding and information theoretic approaches therefore is being an active research area in systems neuroscience lately, with the hope to provide precise quantitative answers about how the brain deals with intrinsically noisy signals (Quiroga et al. 2009). Below we will first describe a generic approach on utilizing information theory for neural coding problem and after we will speculate on the possible usage scenarios in our research.

If a stimulus s belonging to a set S is presented with a probability P(s), a cornerstone concept in information theory, entropy - H(S), can be defined as:

$$H(S) = -\sum_{s} \mathrm{P(s)} \log_2 \mathrm{P(s)}$$

This is essentially the quantification of uncertainty about which stimulus is presented and usually measured in bits, as the base 2 is used for the logarithm. A strictly non-negative measure, it reaches a maximum value of $\log_2 \mathrm{N}$ for S consisting of N

elements, when each N stimuli have equal probability of presentation as in our experiment. If the neuronal population response r contains information about the stimulus, then its observation should reduce the stimulus uncertainty. From there we can calculate similarly the entropy of the posterior distribution P(s|r):

$$H(S|R) = -\sum_{s,r} \mathrm{P(r)P(s|r)}\log_2 \mathrm{P(s|r)}$$

This entropy remaining in the stimulus after observing the neuronal response is called equivocation. And mutual information is defined as the reduction of uncertainty about the stimulus obtained by knowing the neuronal response. It is given by the difference between the stimulus entropy H(S) and the equivocation H(S|R):

$$I(S;R) = -\sum_{s,r} \mathrm{P(r)} \cdot \mathrm{P(s|r)}\log_2 \frac{\mathrm{P(s|r)}}{\mathrm{P(s)}}$$

$$= -\sum_{s,r} \mathrm{P(s,r)}\log_2 \frac{\mathrm{P(s,r)}}{\mathrm{P(s)} \cdot \mathrm{P(r)}}$$

In other words, mutual information can quantify the information on the stimulus we obtained from neural code. If it is possible to reach to a perfect stimulus reconstruction given the neural code, one expects the mutual information of being equal to the entropy of stimulus, H(S). Therefore it also defines an upper-limit for the decoding approaches and clearly relevant to what we were trying to achieve in this chapter. However, there are some practical difficulties of proper application of this approach.

P(s,r) denote the joint probabilities of observing the response r with the stimulus s. These are clearly not known in advance and need to be properly estimated from data in a scenario where we don't know much about the data generating processes. The major difficulty is the fact that estimation of P(s,r) becomes intractable for an animal experiment whenever r contains information from more than a few neurons. The cardinality of the set of all possible responses R increases exponentially with the number of neurons and becomes even larger if we also consider temporal firing patterns. Despite growing interest in more efficient estimation techniques and bias correction methods in the last years, the minimum

number of trials per stimulus that is needed to obtain an unbiased information calculation for neural populations is still approximately in the order of the cardinality of the response set R (Panzeri, 2007). The "curse of dimensionality" prevents the application of information theory to large populations. For illustration purposes; if we assume that we have 100 neurons and we have discretized their firing response space pretty coarsely only by 10 bins, we still need $10^{100} \times N$ samples. It will take a while for the animal to execute that many trials (simple algebra suggests that this will indeed last very much longer than the age of known universe)!

Actually, here we are facing a similar problem about the estimation of joint probability distribution as in our Bayesian decoder. There we have tackled it by making the naïve assumption that neurons are firing independently and represented the joint probability as a multiplication of individual factors (neuron firing probabilities given the stimulus) each with Poisson characteristics. Clearly we can make the very same assumptions here and try to calculate the amount of information obtained via the neural code afterwards but the value we have calculated that way will have the same intrinsic weaknesses of the Naïve Bayesian decoder and won't tell us about the real optimal limit of information that can be captured from the neuronal data.

In other words, the real probabilities are about non-stationary point processes, however due to practical reasons virtually for all the real-life scenarios need to employ some modeling assumptions and data transformations. Examples are: naïve independence firing assumption, binning the spikes into fixed time windows or rate coding assumption. According to "data-processing inequality" every additional transformation can only decrease the Shannon information that was accessible from the original responses. Therefore unless we are given the real probability distributions by a "heavenly creature" the calculation of real limit values of an optimal learner will only be theoretically possible.

However, this does not mean that information theory cannot provide additional information on top of decoding approaches. As an example, we can see the difference between a Bayesian decoder giving the most likely stimulus as the output (arg max P(s|r)), versus information theory providing a smooth integration of

information over the whole posterior probability P(s|r). Therefore we can claim that information theory in general provides a more comprehensive treatment of information contents of the data. That means we can formalize decoding as a second level of transformation since neurons can convey information by means other than that can be captured by a decoder (in the form of most likely stimulus). Information theory is the analytical tool that is utilized in the literature to capture this additional information. Indeed, in one study (Robertson et. al., 1999) it has been shown that, a Bayesian decoding algorithm captures ~95% of the total information available from the neuronal responses (calculated after some approximation) that is represented by the population activity of head-direction cells in primate.

One way to link information theory and decoding is to compute the mutual information between the actual and the predicted stimuli from the decoding outcomes, $I(S;S_{predicted})$. It is pretty straightforward to calculate the decoders' mutual information once we have the confusion matrix. And from there on, either we get the real probability distributions from our "heavenly observer" to calculate the total information encoded by neural code I(S;R) or more likely make some assumptions and estimate an approximate of that value, $I(S;R)$. The difference between the real I(S;R) and $I(S;S_{predicted})$ is the measure of distance of our decoder from the optimal utilization of neural information but again this value cannot be really calculated. However, calculation of $I(S;R) - I(S;S_{predicted})$ is plausible and it essentially gives the amount of information available in the neuronal responses that could be gained by means other than decoding the most likely stimulus and this information can be useful in some applications.

As demonstration purposes we have calculated the average mutual information values for the decoders employed in Chapter 3 for two animals. By utilizing the average confusion matrices in Fig3.7A and Fig3.7B, we have obtained values of 1.15 bits and 0.84 bits of mutual information for the decoders for Animal-S and Animal-Z, respectively. Every bit of information provided by the decoders reduces the overall uncertainty about the stimulus by a factor of two. Therefore, it is not surprising to see a value above 1 for Animal-S's decoder where we have an average decoding accuracy above 50% and vice-versa for Animal-Z. Also note that

maximum information that a perfect decoder can give in this particular setup is $\log_2 10 \approx 3.32$ bits. By the observation of positive values for decoder mutual information we can conclude that the decoders are indeed converting neural signals to stimulus response information successfully (this is similar to saying that average decoding accuracies are significantly above the chance level), however it is also clear that they are far away from employing a perfect transformation, while missing some ~2 bits compared to perfect decoding. Similarly, we could have employed this calculation to all the confusion matrices of different learners employed in this chapter and could come up with another metric for comparing these learners.

By utilizing different classes of learning algorithms in this chapter, we have tried to empirically show that it is possible to extract some extra accuracy compared to a standard Naïve Bayesian classifier. However, our results suggested that this possible accuracy gain after trying many learners is not very big and therefore we have the tendency to speculate that the neural population we are utilizing lacks enough information to achieve a perfect decoding. Mutual information calculations can be useful for putting a number on that missing information amount.

Besides the application we have provided above, information theoretic approaches can be effectively utilized in other cases. Some possible usages are; estimations of neural code's efficiency by comparing H(R) to H(R|S), tests on temporal precision of neural code (by utilizing concepts from Nyquist limit), characterization of information for continuously time-varying stimuli, feature selection for model building, capturing high order correlation structures and merging information from different neuronal signals. In some of these possible application fields the curse of dimensionality limits the practical outcomes currently, however with the advances in data collection technology, available computational power and new statistical methods this rigorous mathematical framework will definitely contribute to our understanding of brain mechanisms and neural coding in the near future.

Other signal modalities (LFP): The local field potential (LFP) is an aggregate signal that represents the net excitatory and inhibitory synaptic and dendritic potential

around the tip of the recording electrode (Mitzdorf 1987, Buzsaki 2006). LFP is easy to record and more stable over long time periods than the spiking activity of single units. Therefore, the LFP has been suggested as an input signal in future brain-machine interface applications (Andersen et al., 2004). Scherberger et al. (2005) analyzed and compared both, spiking activity and LFPs taken from PRR of a monkey performing a delayed reaching/ saccade task. Baumann et al. (2009) compared the coding properties of the LFP with the one of multi unit spiking in AIP. Spikes and LFPs reflect to certain degree different sources, with spikes more indicative of cortical outputs and LFPs indicative of inputs and intracortical processing. A better understanding of the relationship between the LFP and local spiking and proper combination of these signals might therefore give new tools to better study the local processing and contribute to an improved decoding algorithm for brain machine interfaces.

Last but not least, motivation and reward expectation can typically be decoded from parietal cortex in similar setups that we have utilized in this work. Such signals may be useful for brain machine interfaces by registering the preferences and mood of subjects and patients. And combining this information with more conventional signals may prove to be useful. The research on different decoding methods for brain machine interfaces has taken off only in the last decade with increasing data availability and computational power. We believe that the efforts in this line of research will significantly contribute to improved quality of life for patients in the future.

6

Conclusion

In this final chapter, we want to highlight the major findings of this thesis for easy reference and drawing a final conclusion. Please note that, these are mostly simplified repetitions of what we have discussed in detail at the end of each relevant chapter from chapters 3 to 5.

6.1 Major findings of present thesis

6.1.1 Real time decoding of hand grasping signals:

In Chapter 3, we aimed to answer the main research question of this work; namely, in detail feasibility analysis of a real time neuroprosthetic grasping with signals coming from higher order cortical areas, AIP and F5. We have indeed found promising results which supported our initial hypothesis that grasp decoding may be possible from abstract plan signals. The outcome of this work is published in Journal of Neuroscience in 2011 (Townsend, 2011). Below are some important points regarding these findings;

- We managed to collect samples of tuned multi-units in both areas from both monkeys via chronically implanted floating micro-electrode arrays (FMA) and confirmed that information about grip type and orientation was present in the neural data.
- The total numbers of average significantly tuned units for two animals were significantly different (around 121 for Animal S and 56 for Animal Z, in average). And this is also reflected in decoding performances.
- We have observed rather heterogeneous distributions of significantly tuned units across FMAs and found that sampling of tuned activity was concentrated on certain individual electrodes or groups of electrodes. This may motivate interesting questions on electrode array design for future studies.
- Both single neuron recordings and population level analysis of the spiking characteristics of two brain regions confirmed previous single-electrode study findings for the same regions (Murata et al., 1997; Raos et al., 2006; Umilta et al., 2007; Baumann et al., 2009; Fluet et al., 2010). In summary, F5 showed moderate tuning to orientation early in the task and strong grip type tuning, whereas AIP showed a clear tuning for grip type and orientation starting from Cue period throughout the task. These findings indicate that distinct representations of both task parameters exist in the multiunit activity of both areas during movement planning, which can be exploited by the BMI paradigm.
- We have observed a higher preference for precision grip (Fig 3.4 A), and a uniform distribution of preferred orientations (Fig. 3.4 C) in F5. On the other hand, the majority of AIP multi units coded predominantly for the extreme handle orientations during planning (Fig. 3.4 D), whereas they did not show a significant preference for grip type (Fig. 3.4 B).
- One important finding was to show that the tuning pattern characteristics of units from both areas during movement planning remained essentially constant while the monkey performed the real time decoding task (Fig. 3.5).

This was an important finding in terms of claiming feasibility for a realistic neural-prosthesis.

