Statistical Models and Algorithms for Studying Hand and Finger Kinematics and their Neural Mechanisms

Lucia Castellanos

August 2013
CMU-ML-13-108

School of Computer Science
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee
Robert E. Kass, chair
Tom M. Mitchell
Byron M. Yu
Andrew B. Schwartz (University of Pittsburgh)

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

This research was sponsored by grants RO1 MH064537, R90 DA023426. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Variance Decomposition, Multivariate Gaussian Process Factor Analysis, Laplace Gaussian Filter, Functional Data Alignment, Encoding, Decoding.

Abstract

The primate hand, a biomechanical structure with over twenty kinematic degrees of freedom, has an elaborate anatomical architecture. Although the hand requires complex, coordinated neural control, it endows its owner with an astonishing range of dexterous finger movements. Despite a century of research, however, the neural mechanisms that enable finger and grasping movements in primates are largely unknown.

In this thesis, we investigate statistical models of finger movement that can provide insights into the mechanics of the hand, and that can have applications in neural-motor prostheses, enabling people with limb loss to regain natural function of the hands.

There are many challenges associated with (1) the understanding and modeling of the kinematics of fingers, and (2) the mapping of intracortical neural recordings into motor commands that can be used to control a Brain-Machine Interface. These challenges include: potential nonlinearities; confounded sources of variation in experimental datasets; and dealing with high degrees of kinematic freedom.

In this work we analyze kinematic and neural datasets from repeated-trial experiments of hand motion, with the following contributions:

- We identified static, nonlinear, low-dimensional representations of grasping finger motion, with accompanying evidence that these nonlinear representations are better than linear representations at predicting the type of object being grasped over the course of a reach-to-grasp movement. In addition, we show evidence of better encoding of these nonlinear (versus linear) representations in the firing of some neurons collected from the primary motor cortex of rhesus monkeys.
- A functional alignment of grasping trajectories, based on total kinetic energy, as a strategy to account for temporal variation and to exploit a repeated-trial experiment structure.
- An interpretable model for extracting dynamic synergies of finger motion, based on Gaussian Processes, that decomposes and reduces the dimensionality of variance in the dataset. We derive efficient algorithms for parameter estimation, show accurate reconstruction of grasping trajectories, and illustrate the interpretation of the model parameters.
- Sound evidence of single-neuron decoding of interpretable grasping events, plus insights about the amount of grasping information extractable from just a single neuron.
- The Laplace Gaussian Filter (LGF), a deterministic approximation to the posterior mean that is more accurate than Monte Carlo approximations for the same computational cost, and that in an off-line decoding task is more accurate than the standard Population Vector Algorithm.

To Jon Huang, without whose faith, encouragement and support
the completion of this journey would have been unthinkable.

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the United States there are approximately 1.7 million people living with limb loss whose quality of life can be improved by providing them with prosthetic devices controlled solely by neural signals. However, to create effective neural prostheses that can mirror the capabilities of human hands, it is necessary to understand both the mechanics of the fingers and their neural basis. The goals of this thesis are to develop and use statistical machine learning methods which provide insight into finger kinematics and to develop algorithms that allow for fast and accurate neural decoding of continuous motor variables. In particular, we address questions such as: how to effectively summarize and model finger kinematics during reach-to-grasp movements, how to decouple the different sources of variability across repeated trials of the same movement and how to obtain interpretable summaries that allow for accurate reconstruction of movement. We also investigate potential relationships between some of these kinematic summaries and neural signals collected from the hand region of non-human primates.

The main scientific challenge is that we still lack a full understanding of the mechanism by which the motor cortex controls complex dexterous hand motions. In addition, there a number of challenges associated with working with kinematic data sets as well as with neural recordings. These challenges include confounded sources of variation in repeated trial experiments, a large number of degrees of freedom in finger movement, plausible non linear effects in the kinematic data, as well as intrinsic noise in neural and kinematic recordings.