- After showing the favorable characteristics of tuning for both areas for decoding, we moved to real-time decoding experiment, which is one of the most important outcomes of this work. There we showed that for both animals decoding is plausible in real-time. The average decoding accuracies remained consistently above chance level for both animals with a mean performance of 50.4% (±7.6%) in monkey S and of 33.5% (± 5.9%) in monkey Z.
- Further analysis on separate decoding accuracies of grasp type and orientation showed that the classification of the grip type was always highly accurate in both monkeys (monkey S mean: 85.5%, monkey Z mean: 90.6%). Decoding of orientation was performed with less accuracy in both animals (56% in monkey S; and 35.5% in monkey Z). This was expected to some extent, given that there was more orientation (5) than grip type conditions (2) to classify. However, the orientation accuracy in monkey Z was rather poor, most likely due to low number of tuned units from AIP.
- In order to test the effect of on-line spike sorting algorithm, we have re-run the same decoding analysis in an off-line setting while using a more sophisticated spike-sorting algorithm. In both animals, decoding performance increased in 24 out of 26 decoding sessions with a mean increase of 6.2 % (monkey Z) and 6.3 % (monkey S).
- All together, we showed that F5 tended to perform better than AIP at grip type-only decoding, and AIP consistently outperformed F5 during orientation-only decoding. However, utilization of data from both areas was necessary for optimal decoding accuracy of grip type and orientation in the 10-condition task. Further ROC analysis of individual neurons confirmed these findings.
- Overall, these results demonstrate the real time decoding of intended grasping goals using multi-unit signals from higher-order motor areas obtained during planning of these movements, and underscore the importance of utilizing

signals from multiple cortical areas for control of BMIs to restore movement function. Research in the field of BMIs for motor control has seen rapid expansion in recent years (Scherberger 2009). And a key approach in the literature has been the closed-loop decoding of 2D and 3D arm and hand trajectories, derived mainly from M1, to control robotic arms for grasping objects or to decode individual finger movements. A key difference in the present study was our decision to target discrete decoding and using signals from parietal and frontal areas which are thought to play a role in sensorimotor integration during movement planning. In chapter 3 we managed to show this to be a feasible approach for neuroprosthetic devices.

6.1.2 Temporal Decoding of Grasp Execution:

In chapter 4, we claimed that a fully-autonomous neuroprosthetic device should be able to differentiate among different cognitive states and act upon them accordingly. In other words, in this chapter we aimed to decode the temporal component of the behavioral signal and infer when to grasp instead of how/what to grasp. To do that we followed the steps from successful studies from literature in terms of decoding paradigm to be utilized (Shenoy et al. 2003, Achtman et al. 2007, Kemere et al. 2008). These studies have already shown that it is possible to decode different behavioral states with significant success using major motor cortical areas, mainly PRR/MIP, PMd/M1. However, we showed that it is also possible to decode brain states for grasping from cortical areas AIP and F5. To do so, we have re-formulated our objective as decoding the experimental states and defined our framework as making predictions on a sliding window of length 250 ms of neural spike rate data. The outcome of this research was presented in an international conference, in Neuroscience 2008 Washington (Subasi et al., 2008).

- We first used a Naïve Bayesian decoder and managed to get promising decoding accuracies; 65% (SD 15%) for Animal Z and 82% (SD 16%) for Animal S.

- Later, in order to utilize the temporal structure of states, we introduced a Finite State Machine of a fixed Markovian order and achieved a decoding performance of 69% (SD 15%) for animal Z and 88% (SD %15) for animal S, with also improved robustness.
- Finally, in an attempt to capture the temporal stochastic properties of underlying states more effectively we utilized a simple hidden markov model to obtain decoding accuracies of 61% (SD 9%) for Animal Z and 85% (SD 10%) for Animal S. The main contribution of HMM was increased robustness.
- Furthermore, in the second part of this chapter, with an attempt to improve our base decoder, we redefined the problem as a dichotomy and tried to classify the movement period vs. non-movement periods among the behavioral states. We used different machine learning algorithms (Bayesian Logistic Regression, Naive Bayes, k-Nearest Neighbour and Support Vector Machine) and showed that some significant increase in decoding accuracy and robustness is possible with stronger learners (Table 4.1). The best results here were obtained with Support Vector Machines with linear kernels both in terms of prediction accuracy and robustness. SVM's also showed a desirable characteristic for a real time setup of faster classification once the training is done.

6.1.3 In Search of a More Robust Decoding Algorithm:

In the 5th chapter, we put our efforts to search for an better decoding algorithm for the data set at hand. In system neuroscience, up to date we still lack a general theory that captures all elements of how the cortical computations arise and decisions are taken robustly based on noisy data. Therefore instead of proposing models which can capture the underlying computation, we utilized here a fully data driven approach to aim at the highest decoding accuracy for our particular problem and data set. To do so, the performance of different learners is quantified by cross validations. In other words, we have followed a data mining approach to find an optimal decoder for our data set while being careful against over-fitting. We have used the multi-unit

neural spike data with similar characteristics presented in the 3rd Chapter; but via use of a family of standard learning algorithms and their combinations we have investigated if it is possible to find better performing decoding algorithms for our existing brain machine interface implementation. In total, we have presented the decoding results with 24 different learning methods from 8 recording sessions from each animal (Table 5.1, Table 5.2). The outcome of this work was published and presented in a peer reviewed IEEE conference in 2010 in Buenos Aires (Subasi, 2010).

- We have formulated our decoding task as a classification problem and first applied 11 different learning algorithms (Naïve Bayes with Poisson and Gaussian distributions, Bayesian Logistic Regression, Bayesian Network, k-NN, Decision Tree, Naïve Bayes Tree, Simple Perceptron, Multi-Layer Perceptron, and Support Vector Machines with linear and Rbf kernels) to our data set.
- Among single learners, Naïve Bayes (NB) and SVM-Rbf showed best average decoding accuracy (~58%) for both Animal-S and Animal-Z (~32%).
- Furthermore, another set of 11 algorithms were constructed using boosting based ensemble methods (Adaboost, Multiboosting, Bayesian Boosting) which utilize some learners from the previous set as their base learners.
- Among these ensemble methods, BayesianBoosting with NB-Poisson base learners had the best average accuracy for Animal-S. For Animal-Z, MultiBoosting and Adaboost showed best results with both NB-Poisson base learners.
- In general, boosting based ensemble methods improved accuracy in some cases for Animal-S, but were beaten in average performance in the end. For Animal-Z, on the other hand, we observed a better performance with a MultiBoosting ensemble compared to single learners' winner SVM-Rbf. Therefore one can speculate that whenever the signal-to-noise ratio is high and the model assumptions are violated heavily, ensemble approaches may outperform state of art single learners.

- Finally, after collecting results with all of the above mentioned methods, we designed two new ensemble stacking models. Here we wanted to combine different characteristics of model-based and discriminant-based learners in an optimal setup. As the root learner we selected a decision tree finally, and as base learners we utilized NB-Poisson, NB-Gaussian and Rbf-SVM.
- We observed that, this approach not only provided the best average accuracy for both animals (~60% for Animal S, ~33% for Animal Z) but also gave most robust decoding (minimum variance among different days' data – Table 5.3).
- All in all, the more sophisticated treatment of the data showed that it is indeed possible to get some improvements in terms of both decoding accuracy and robustness. However, similar to our findings in chapter 3, we have observed a significant difference for decoding accuracies of both animals' data, which tells us the main limiting factor is data quality not the analytical methods to analyze it. Also, Naïve Bayes classifiers, in general, ranked always close to the top decoders, therefore it is not straight forward to argue in favor of more complex learners all the time.
- To sum up, we argue that claiming universal decoding-optimality using such data mining approaches is not possible mainly due to lack of understanding of the underlying cortical principals and that learning in general is an ill-posed problem. However, we still believe that results presented here will point towards the right direction for an improved discrete spatial decoding with higher order cortical signals, and, by providing such a formal comparison of different analytical approaches in a systematic fashion, we hope to contribute to future research in this fascinating field.

6.2 Final Words

"There are good reasons to believe that we are at a turning point, and that it will be possible in the next two decades to formulate a meaningful understanding of brain function"

Lloyd Watts, 2003. (Fogel et al., 2003)

The prominent inventor and neuroscientist Dr. Watts had an optimistic view about the imminent future of our understanding of brain function, as seen in 2003 published book "Computational Intelligence: Experts Speak". He argued that; "Scientific advances are enabled by a technology advance that allows us to see what we have not been able to see before. At about the turn of the twenty-first century, we passed a detectable turning point in both neuroscience knowledge and computing power. For the first time in history, we (collectively) know enough about our own brains, and have developed such advanced computing technology, that we can now seriously undertake the construction of a verifiable, real-time, high-resolution model of significant parts of our own intelligence." He further reasoned "Revolutions in scientific research for specific areas directly correlated to breakthroughs in instrumentation. Our society witnessed this in physics, astronomy and biology in the past centuries many times. One can argue that a similar course is happening in neuroscience today."

Here, if we focus our attention to neural interface technologies we can provide some already successful brain machine interface applications like, cochlear implants and deep-brain stimulation devices (for Parkinson's disease) as evidences of this breakthrough, and we can mention the possible usage of upper-limb prosthetic BMIs for spinal cord injuries/amputees as the next logical step. Furthermore, we also believe that BMIs will play a very important role in society in the future, not only in clinical applications but also in learning about the operation and the mechanism of the brain which will ultimately have huge impact in completely different domains (e.g. computation).

On the other hand, Dr. Carver Mead, one of the living legends of computing in academia and industry, a pioneer not only in solid-state electronics and VLSI

circuits but also in neuromorphic systems design (actually he also happened to be the doctorate supervisor of Dr. Lloyd Watts), does not sound necessarily as optimistic as Dr. Watts, in an interview given to American Spectator Magazine (September/October 2001, Vol. 34 Issue 7). He says that;

"Biological solutions are many orders of magnitude more effective than those we've been able to implement using digital methods." He gives fruit flies as an example and continues; "The fly has an autonomous system that avoids being swatted. It has the ability to see and navigate and make decisions on millisecond time scales. We've never been able to make artificial vision systems that come within orders of magnitude of that, with all the computation we can throw at them."

His answer to journalist's "Why not?" question is interesting;

"That's what I was trying to find out. It makes us look so stupid. And you don't get popular by saying that. But it's true. And the more we try, the more we realize it's a much harder problem than we thought. What is it about the way that the fly, or the cat, or the fish process their information that makes it so much more effective at computing these things? They use what seems like really slow, slimy computational material, and yet they perform miracles with tiny amounts of power, tiny amounts of space and in real time and very fast." Finally, he concludes on the source of the problem as follows;

"We don't know how even to formulate that problem and we've been working on it since the dawn of computing. Every time we get another order of magnitude in computing capability, somebody says, "Now we've got enough!" But we haven't begun to get it."

Now, after almost a decade later from Dr. Mead's and Dr. Watts' assertations, we still cannot know who will be proven to be right in two decades time. In the last decade, we have witnessed some promising developments both in system neuroscience and instrumentation. BMI research accelerated significantly; number of publications and researchers in the field is increasing drastically in the last years, we have already seen upper-limb prosthetic devices applied successfully to humans (Hochberg et al., 2006), stimulation technology maturing and some successful visual

prostheses implanted to first patients. In parallel, neuromorphic-engineering is developing with an increasing pace and it is almost certain that we will be exposed to this technology more and more in our daily lifes in the future.

However, we still think that it is very early to declare victory in our understanding of cortical computation. We believe that BMI applications ultimately will hold very important implications in this line of research as well. Such a development eventually may lead even to a paradigm shift in our understanding of computation in general.

But before this happen, we will first witness BMIs contributing to the well-being of patients of various neurological and motor disorders. In this thesis, we have put our efforts in contributing to the field by improving the available knowledge on grasp related neuro-prosthetic interfaces by analyzing novel brain regions in such a context. Future research will show if our approach will advance the field of neural prosthetics and will bring us several steps closer to restoring limb control in patients or even ultimately contribute to our understanding of brain function.

References

Achtman N, Afshar A, Santhanam G, Yu BM, Ryu SI, Shenoy KV (2007) Free-paced high-performance brain-computer interfaces. J Neural Eng 4:336-347.

Aggarwal V, Tenore F, Acharya S, Schieber MH, Thakor NV (2009) Cortical decoding of individual finger and wrist kinematics for an upper-limb neuroprosthesis. Conf Proc IEEE ENg Med Biol Soc 2009:4535-4538.

Alpaydin E (2010) Introduction to Machine Learning, The MIT Press.

Andersen RA, Hwang EJ, Mulliken GH (2010) Cognitive neural prosthetics. Annu Rev Psychol 61:169-3.

Anderson KD (2009) Consideration of user priorities when developing neural prosthetics. J Neural Eng 6.

Asher I, Zinger N, Yanai Y, Israel Z, Prut Y (2010) Population-Based Corticospinal Interactions in Macaques Are Correlated with Visuomotor Processing. Cereb Cortex 20:241-252.

Bai O, Lin P, Vorbach S, Floeter MK, Hattori N, Hallett M (2008) A high performance sensorimotor beta rhythm-based brain-computer interface associated with human natural motor behavior. J Neural Eng 5:24-35.

Bauer E, Kohavi R (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants In: Machine Learning, 36(1-2).

Baumann MA, Fluet MC, Scherberger H (2009) Context-Specific Grasp Movement Representation in the Macaque Anterior Intraparietal Area. J Neurosci 29:6436-6448.

Borra E, Belmalih A, Calzavara R, Gerbella M, Murata A, Rozzi S, Luppino G (2008) Cortical connections of the macaque anterior intraparietal (AIP) area. Cereb Cortex 18:1094-1111.

Breiman L, Friedman JH, Olshen RA, Stone CJ (1994) Classification and Regression Trees. Belmont, CA: Wadsworth International Group.

Breiman L (1996) Bagging Predictors. Machine Learning 123-140.

Breiman L (2001) Statistical modeling: The two cultures. Statistical Science, JSTOR.

Breiman L (2001) Random Forests. Machine Learning vol. 45 no 1 pp: 5-32.

Brochier T, Spinks RL, Umilta MA, Lemon RN (2004) Patterns of muscle activity underlying object-specific grasp by the Macaque monkey. J Neurophysiol 92:1770-1782.

Brochier T, Umilta MA (2007) Cortical control of grasp in non-human primates. Curr Opin Neurobiol 17:637-643.

Brown EN, Kass RE, Mitra PP (2004) Multiple neural spike train data analysis: state-of-the-art and future challenges. Nat Neurosci 7:456-461.

Buzsaki G. (2006) Rhythms of the Brain. Oxford University Press.

Carmena JM, Lebedev MA, Crist RE, O'Doherty JE, Santucci DM, Dimitrov DF, Patil PG, Henriquez CS, Nicolelis MAL (2003) Learning to control a brain-machine interface for reaching and grasping by primates. Plos Biology 1:193-208.

Cerri G, Shimazu H, Maier MA, Lemon RN (2003) Facilitation from ventral premotor cortex of primary motor cortex outputs to macaque hand muscles. J Neurophysiol 90:832-842.

Choi YS, Koenig MA, Jia XF, Thakor NV (2010) Quantifying Time-Varying Multiunit Neural Activity Using Entropy-Based Measures. IEEE T Bio-Med Eng 57:2771-2777.

Dayan P, Abbott L (2001) Theoretical neuroscience: computational and mathematical modeling of neural systems. Cambridge, MA: MIT.

Cortes C, Vapnik V (1995) Support Vector Networks. Machine Learning 20: 273-297.

Desimone R, Gross CG (1979) Visual Areas in the Temporal Cortex of the Macaque. Brain Res 178:363-380.

Dietterich TG (2003) Machine Learning. In Nature Encylopedia of Cognitive Science, London: Macmillan

Eberhard E Fetz (1969) Operant Conditioning of Cortical Unit Activity. Science: Vol. 163 no. 3870 pp. 955-958.

Fluet MC, Baumann MA, Scherberger H (2010) Context-Specific Grasp Movement Representation in Macaque Ventral Premotor Cortex. J Neurosci 30:15175-15184.

Fogassi L, Gallese V, Buccino G, Craighero L, Fadiga L, Rizzolatti G (2001) Cortical mechanism for the visual guidance of hand grasping movements in the monkey: A reversible inactivation study. Brain 124:571-586.

Fogel DB, Robinson CJ (2003) Computational intelligence: the experts speak. Wiley-IEEE Press; 1 edition (June 16, 2003) ISBN-10: 0471274542

Follett KA (2000) The surgical treatment of Parkinson's disease. Annu Rev Med, 51:135-147.

Frank A, Asuncion A (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Fraser GW, Chase SM, Whitford A, Schwartz AB (2009) Control of a brain-computer interface without spike sorting. J Neural Eng 6.

Gallese V, Murata A, Kaseda M, Niki N, Sakata H (1994) Deficit of hand preshaping after muscimol injection in monkey parietal cortex. Neuroreport 5:1525-1529.

Galletti C, Kutz DF, Gamberini M, Breveglieri R, Fattori P (2003) Role of the medial parieto-occipital cortex in the control of reaching and grasping movements. Exp Brain Res 153:158-170.

Gamberini M, Passarelli L, Fattori P, Zucchelli M, Bakola S, Luppino G, Galletti C (2009) Cortical connections of the visuomotor parietooccipital area V6Ad of the macaque monkey. J Comp Neurol 513:622-642.

Geman S, Bienenstock E, Doursat R (1992) Neural Networks and the Bias/Variance Dilemma. Neural Computation 4: 1-58.

Georgopoulos AP, Kalaska JF, Caminiti R, Massey JT (1982) On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex

Georgopoulos AP, Lurito JT, Petrides M, Schwartz AB, Massey JT (1989) Mental rotation of the neuronal population vector. Science 243 234-236.

Gerstein GL, Mandelbrot B (1964) Random walk models for the spike activity of a single neuron."; Biophys J. 4:41-68.

Guo H., Gelfand S.B. (1992) Classification trees with neural network feature extraction. Neural Networks, 3, 923-933

Hanke M, Halchenko YO, Sederberg PB, Hanson SJ, Haxby JV, Pollmann S (2009). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. Neuroinformatics, 7, 37–53.

Hatsopoulos N, Joshi J, O'Leary JG (2004) Decoding continuous and discrete motor behaviors using motor and premotor cortical ensembles. J Neurophysiol 92:1165-1174.

Hatsopoulos NG, Donoghue JP (2009) The Science of Neural Interface Systems. Ann Rev Neurosci 32:249-266.

Hastie T, Tibshirani R, Friedman J (2009) The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Hochberg LR, Serruya MD, Friehs GM, Mukand JA, Saleh M, Caplan AH, Branner A, Chen D, Penn RD, Donoghue JP (2006) Neuronal ensemble control of prosthetic devices by a human with tetraplegia. Nature 442:164-171.

Hodgkin AL, and Huxley AF (1952) Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo. J. Physiol (London) 116:449-472.

Hubel DH, Wiesel TN (1959) Receptive fields of single neurones in the cat's striate cortex. The Journal of Physiology, 1959 - Physiological Soc.

Hudson N, Burdick J (2007) Learning Hybrid Systems Models for Supervisory Decoding of Discrete State, with applications to the Parietal Reach Region.

Neural Engineering, CNE'07 3rd International IEEE/EMBS Conference on pp. 587-592

Jeannerod M, Michel F, Prablanc C (1984) The Control of Hand Movements in A Case of Hemianesthesia Following A Parietal Lesion. Brain 107:899-920.

Joachims T. (2002) Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms. Kluwer Academic Publishers, ISBN 0-7923-7679-X.

Kakei S, Hoffman DS, Strick PL (1999) Muscle and movement representations in the primary motor cortex. Science 285:2136-2139.

Kakei S, Hoffman DS, Strick PL (2003) Sensorimotor transformations in cortical motor areas. Neurosci Res 46:1-10.

Kass RE, Ventura V, Cai C (2003) Statistical smoothing of neuronal data. Network-Comp Neural 14:5-15.

Kemere C, Santhanam G, Yu BM, Afshar A, Ryu SI, Meng TH, Shenoy KV (2008) Detecting Neural-State Transitions Using Hidden Markov Models for Motor Cortical Prostheses. JN Physiol October 2008 vol. 100 no. 4 2441-2452.

Kuncheva LI (2004) Combining Pattern Classifiers: Methods and Algorithms. Hoboken, NJ: Wiley.

Lebedev M. A., Carmena J. M., O'Doherty J. E., Zacksenhouse M., Henriquez C. S., Principe J. C., Nicolelis M. A. L. (2005) Cortical Ensemble Adaptation to Represent Velocity of an Artificial Actuator Controlled by a Brain-Machine Interface. The Journal of Neuroscience, 25(19):4681-4693

Lemon RN (1993) Cortical Control of the Primate Hand. Exp Physiol 78:263-301.

Lemon RN (2008) Descending pathways in motor control. Annu Rev Neurosci 31:195-218.

Leuthardt EC, Schalk G, Wolpaw JR, Ojemann JG, Moran DW (2004) A brain-computer interface using electrocorticographic signals in humans. J Neural Eng 1:63-71.

Loeb GE (1990) Cochlear prosthetics. Annu Rev Neurosci 1990, 13:357-371.

Luppino G, Murata A, Govoni P, Matelli M (1999) Largely segregated parietofrontal connections linking rostral intraparietal cortex (areas AIP and VIP) and the ventral premotor cortex (areas F5 and F4). Exp Brain Res 128:181-187.

Ma WJ, Beck JM, Latham PE, Pouget A (2006) Bayesian inference with probabilistic population codes. Nature Neuroscience - 9, 1432 - 1438.

Merzenich MM (1983) Coding of sound in a cochlear prosthesis: some theoretical and practical considerations. Ann N Y Acad Sci, 405:502-508.

Mierswa I, Wurst M, Klinkenberg R, Scholz M, Euler T. (2006) YALE: Rapid Prototyping for Complex Data Mining Tasks. Proceedings of the 12th ACM

SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06).

Moran DW, Schwartz AB (1999) Motor cortical representation of speed and direction during reaching. J Neurophysiol 82:2676-2692.

Morgan James P (1982) The First Reported Case of Electrical Stimulation of the Human Brain. J. Hist. Med Allied Sci. 51-64.

Morrow MM, Jordan LR, Miller LE (2007) Direct comparison of the task-dependent discharge of M1 in hand space and muscle space. J Neurophysiol 97:1786-1798.

Morrow MM, Miller LE (2003) Prediction of muscle activity by populations of sequentially recorded primary motor cortex neurons. J Neurophysiol 89:2279-2288.

Mountcastle VB, Lynch JC, Georgopoulos A, Sakata H, Acuna C (1975) Posterior parietal association cortex of the monkey: command functions for operations within extrapersonal space. J Neurophysiol 38:871-908.