In this work we address the above challenges in a number of ways: we developed statistical algorithms to efficiently predict hand motion from neural recordings (Koyama et al., 2010a, 2008), modelled hand dynamics (Castellanos et al., 2013; Castellanos, 2010; Castellanos et al., 2008), investigated the neural encoding of some these representations (Castellanos, 2010; Castellanos et al., 2008); and defined an approach for decoding discrete grasping features (Castellanos et al., 2010). We elaborate on each of these topics as follows.

Modelling hand dynamics and neural encoding of movement. In the past people have reduced noise and modelled the coordinated and aggregated action of hand components through linear dimensionality reduction methods (Santello et al., 1998; Todorov and Ghahramani, 2004; Mason et al., 2001, 2004; Soechting

and Flanders, 1997; Pesyna et al., 2011; Thakur et al., 2008). In our work we found evidence of non linearities in the data, and therefore we modelled hand kinematics through non linear dimensionality reduction methods. We also found some evidence of neurons encoding the non linear synergies of the data (Castellanos, 2010; Castellanos et al., 2008). In (Castellanos et al., 2013) we incorporated temporal variation into the modelling of the data, and exploited the repeated trial structure of the grasping dataset that we studied. We adapted a Multivariate Gaussian Process (MGP) based model, the *Gaussian Process Factor Analysis* model (Yu et al., 2009), to decompose finger motion into two interpretable terms: a term that is shared among all replications of the same reach-and-grasp task and a term that is particular to each replication and that is modelled with a MGP, which provided a dynamic lower-dimensional representation of finger motion.

Predicting hand motion from neural recordings (Koyama et al., 2010a, 2008). Estimating hand kinematics can be challenging due to the large number of parameters involved in representing hand configuration and its dynamics, as well as the fact that neural activity is not normally distributed. Moreover, computing the optimal hand kinematic states is made even more difficult in the presence of non linearities in dynamics. We developed the *Laplace Gaussian Filter*, a fast, accurate and deterministic approximation for the recursive Bayesian computation (based on Laplace's approximation) of the posterior expectation, that allows one to effectively deal with non linearities and with high dimensional states. Our approach outperforms sequential Monte Carlo methods in a fraction of the time, and we show results in simulations and on real data.

Decoding event-based grasping (Castellanos et al., 2010). In order to investigate the encoding of motor commands, only a single neuron is required. In contrast, to solve the reciprocal problem of decoding or predicting hand dynamics from neural activity several dozens of simultaneously recorded neurons are typically needed. In this thesis, we investigated whether some information that meaningfully describes grasp is signalled by *single* neurons. We defined interpretable discrete variables that summarize and describe the aggregated motion of fingers. For example, we consider the event corresponding to the time at which the maximum or the minimum fist aperture is attained, or the maximum or minimum finger spread or curliness is attained. Then we proposed a framework to decode these events (in contrast to decoding full grasp trajectories) based on the historical pattern of firing of the neurons. We were able to show the existence of single neurons that reliably decode relevant features of grasping that potentially can be used to control a basic prosthetic device with discretized commands. In fact, we found that we only require a single neuron to reliably perform the decoding, in some cases getting accuracies up to 96%. Even when more neurons are available, these discrete events might be more stable to estimate, and can be thought of in some sense as a dimensionality reduction of the kinematic data.

In all these studies we used kinematic and neural recording datasets from experiments from rhesus monkeys. In Section 1.1 we briefly explain the similarities that allow us to draw general conclusions about humans from data collected from non-human primates and we explain the type of neural data that we analyze. In Section 1.2 we survey the different ways of studying and modelling movement. In Section 1.3 we explain the problems of motor-neural encoding and decoding together with a literature review of related work and our contributions put in context. Readers familiar with these concepts can skip to Section 1.4 where we present the thesis structure and main contributions.

1.1 Rhesus monkeys as models

In order to study human processes, it is common to use animal models. In particular, *Macaca mulatta* monkeys (or rhesus monkeys) are good models of human beings for motor tasks as they share anatomical and functional features of brain and upper limbs. Like humans, Rhesus monkeys have arms and hands with five fingers including an opposable thumb (see Figure 1.1). This similarity allows for the design of experiments where the non-human primates perform motor tasks that humans would do, that bring insights into the mechanics of human motor control.

As for the brain, the topology of the somatosensory and motor areas is very similar in both species (see Figure 1.1). In particular, there exists 'great' architectonic similarities between the somatosensory and motor areas in macaques and humans, and the distribution of patterns of many receptors in the somatosensory and motor areas of both species have many features in common (Zilles et al., 1995).

In order to study neural-motor processes, technological devices need to record the activity of specific motor areas while subjects perform a specific motor task (for a review of motor areas associated with voluntary movement refer to (Kandel et al., 1991)). There are different methods for recording neural activity when performing a motor task. These methods vary in the type of signals sensed, granularity (in time and space), and in the degree of invasiveness for the subject. Non-invasive methods involve placing external sensors on or near the scalp, while invasive techniques are based on arrays of electrodes that are typically placed, through a surgical implantation procedure, on the surface of the cerebral cortex penetrating a few millimetres into the brain, situating the electrodes' tips within close proximity of individual neurons.

Individual neurons are considered to be the the fundamental information encoding unit in the nervous system. Intracortical electrode arrays allow the extraction of single unit behaviour at the highest spatial resolution (individual neurons, micro spatial scale) and temporal resolution (milisecond timescale)(Gilja et al., 2011). Therefore, at the cost of a surgical intervention, this technology is very useful for understanding the way by which the brain *encodes* motor commands and, at the same time, for extracting information to control motor-neural prosthesis. We refer the reader to a discussion of the burden-benefit of this technology in (Gilja et al., 2011).

In this work, we focus on hand kinematic data sets paired with intracortical recordings taken from the hand region of the primary motor cortex (Figure 1.2) of a rhesus monkey performing a motor task.

1.2 Studies of hand and fingers kinematics

In the literature, hand kinematics can be studied through experiments where kinematic variables, such as position, velocity, acceleration, direction, force, torque, etc. are recorded from a subject (human or non-human primate) performing a specific motor task. Some examples of motor experiments involve subjects moving cursors, moving their hand following a target in a virtual environment (*target pursuit*), moving a cursor or the hand from a central location to surrounding targets (*center-out-tasks*), pulling levers, performing sequences of movements, pushing buttons, exploring, reaching and/or grasping objects of different sizes and shapes, etc. While the subject performs the task, different kinematic variables can be simultaneously recorded and analyzed or correlated with other variables such as neural signals.

In this thesis we analyze several such datasets, most of which correspond to rhesus monkeys trained to reach-and-grasp different objects presented in different orientations and spatial locations. The subjects are comfortably seated in a primate chair, with one hand restrained and the other free to move to perform the