Muir RB, Lemon RN (1983) Corticospinal neurons with a special role in precision grip. Brain Res 261:312-316.

Mulliken GH, Musallam S, Andersen RA (2008) Decoding Trajectories from Posterior Parietal Cortex Ensembles. J Neurosci 28:12913-12926.

Murata A, Fadiga L, Fogassi L, Gallese V, Raos V, Rizzolatti G (1997) Object representation in the ventral premotor cortex (area F5) of the monkey. J Neurophysiol 78:2226-2230.

Murata A, Gallese V, Luppino G, Kaseda M, Sakata H (2000) Selectivity for the shape, size, and orientation of objects for grasping in neurons of monkey parietal area AIP. J Neurophysiol 83:2580-2601.

Musallam S, Corneil BD, Greger B, Scherberger H, Andersen RA (2004) Cognitive control signals for neural prosthetics. Science 305:258-262.

Nakamura H, Kuroda T, Wakita M, Kusunoki M, Kato A, Mikami A, Sakata H, Itoh K (2001) From three-dimensional space vision to prehensile hand movements: The lateral intraparietal area links the area V3A and the anterior intraparietal area in macaques. J Neurosci 21:8174-8187.

National Research Council (2003) Guidelines for the care and use of mammals in neuroscience and behavioral research. Washington, D.C.: National Academies.

Nicolelis MAL, Lebedev MA (2009) Principles of neural ensemble physiology underlying the operation of brain-machine interfaces. Nat. Rev. Neurosci. 10:530-540.

Oram MW, Földiák P, Perrett DI, Sengpiel F (1998) The ‘Ideal Homunculus': decoding neural population signals.Trends in Neurosciences Volume 21, Issue 6, 259-265

Paddock C. (2009) Paralysis Affects More Americans Than Previously Thought. http://www.medicalnewstoday.com/articles/146819.php

Paninski L, Shoham S, Fellows MR, Hatsopoulos NG, Donoghue JP (2004) Superlinear population encoding of dynamic hand trajectory in primary motor cortex. J Neurosci 24:8551-8561.

Panzeri S, Senatore R, Montemurro MA, Petersen RS (2007) Correcting for the sampling bias problem in spike train information measures. J. neurophysiol. 98, 1064–1072.

Polikov VS, Tresco PA, Reichert WM (2005) Response of brain tissue to chronically implanted neural electrodes. J Neurosci Methods 148:1-18.

Polikar R, (2006) Ensemble based systems in decision making. IEEE Circuits and Systems Magazine Volume: 6, Issue: 3: 21-45.

Pouget A, Dayan P, Zemel R (2000) Information processing with population codes", Nature Reviews Neuroscience 1, 125-132.

Quinlan JR (1986) Induction of Decision Trees. Machine Learning 1: 81-106.

Quinlan JR (1993) C4.5: Programs for Machine Learning. San Mateo CA: Morgan Kaufmann.

Quinlan JR (1996) Bagging, Boosting and C4.5. 13th Intl. Conference on Artificial Intelligence: 725-730.

Quiroga RQ, Nadasdy Z, Ben-Shaul Y (2004) Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Comp 16:1661-1687.

Quiroga R. Q. and Panzeri S. (2009) Extracting information from neuronal populations: information theory and decoding approaches. Nature reviews. Neuroscience 10, 530-540.

Rabiner LR (1989) A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE 77 (2): 257–286.

Raos V, Umilta MA, Murata A, Fogassi L, Gallese V (2006) Functional properties of grasping-related neurons in the ventral premotor area F5 of the macaque monkey. J Neurophysiol 95:709-729.

Rizzolatti G, Camarda R, Fogassi L, Gentilucci M, Luppino G, Matelli M (1988) Functional-Organization of Inferior Area-6 in the Macaque Monkey. II. Area F5 and the Control of Distal Movements. Exp Brain Res 71:491-507.

Robertson RG, Rolls ET, Georges-Francois P, Panzeri S. (1999)Head direction cells in the primate presubiculum. Hippocampus 9, 206–219.

Sakata H, Taira M, Kusunoki M, Murata A, Tanaka Y (1997) The TINS Lecture. The parietal association cortex in depth perception and visual control of hand action. Trends Neurosci 20:350-357.

Sakata H, Taira M, Murata A, Mine S (1995) Neural mechanisms of visual guidance of hand action in the parietal cortex of the monkey. Cereb Cortex 5:429-438.

Santhanam G, Ryu SI, Yu BM, Afshar A, Shenoy KV (2006) A high-performance brain-computer interface. Nature 442:195-198.

Schapire RE (1990) The strength of weak learnability. Machine Learning 5: 197-227.

Schapire RE, Freund Y, Bartlett P, Lee WS (1998) Boosting the margin: A new explanation for the effectiveness of voting methods. Annals of Statistics 26: 1651-1686.

Scherberger H (2009) Neural control of motor prostheses. Curr Opin Neurobiol 19:629-633.

Scherberger H, Jarvis MR, Andersen RA (2005) Cortical local field potential encodes movement intentions in the posterior parietal cortex. Neuron 46:347-354.

Schieber MH, Santello M (2004) Hand function: peripheral and central constraints on performance. J Appl Physiol 96:2293-2300.

Schmolesky MT, Wang YC, Hanes DP, Thompson KG, Leutgeb S, Schall JD, Leventhal AG (1998) Signal timing across the macaque visual system. J Neurophysiol 79:3272-3278.

Scholz M, Klinkenberg R (2005) An Ensemble Classifier for Drifting Concepts. In Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams.

Schölkoph B, Tsuda K, Vert JP (2004) Kernel Methods in Computational Biology. Cambridge MA, MIT Press.

Schwartz AB, Cui XT, Weber DJ, Moran DW (2006) Brain-controlled interfaces: Movement restoration with neural prosthetics. Neuron 52:205-220.

Scott SH (2000) One motor cortex, two different views - Reply. Nat Neurosci 3:964-965.

Serruya MD, Hatsopoulos NG, Paninski L, Fellows MR, Donoghue JP (2002) Instant neural control of a movement signal. Nature 416:141-142.

Shadlen MN, Newsome WT (1998) The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. J Neurosci. 18(10):3870-96.

Shannon CE (1948) A mathematical theory of communication. Bell Syst. Tech. J. 27, 379–423.

Shawe-Taylor J, Cristianini N (2004) Kernel Methods for Pattern Analysis. Cambridge, UK: Cambridge Uni. Press.

Shenoy KV, Meeker D, Cao SY, Kureshi SA, Pesaran B, Buneo CA, Batista AR, Mitra PP, Burdick JW, Andersen RA (2003) Neural prosthetic control signals from plan activity. Neuroreport 14:591-596.

Shimazu H, Maier MA, Cerri G, Kirkwood PA, Lemon RN (2004) Macaque ventral premotor cortex exerts powerful facilitation of motor cortex outputs to upper limb motoneurons. J Neurosci 24:1200-1211.

Sill J, Takacs G, Mackey L, Lin D (2009) Feature-Weighted Linear Stacking arXiv:0911.0460

Snoek GJ, IJzerman MJ, Hermens HJ, Maxwell D, Biering-Sorensen F (2004) Survey of the needs of patients with spinal cord injury: impact and priority for improvement in hand function in tetraplegics. Spinal Cord 42:526-532.

Stark E, Abeles M (2007) Predicting movement from multiunit activity. J Neurosci 27:8387-8394.

Subasi E, Townsend B, Scherberger H (2010) In search of more robust decoding algorithms for neural prostheses, a data driven approach. Conf Proc IEEE ENg Med Biol Soc 2010:4172-4175.

Subasi E, Townsend, BR, Scherberger H, (2009): An online boosting approach for hand grasping brain machine interfaces. Chicago, DC: Neuroscience 2009.

Subasi E, Townsend, BR, Scherberger H, (2008): Using Markovian models and maximum likelihood estimation for behavioral state decoding of neural signals. Washington, DC: Neuroscience 2008.

Subasi E, Townsend B, Scherberger H (2007) A software tool for developing and testing online neural decoding algorithms. San Diego, CA: Neuroscience 2007.

Super H, Roelfsema PR (2005) Chronic multiunit recordings in behaving animals: advantages and limitations. Prog Brain Res 147:263-282.

Taira M, Mine S, Georgopoulos AP, Murata A, Sakata H (1990) Parietal Cortex Neurons of the Monkey Related to the Visual Guidance of Hand Movement. Exp Brain Res 83:29-36.

Taylor DM, Tillery SIH, Schwartz AB (2002) Direct cortical control of 3D neuroprosthetic devices. Science 296:1829-1832.

Thorpe S, Fize D, Marlot C (1996) Speed of processing in the human visual system. Nature 381:520-522.

Townsend B, Subasi E, Scherberger H (2011). Grasp Movement Decoding from Premotor and Parietal Cortex. Journal of Neuroscience. October 5, 2011 31(40):14386 –14398.

Townsend B, Subasi E, Scherberger H (2008) Real time decoding of hand grasping signals from macaque premotor and paritetal cortex.13th Annual Conference of IFES Society “From Movement to Mind” Freiburg, Germany: Biomedical Engineering, 53 supp. 1.

Townsend BR, Paninski L, Lemon RN (2006) Linear encoding of muscle activity in primary motor cortex and cerebellum. J Neurophysiol 96:2578-2592.

Tsutsui KI, Jiang M, Yara K, Sakata H, Taira M (2001) Integration of perspective and disparity cues in surface-orientation-selective neurons of area CIP. J Neurophysiol 86:2856-2867.

Tsutsui KI, Sakata H, Naganuma T, Taira M (2002) Neural correlates for perception of 3D surface orientation from texture gradient. Science 298:409-412.

Umilta MA, Brochier T, Spinks RL, Lemon RN (2007) Simultaneous recording of macaque premotor and primary motor cortex neuronal populations reveals different functional contributions to visuomotor grasp. J Neurophysiol 98:488-501.

Vapnik V. (1995) The nature of statistical learning theory. New York, Springer.

Vargas-Irwin CE, Shakhnarovich G, Yadollahpour P, Mislow JM, Black MJ, Donoghue JP (2010) Decoding complete reach and grasp actions from local primary motor cortex populations. J Neurosci 30: 9659-9669.

Velliste M, Perel S, Spalding MC, Whitford AS, Schwartz AB (2008) Cortical control of a prosthetic arm for self-feeding. Nature 453:1098-1101.

Ventura V (2008) Spike train decoding without spike sorting. Neural Comp 20:923-963.

Wessberg J, Stambaugh CR, Kralik JD, Beck PD, Laubach M, Chapin JK, Kim J, Biggs J, Srinivasan MA, Nicolelis MAL (2000) Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature 408:361-365.

Wolpert DH (1992) Stacked generalization. Neural Networks, Volume 5, Issue 2, 241-259.

Wolpaw JR, McFarland DJ (2004) Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. P Natl Acad Sci USA 101:17849-17854.

Yanai Y, Adamit N, Harel R, Israel Z, Prut Y (2007) Connected corticospinal sites show enhanced tuning similarity at the onset of voluntary action. J Neurosci 27:12349-12357.

Zeitler M, Fries P, Gielen S (2006) Assessing neuronal coherence with single-unit, multi-unit, and local field potentials. Neural Comp 18:2256-2281.

Appendix

A. A sample Rapidminer study description in xml format

The following xml code is a Rapidminer file which describes a cross validation setup by utilizing SVMs and Naive Bayesian decoders. It is provided for illustrative purposes on how the analysis is carried out in Chapter 5.

```
<?xml version="1.0" encoding="windows-1252"?>
<process version="4.4">

 <operator name="Root" class="Process" expanded="yes">
    <parameter key="logverbosity"    value="init"/>
    <parameter key="random_seed" value="2001"/>
    <parameter key="encoding"        value="SYSTEM"/>
    <operator name="FileIterator" class="FileIterator" expanded="yes">
       <parameter key="directory"     value="/Users/erk/work/INI/DATA/analysis/union/all"/>
       <parameter key="filter"           value=".*aml"/>
       <parameter key="file_name_macro"          value="file_name"/>
       <parameter key="file_path_macro"          value="file_path"/>
       <parameter key="parent_path_macro"       value="parent_path"/>
       <parameter key="recursive"    value="false"/>
       <parameter key="iterate_over_files"          value="true"/>
       <parameter key="iterate_over_subdirs"      value="false"/>
       <operator name="ExampleSource" class="ExampleSource">
          <parameter key="attributes" value="%{file_path}"/>
          <parameter key="sample_ratio"             value="1.0"/>
          <parameter key="sample_size"              value="-1"/>
          <parameter key="permutate"                value="false"/>
          <parameter key="decimal_point_character"             value="."/>
          <parameter key="column_separators"    value=",\s*|;\s*|\s+"/>
          <parameter key="use_comment_characters"           value="true"/>
          <parameter key="comment_chars"        value="#"/>
          <parameter key="use_quotes"               value="true"/>
          <parameter key="quote_character"        value="&amp;quot;"/>
          <parameter key="quoting_escape_character"          value="\"/>
          <parameter key="trim_lines"value="false"/>
          <parameter key="datamanagement"      value="double_array"/>
          <parameter key="local_random_seed"   value="-1"/>
       </operator>
       <operator name="IOStorer" class="IOStorer" breakpoints="after">
          <parameter key="name"      value="exm"/>
          <parameter key="io_object" value="ExampleSet"/>
          <parameter key="store_which"              value="1"/>
          <parameter key="remove_from_process"              value="true"/>
       </operator>
       <operator name="ParameterIteration" class="ParameterIteration" expanded="yes">
          <list key="parameters">
            <parameter key="OperatorSelector.select_which"   value="12"/>
          </list>
          <parameter key="synchronize"              value="false"/>
          <parameter key="keep_output"              value="false"/>
          <operator name="OperatorSelector" class="OperatorSelector" expanded="yes">
             <parameter key="select_which"         value="13"/>
             <operator name="NaiveBayes_Poisson" class="OperatorChain" activated="no" expanded="no">
                <operator name="IORetriever" class="IORetriever">
                   <parameter key="name"             value="exm"/>
                   <parameter key="io_object"        value="ExampleSet"/>
                   <parameter key="remove_from_store"          value="false"/>
                </operator>
                <operator name="SimpleValidation" class="SimpleValidation" expanded="no">
```

```
      <parameter key="keep_example_set"            value="false"/>
      <parameter key="create_complete_model"    value="false"/>
      <parameter key="split_ratio"       value="0.5"/>
      <parameter key="sampling_type"value="linear sampling"/>
      <parameter key="local_random_seed"           value="-1"/>
      <operator name="FS-Learner" class="OperatorChain" expanded="no">
        <operator name="AnovaMatrixWeighting" class="AnovaMatrixWeighting">
          <parameter key="normalize_weights"     value="true"/>
          <parameter key="significance_level"      value="0.05"/>
        </operator>
        <operator name="AttributeWeightSelection" class="AttributeWeightSelection">
          <parameter key="keep_attribute_weights"               value="false"/>
          <parameter key="weight"    value="1.0"/>
          <parameter key="weight_relation"          value="greater equals"/>
          <parameter key="k"            value="10"/>
          <parameter key="p"            value="0.5"/>
          <parameter key="deselect_unknown"     value="true"/>
          <parameter key="use_absolute_weights"                value="true"/>
        </operator>
        <operator name="NaiveBayes" class="NaiveBayes">
          <parameter key="keep_example_set"     value="false"/>
          <parameter key="laplace_correction"      value="true"/>
          <parameter key="poisson_distribution"    value="true"/>
          <parameter key="homogeneous_priors" value="true"/>
          <parameter key="log_likelihoods"          value="true"/>
        </operator>
      </operator>
      <operator name="PerformanceEvaluation" class="OperatorChain" expanded="no">
        <operator name="ModelApplier" class="ModelApplier">
          <parameter key="keep_model"              value="false"/>
          <list key="application_parameters">
          </list>
          <parameter key="create_view"              value="false"/>
        </operator>
        <operator name="Performance" class="Performance">
          <parameter key="keep_example_set"     value="false"/>
          <parameter key="use_example_weights"                value="true"/>
        </operator>
      </operator>
    </operator>
    <operator name="_01-NaiveBayes-Poisson" class="PerformanceWriter">
      <parameter key="performance_file"              value="%{parent_path}/results/%{file_name}%{n}.res"/>
    </operator>
    <operator name="Cleaning" class="OperatorChain" expanded="no">
      <operator name="IOConsumer" class="IOConsumer">
        <parameter key="io_object"     value="PerformanceVector"/>
        <parameter key="deletion_type"               value="delete_all"/>
        <parameter key="delete_which"                value="1"/>
        <parameter key="except"       value="1"/>
      </operator>
      <operator name="MemoryCleanUp" class="MemoryCleanUp">
      </operator>
    </operator>
  </operator>
  <operator name="NaiveBayes_WekaMultiNomial" class="OperatorChain" expanded="no">
    <operator name="IORetriever (2)" class="IORetriever">
      <parameter key="name"            value="exm"/>
      <parameter key="io_object"       value="ExampleSet"/>
      <parameter key="remove_from_store"           value="false"/>
    </operator>
    <operator name="SimpleValidation (10)" class="SimpleValidation" expanded="yes">
      <parameter key="keep_example_set"            value="false"/>
      <parameter key="create_complete_model"    value="false"/>
      <parameter key="split_ratio"       value="0.5"/>
      <parameter key="sampling_type"value="linear sampling"/>
      <parameter key="local_random_seed"           value="-1"/>
      <operator name="FS-Learner (10)" class="OperatorChain" expanded="yes">
        <operator name="AnovaMatrixWeighting (10)" class="AnovaMatrixWeighting">
          <parameter key="normalize_weights"     value="true"/>
          <parameter key="significance_level"      value="0.05"/>
        </operator>
        <operator name="AttributeWeightSelection (10)" class="AttributeWeightSelection">
          <parameter key="keep_attribute_weights"               value="false"/>
```

```
            <parameter key="weight"    value="1.0"/>
            <parameter key="weight_relation"          value="greater equals"/>
            <parameter key="k"          value="10"/>
            <parameter key="p"          value="0.5"/>
            <parameter key="deselect_unknown"    value="true"/>
            <parameter key="use_absolute_weights"                value="true"/>
          </operator>
          <operator name="W-NaiveBayesMultinomial" class="W-NaiveBayesMultinomial">
            <parameter key="keep_example_set"    value="false"/>
            <parameter key="D"          value="false"/>
          </operator>
        </operator>
        <operator name="PerformanceEvaluation (10)" class="OperatorChain" expanded="yes">
          <operator name="ModelApplier (10)" class="ModelApplier">
            <parameter key="keep_model"             value="false"/>
            <list key="application_parameters">
            </list>
            <parameter key="create_view"             value="false"/>
          </operator>
          <operator name="Performance (10)" class="Performance">
            <parameter key="keep_example_set"    value="false"/>
            <parameter key="use_example_weights"                value="true"/>
          </operator>
        </operator>
      </operator>
      <operator name="_02-NaiveBayes-WekaMultiNomial" class="PerformanceWriter">
        <parameter key="performance_file"             value="%{parent_path}/results/%{file_name}%{n}.res"/>
      </operator>
      <operator name="Cleaning (2)" class="OperatorChain" expanded="yes">
        <operator name="IOConsumer (2)" class="IOConsumer">
          <parameter key="io_object"    value="PerformanceVector"/>
          <parameter key="deletion_type"                value="delete_all"/>
          <parameter key="delete_which"                value="1"/>
          <parameter key="except"        value="1"/>
        </operator>
        <operator name="MemoryCleanUp (2)" class="MemoryCleanUp">
        </operator>
      </operator>
    </operator>
    <operator name="RbfSVM" class="OperatorChain" activated="no" expanded="no">
      <operator name="IORetriever (7)" class="IORetriever">
        <parameter key="name"          value="exm"/>
        <parameter key="io_object"        value="ExampleSet"/>
        <parameter key="remove_from_store"          value="false"/>
      </operator>
      <operator name="Normalization (2)" class="Normalization">
        <parameter key="return_preprocessing_model"              value="false"/>
        <parameter key="create_view"    value="false"/>
        <parameter key="method"          value="Z-Transformation"/>
        <parameter key="min"          value="0.0"/>
        <parameter key="max"          value="1.0"/>
      </operator>
      <operator name="SimpleValidation (9)" class="SimpleValidation" expanded="yes">
        <parameter key="keep_example_set"          value="false"/>
        <parameter key="create_complete_model"    value="false"/>
        <parameter key="split_ratio"       value="0.5"/>
        <parameter key="sampling_type"value="linear sampling"/>
        <parameter key="local_random_seed"          value="-1"/>
        <operator name="FS-Learner (9)" class="OperatorChain" expanded="yes">
          <operator name="AnovaMatrixWeighting (9)" class="AnovaMatrixWeighting">
            <parameter key="normalize_weights"    value="true"/>
            <parameter key="significance_level"      value="0.05"/>
          </operator>
          <operator name="AttributeWeightSelection (9)" class="AttributeWeightSelection">
            <parameter key="keep_attribute_weights"              value="false"/>
            <parameter key="weight"    value="1.0"/>
            <parameter key="weight_relation"          value="greater equals"/>
            <parameter key="k"          value="10"/>
            <parameter key="p"          value="0.5"/>
            <parameter key="deselect_unknown"    value="true"/>
            <parameter key="use_absolute_weights"                value="true"/>
          </operator>
          <operator name="ParameterOptimization" class="OperatorChain" expanded="no">
```

```
    <operator name="IOMultiplier" class="IOMultiplier">
      <parameter key="number_of_copies" value="1"/>
      <parameter key="io_object"        value="ExampleSet"/>
      <parameter key="multiply_type"    value="multiply_one"/>
      <parameter key="multiply_which"   value="1"/>
    </operator>
    <operator name="GridParameterOptimization" class="GridParameterOptimization" expanded="yes">
      <list key="parameters">
        <parameter key="RbfSVM_Evl.C"     value="0.01,0.1,10,100,1000,10000,0"/>
        <parameter key="RbfSVM_Evl.gamma"         value="0.0001,0.001,0.01,0.1,1,10,100,1000,0"/>
      </list>
      <operator name="XValidation" class="XValidation" expanded="no">
        <parameter key="keep_example_set"         value="false"/>
        <parameter key="create_complete_model"    value="false"/>
        <parameter key="average_performances_only" value="true"/>
        <parameter key="leave_one_out"    value="false"/>
        <parameter key="number_of_validations"    value="5"/>
        <parameter key="sampling_type"    value="stratified sampling"/>
        <parameter key="local_random_seed"        value="-1"/>
        <operator name="RbfSVM_Evl" class="LibSVMLearner">
          <parameter key="keep_example_set"       value="true"/>
          <parameter key="svm_type"       value="C-SVC"/>
          <parameter key="kernel_type"    value="rbf"/>
          <parameter key="degree"         value="3"/>
          <parameter key="gamma"          value="0"/>
          <parameter key="coef0"          value="0.0"/>
          <parameter key="C" value="0"/>
          <parameter key="nu"             value="0.5"/>
          <parameter key="cache_size"     value="80"/>
          <parameter key="epsilon"        value="0.0010"/>
          <parameter key="p" value="0.1"/>
          <list key="class_weights">
          </list>
          <parameter key="shrinking"      value="true"/>
          <parameter key="calculate_confidences"    value="false"/>
          <parameter key="confidence_for_multiclass" value="true"/>
        </operator>
        <operator name="OperatorChain" class="OperatorChain" expanded="no">
          <operator name="ModelApplier (13)" class="ModelApplier">
            <parameter key="keep_model"             value="false"/>
            <list key="application_parameters">
            </list>
            <parameter key="create_view"            value="false"/>
          </operator>
          <operator name="Performance (13)" class="Performance">
            <parameter key="keep_example_set"       value="false"/>
            <parameter key="use_example_weights"    value="true"/>
          </operator>
        </operator>
      </operator>
    </operator>
    <operator name="ParameterSetWriter" class="ParameterSetWriter" breakpoints="after" activated="no">
      <parameter key="parameter_file"       value="%{parent_path}/results/%{file_name}%{n}.par"/>
    </operator>
    <operator name="ParameterSetter" class="ParameterSetter">
      <list key="name_map">
        <parameter key="RbfSVM_Evl"       value="RbfSVM_Opt"/>
      </list>
    </operator>
  </operator>
  <operator name="RbfSVM_Opt" class="LibSVMLearner">
    <parameter key="keep_example_set"    value="false"/>
    <parameter key="svm_type"            value="C-SVC"/>
    <parameter key="kernel_type"         value="rbf"/>
    <parameter key="degree"    value="3"/>
    <parameter key="gamma"     value="0.001"/>
    <parameter key="coef0"     value="0.0"/>
    <parameter key="C"         value="100"/>
    <parameter key="nu"        value="0.5"/>
    <parameter key="cache_size"          value="80"/>
    <parameter key="epsilon"   value="0.0010"/>
    <parameter key="p"         value="0.1"/>
    <list key="class_weights">
```

```
            </list>
            <parameter key="shrinking" value="true"/>
            <parameter key="calculate_confidences"          value="false"/>
            <parameter key="confidence_for_multiclass"      value="true"/>
          </operator>
        </operator>
        <operator name="PerformanceEvaluation (9)" class="OperatorChain" expanded="no">
          <operator name="ModelWriter" class="ModelWriter" breakpoints="after" activated="no">
            <parameter key="model_file"             value="%{parent_path}/results/%{file_name}_rbfmodel.mod"/>
            <parameter key="overwrite_existing_file"        value="true"/>
            <parameter key="output_type"            value="XML Zipped"/>
          </operator>
          <operator name="ModelApplier (9)" class="ModelApplier">
            <parameter key="keep_model"             value="false"/>
            <list key="application_parameters">
            </list>
            <parameter key="create_view"            value="false"/>
          </operator>
          <operator name="Performance (9)" class="Performance">
            <parameter key="keep_example_set"   value="false"/>
            <parameter key="use_example_weights"            value="true"/>
          </operator>
        </operator>
      </operator>
      <operator name="_11-RbfSVM" class="PerformanceWriter">
        <parameter key="performance_file"           value="%{parent_path}/results/%{file_name}%{n}.res"/>
      </operator>
      <operator name="Cleaning (7)" class="OperatorChain" expanded="no">
        <operator name="IOConsumer (7)" class="IOConsumer">
          <parameter key="io_object"    value="PerformanceVector"/>
          <parameter key="deletion_type"            value="delete_all"/>
          <parameter key="delete_which"             value="1"/>
          <parameter key="except"       value="1"/>
        </operator>
        <operator name="MemoryCleanUp (7)" class="MemoryCleanUp">
        </operator>
      </operator>
    </operator>
    <operator name="ClassifiersVoting_DecisionTree" class="OperatorChain" activated="no" expanded="no">
      <operator name="IORetriever (11)" class="IORetriever">
        <parameter key="name"           value="exm"/>
        <parameter key="io_object"      value="ExampleSet"/>
        <parameter key="remove_from_store"          value="false"/>
      </operator>
      <operator name="SimpleValidation (12)" class="SimpleValidation" expanded="no">
        <parameter key="keep_example_set"           value="false"/>
        <parameter key="create_complete_model"  value="false"/>
        <parameter key="split_ratio"    value="0.5"/>
        <parameter key="sampling_type"value="linear sampling"/>
        <parameter key="local_random_seed"          value="-1"/>
        <operator name="FS-Learner (12)" class="OperatorChain" expanded="yes">
          <operator name="AnovaMatrixWeighting (12)" class="AnovaMatrixWeighting">
            <parameter key="normalize_weights"  value="true"/>
            <parameter key="significance_level"     value="0.05"/>
          </operator>
          <operator name="AttributeWeightSelection (12)" class="AttributeWeightSelection">
            <parameter key="keep_attribute_weights"             value="false"/>
            <parameter key="weight"     value="1.0"/>
            <parameter key="weight_relation"        value="greater equals"/>
            <parameter key="k"          value="10"/>
            <parameter key="p"          value="0.5"/>
            <parameter key="deselect_unknown"   value="true"/>
            <parameter key="use_absolute_weights"           value="true"/>
          </operator>
          <operator name="Vote" class="Vote" expanded="no">
            <parameter key="keep_example_set"   value="false"/>
            <operator name="ID3Numerical" class="ID3Numerical">
              <parameter key="keep_example_set" value="false"/>
              <parameter key="criterion"            value="information_gain"/>
              <parameter key="minimal_size_for_split"           value="4"/>
              <parameter key="minimal_leaf_size"  value="2"/>
              <parameter key="minimal_gain"       value="0.1"/>
            </operator>
```

```
<operator name="NaiveBayes (3)" class="NaiveBayes">
   <parameter key="keep_example_set" value="false"/>
   <parameter key="laplace_correction"  value="true"/>
   <parameter key="poisson_distribution"                value="true"/>
   <parameter key="homogeneous_priors"                  value="true"/>
   <parameter key="log_likelihoods"       value="true"/>
</operator>
<operator name="W-NaiveBayesMultinomial (2)" class="W-NaiveBayesMultinomial">
   <parameter key="keep_example_set" value="false"/>
   <parameter key="D"        value="false"/>
</operator>
<operator name="RbfSVM (2)" class="OperatorChain" expanded="yes">
   <operator name="OptimizedRbfSVM" class="OperatorChain" breakpoints="after" activated="no" expanded="yes">
      <operator name="IOMultiplier (3)" class="IOMultiplier">
         <parameter key="number_of_copies"           value="1"/>
         <parameter key="io_object"        value="ExampleSet"/>
         <parameter key="multiply_type"  value="multiply_one"/>
         <parameter key="multiply_which"                value="1"/>
      </operator>
      <operator name="GridParameterOptimization (3)" class="GridParameterOptimization" expanded="yes">
         <list key="parameters">
           <parameter key="RbfSVM_Evl.C"              value="0.01,0.1,10,100,1000,10000,0"/>
           <parameter key="RbfSVM_Evl.gamma"       value="0.0001,0.001,0.01,0.1,1,10,100,1000,0"/>
         </list>
         <operator name="XValidation (3)" class="XValidation" expanded="yes">
            <parameter key="keep_example_set"        value="false"/>
            <parameter key="create_complete_model"value="false"/>
            <parameter key="average_performances_only"          value="true"/>
            <parameter key="leave_one_out"              value="false"/>
            <parameter key="number_of_validations"   value="5"/>
            <parameter key="sampling_type"              value="stratified sampling"/>
            <parameter key="local_random_seed"       value="-1"/>
            <operator name="RbfSVM_Evl (2)" class="LibSVMLearner">
               <parameter key="keep_example_set"    value="true"/>
               <parameter key="svm_type"                value="C-SVC"/>
               <parameter key="kernel_type"             value="rbf"/>
               <parameter key="degree"    value="3"/>
               <parameter key="gamma"   value="0"/>
               <parameter key="coef0"      value="0.0"/>
               <parameter key="C"          value="0"/>
               <parameter key="nu"         value="0.5"/>
               <parameter key="cache_size"              value="80"/>
               <parameter key="epsilon"    value="0.0010"/>
               <parameter key="p"          value="0.1"/>
               <list key="class_weights">
               </list>
               <parameter key="shrinking" value="true"/>
               <parameter key="calculate_confidences"                value="false"/>
               <parameter key="confidence_for_multiclass"          value="true"/>
            </operator>
            <operator name="OperatorChain (3)" class="OperatorChain" expanded="no">
               <operator name="ModelApplier (15)" class="ModelApplier">
                  <parameter key="keep_model"          value="false"/>
                  <list key="application_parameters">
                  </list>
                  <parameter key="create_view"          value="false"/>
               </operator>
               <operator name="Performance (15)" class="Performance">
                  <parameter key="keep_example_set" value="false"/>
                  <parameter key="use_example_weights"             value="true"/>
               </operator>
            </operator>
         </operator>
      </operator>
      <operator name="ParameterSetWriter (3)" class="ParameterSetWriter" breakpoints="after" activated="no">
         <parameter key="parameter_file"                value="%{parent_path}/results/%{file_name}%{n}.par"/>
      </operator>
      <operator name="ParameterSetter (3)" class="ParameterSetter">
         <list key="name_map">
           <parameter key="RbfSVM_Evl" value="RbfSVM_Opt"/>
         </list>
      </operator>
      <operator name="RbfSVM_Opt (2)" class="LibSVMLearner">
```

```
                        <parameter key="keep_example_set"           value="false"/>
                        <parameter key="svm_type"       value="C-SVC"/>
                        <parameter key="kernel_type"    value="rbf"/>
                        <parameter key="degree"         value="3"/>
                        <parameter key="gamma"          value="0.1"/>
                        <parameter key="coef0"          value="0.0"/>
                        <parameter key="C" value="0"/>
                        <parameter key="nu"             value="0.5"/>
                        <parameter key="cache_size"     value="80"/>
                        <parameter key="epsilon"        value="0.0010"/>
                        <parameter key="p" value="0.1"/>
                        <list key="class_weights">
                        </list>
                        <parameter key="shrinking"      value="true"/>
                        <parameter key="calculate_confidences"      value="false"/>
                        <parameter key="confidence_for_multiclass" value="true"/>
                      </operator>
                    </operator>
                    <operator name="ModelLoader" class="ModelLoader">
                      <parameter key="model_file"       value="%{parent_path}/results/models/%{file_name}_rbfmodel.mod"/>
                    </operator>
                  </operator>
                </operator>
                <operator name="PerformanceEvaluation (12)" class="OperatorChain" expanded="no">
                  <operator name="ModelApplier (12)" class="ModelApplier">
                    <parameter key="keep_model"             value="false"/>
                    <list key="application_parameters">
                    </list>
                    <parameter key="create_view"            value="false"/>
                  </operator>
                  <operator name="Performance (12)" class="Performance">
                    <parameter key="keep_example_set"   value="false"/>
                    <parameter key="use_example_weights"                value="true"/>
                  </operator>
                </operator>
              </operator>
              <operator name="_12-Stacking-Classifiers-DecisionTree" class="PerformanceWriter">
                <parameter key="performance_file"           value="%{parent_path}/results/%{file_name}%{n}.res"/>
              </operator>
              <operator name="Cleaning (11)" class="OperatorChain" expanded="no">
                <operator name="IOConsumer (11)" class="IOConsumer">
                  <parameter key="io_object"    value="PerformanceVector"/>
                  <parameter key="deletion_type"            value="delete_all"/>
                  <parameter key="delete_which"             value="1"/>
                  <parameter key="except"       value="1"/>
                </operator>
                <operator name="MemoryCleanUp (11)" class="MemoryCleanUp">
                </operator>
              </operator>
            </operator>
          </operator>
        </operator>
      </operator>
      <operator name="CommandLineOperator" class="CommandLineOperator" activated="no">
        <parameter key="command"    value="/Users/erk/Documents/RapidMiner/IniAnalysisParser.py
/Users/erk/work/INI/DATA/analysis/union/all/results"/>
        <parameter key="log_stdout"   value="true"/>
        <parameter key="log_stderr"   value="true"/>
      </operator>
    </operator>

</process>
```

B. Class diagramms for Decoder and Simulator software

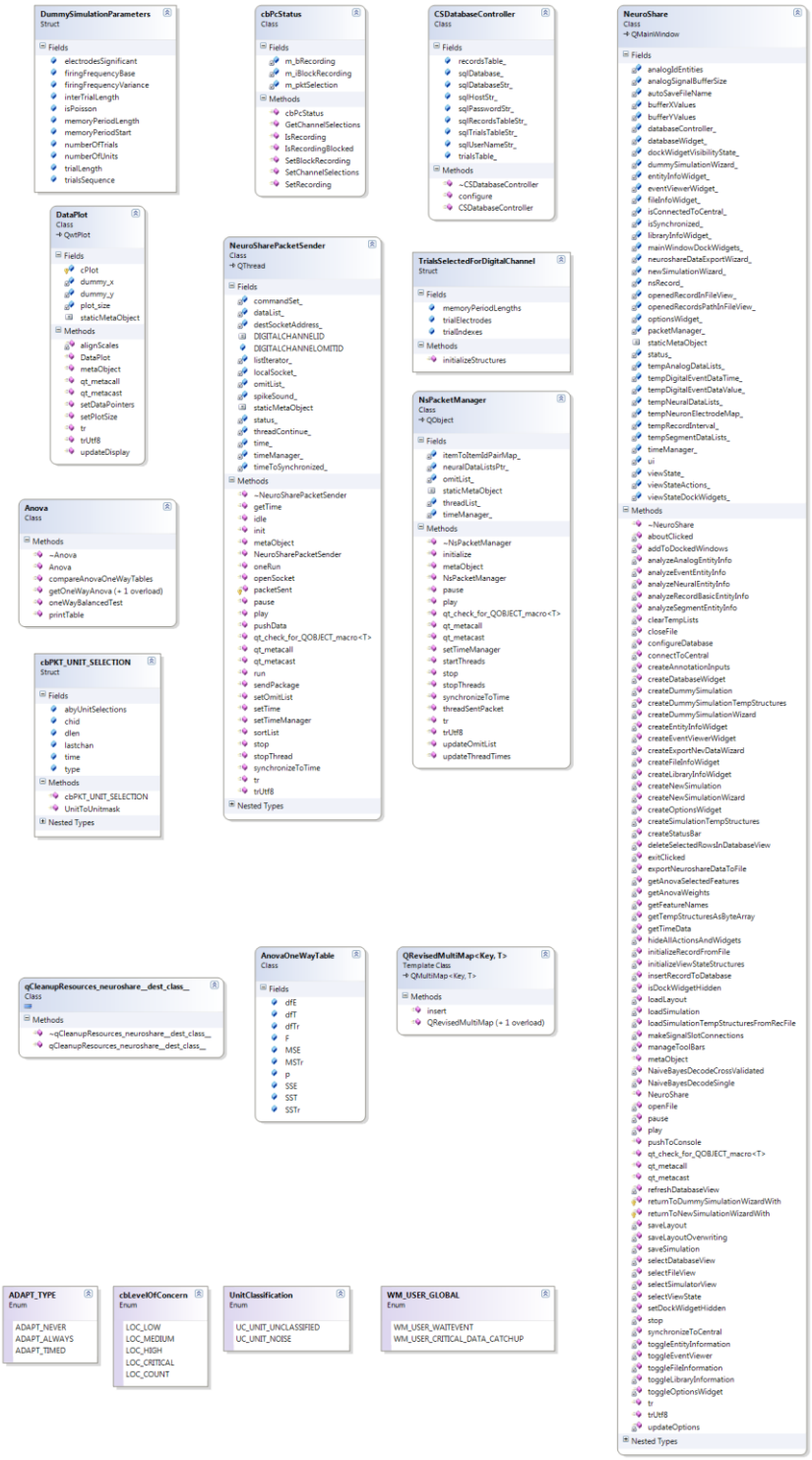

Figure Appendix-B.1: Expanded class diagram for Simulator Software, part 1. The attribute and method names for major classes and data structures of Simulator software.

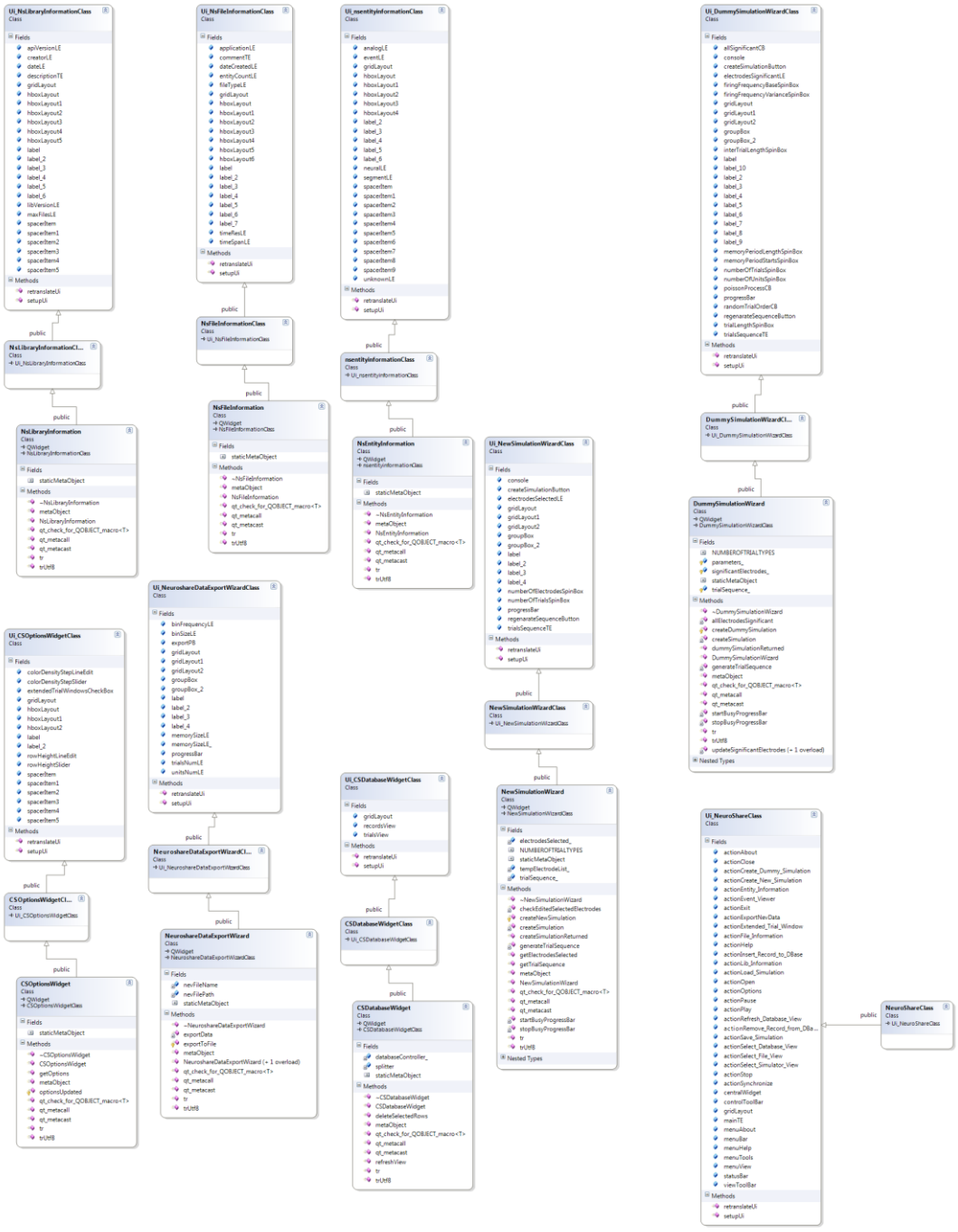

Figure Appendix-B.2: Expanded class diagram for Simulator Software, part 2. The attribute and method names for user interface classes of Simulator software.

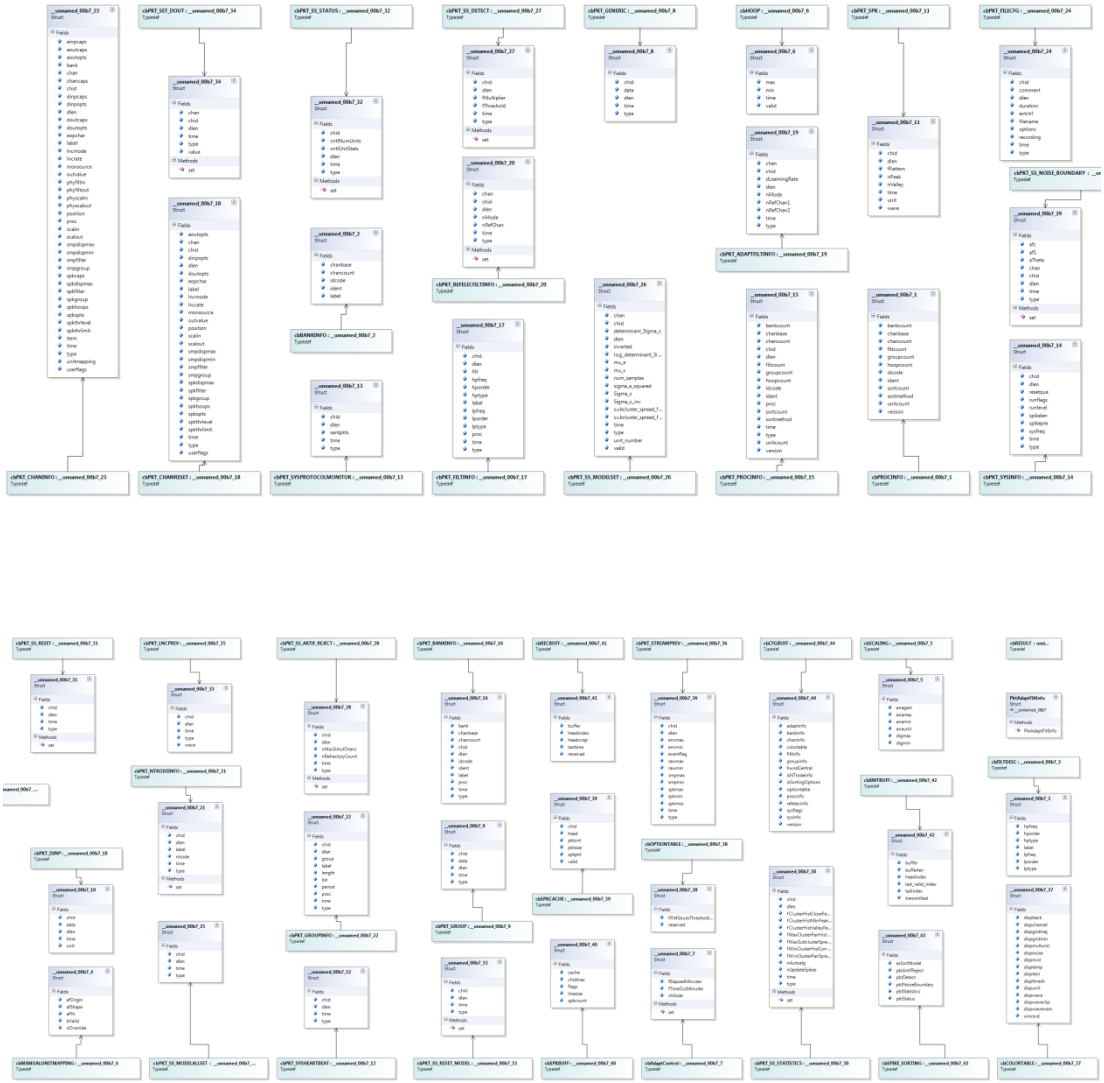

Figure Appendix-B.3: Expanded class diagram for Simulator Software, part 3. The attribute and method names for Neuroshare API (the native api we used for parsing the industry standard streaming file format for neural recordings) wrapper classes of Simulator software.

Figure Appendix-B.4: Expanded class diagram for Decoder Software, part 1. The attribute and method names for user interface and some utility classes and data structures of Decoder software.

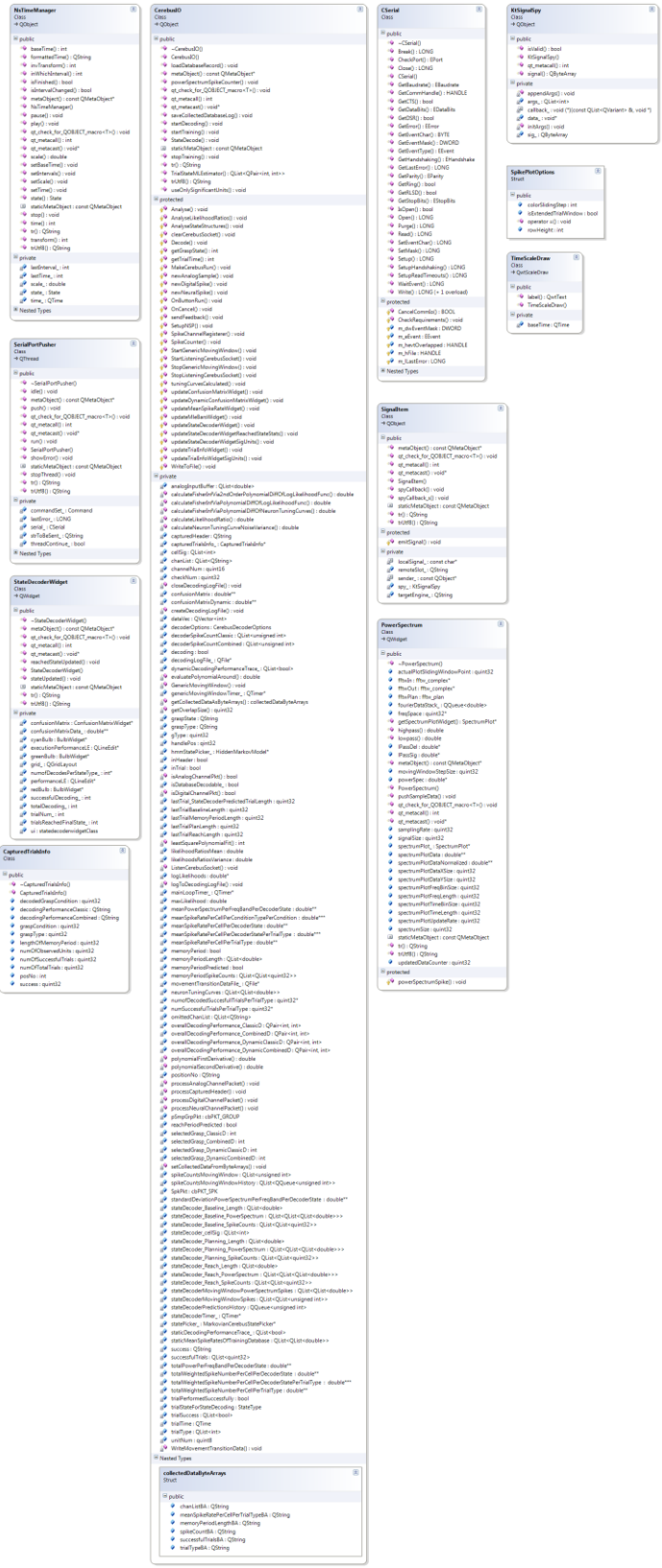

Figure Appendix-B.5: Expanded class diagram for Decoder Software, part 2.
The attribute and method names for major control classes.

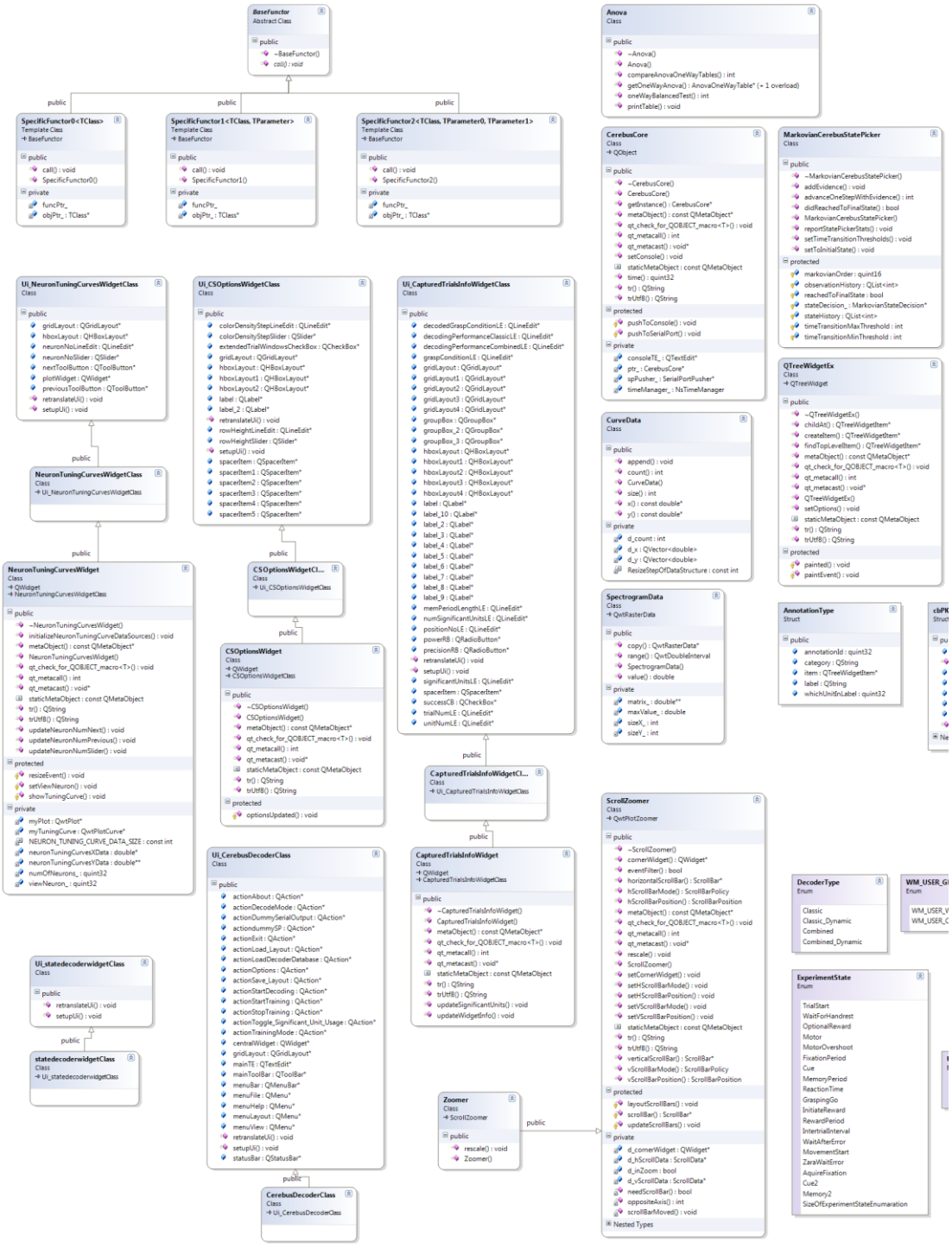

Figure Appendix-B.6: Expanded class diagram for Decoder Software, part 3
The remaining user interface and control classes and data structures for Decoder.

Curriculum vitae

Personal

Name:	Erk Subasi
Birth Day:	29. May, 1980
Nationality:	Turkey

Education

2006-2011	Ph.D. in Institute of Neuroinformatics, ETH and University of Zürich
2003-2006	M.Sc. in Computational Science & Engineering Koç Uni., Istanbul
1998-2003	B.Sc. in Electrical & Electronics Engineering, METU, Ankara
1991-1998	Bornova Anatolian High School, İzmir

Awards & Achievements

PhD scholarship for 3 years from Zurich Neurowissenschaft Zentrum (2007-2009).

Full scholarship during M.Sc. in Koc University (2003-2005).

Publications

Journal Papers:

Townsend B, Subasi E, Scherberger H (2011). Grasp Movement Decoding from Premotor and Parietal Cortex. Journal of Neuroscience. October 5, 2011 31(40):14386 –14398.

Subasi E, Basdogan C (2008) A New Haptic Interaction and Visualization Approach for Rigid Molecular Docking in Virtual Environments. Presence: Teleoperators & Virtual Environments February 2008, Vol. 17, No. 1: 73–90

Conference Papers (peer reviewed):

Subasi E, Townsend, BR, Scherberger H, (2010): In Search of More Robust Decoding Algorithms for Neural Prostheses, a Data Driven Approach**.** In:

Proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'10), Sep 1-4, 2010, Buenos Aires, Argentina.

Townsend B, Subasi E, Scherberger H (2008): Real time decoding of hand grasping signals from macaque premotor and parietal cortex. In: Stieglitz T. & Schuettler M.: Proceedings of the 13th Annual Conference of the International Functional Electrical Stimulation Society "From Movement to Mind", Sep 21-25, 2008, Freiburg, Germany. Printed in: Biomedical Engineering 53, Supp. 1, 203-205

Subasi E, Basdogan C (2006) A New Approach to Molecular Docking in Virtual Environments with Haptic Feedback. Proceedings of EuroHaptics Conference, 2006 , Paris.

Conference Abstracts & Presentations:

Subasi E, Townsend, BR, Scherberger H, (2009): An online boosting approach for hand grasping brain machine interfaces. Chicago, DC: Neuroscience 2009.

Subasi E, Townsend, BR, Scherberger H, (2008): Using Markovian models and maximum likelihood estimation for behavioral state decoding of neural signals. Washington, DC: Neuroscience 2008.

Subasi E, Townsend B, Scherberger H (2007) A software tool for developing and testing online neural decoding algorithms. San Diego, CA: Neuroscience 2007.

Townsend B, Lehmann S, Subasi E, Scherberger H (2007) Decoding hand grasping signals from primate premotor and parietal cortex. San Diego, CA: Neuroscience 2007.