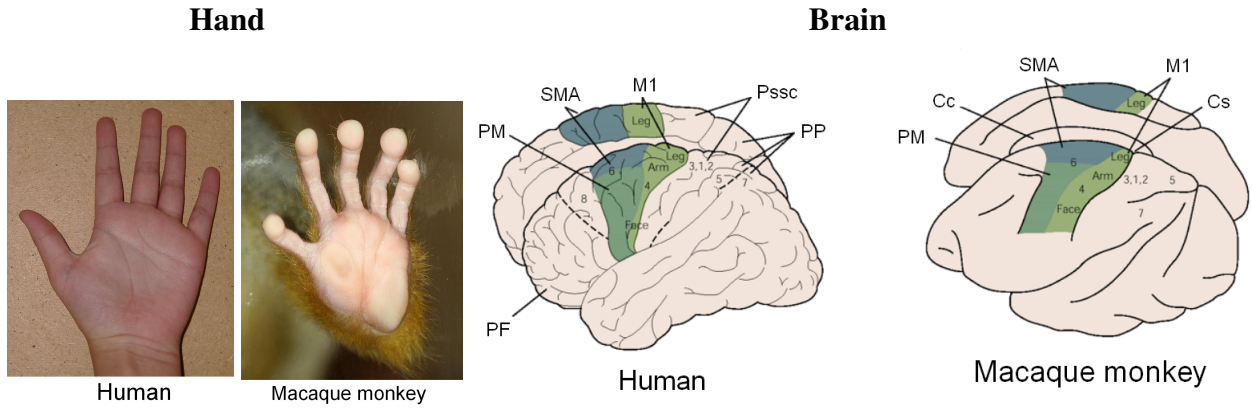


Figure 1.1: **Hand.** Correspondence of hands between human and macaque monkey. **Brain.** Correspondence of Motor Areas in human and macaque monkey brain. *Premotor cortex* (PM), *Supplementary motor area* (SMA), *Primary motor cortex* (M1), *Prefrontal cortex* (PF), *Primary somatic sensory cortex* (Pssc), *Posterior Parietal Area* (PP), *Corpus callosum* (Cc), *Central sulcus* (Cs). (Figure extracted from (Kandel et al., 1991).)

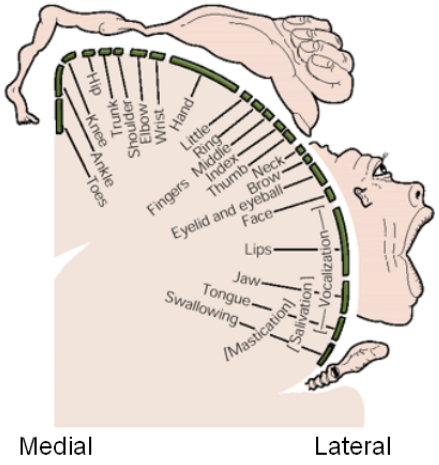


Figure 1.2: Somatotopic organization in *Primary motor cortex* (M1) that is similar to motor cortical areas of rhesus monkeys. (Figure extracted from (Kandel et al., 1991).)

task. A state-of-the-art motion tracking system (Vicon Inc) is used to record the three-dimensional (3D) positions of passive markers placed on a thin custom made glove worn by the monkey, at a rate of 200Hz. The markers are positioned at the center of each of the fingers' phalanges, on the wrist and on the back of the hand. Each replication of the reach-to-grasp task corresponds to a specific condition (i.e. an object presented in a specific orientation) and constitutes a multivariate time series of markers' position. We analyze these datasets from Chapter 2 through Section 5.1.

The other dataset that we analyze corresponds to a monkey performing a 3D center-out-task where the 3D trajectory of the hand is recorded while the monkey reaches from the central position to the desired target in a repeated trial experiment (Section 5.2).

1.2.1 Studies on grasping

The earliest studies of grasping tried to understand hand configurations by creating grasp taxonomies (Napier, 1956; Cutkosky, 1989; Iberall, 1987, 1997; Goldfeder et al., 2009; Bock and Feix, 2008) – see Figure 1.3 upper panel for an example. The feasibility of defining a taxonomy suggests that it is possible to include virtually all of the possible grasp configurations into a finite set. That is, while the configuration space of dexterous hands is high dimensional, it is possible that most useful grasps can be found in the neighborhood of a small number of discrete points (Ciocarlie et al., 2007).

It has also been observed that there is clear mechanical coupling on the hand components, meaning that the digits do not move in isolation from contiguous digits (Ross and Schieber, 2000), and that there exist clear correlations in the motion of digits. In the kinesiology and neurophysiology literature there has been work trying to explain the space of hand configurations based on the coupling of the digits and their correlation, through a small number of elements that have been called *synergies*.

The extraction of linear static synergies (understood as summaries of variation) in primate grasping has been performed in different experimental settings including static hand poses grabbing imaginary (Santello et al., 1998) and real (Todorov and Ghahramani, 2004) objects, reach-to-grasp tasks (Mason et al., 2001, 2004), skilled activities (Soechting and Flanders, 1997), and unconstrained haptic exploration tasks (Pesyna et al., 2011; Thakur et al., 2008) (see Table 7.1). In these works it has been shown that it is possible to obtain a small number of synergies that describe most of the variance of the different possible hand configurations. In all of these works, synergies are obtained via principal components analysis (PCA), a dimensionality reduction technique that assumes linear relationships on the variables representing the digits and that cannot capture non linear structure. Matrix decomposition methods, like PCA or singular value decomposition (SVD), in their standard formulations, are not suited for the the experimental trial structure of the data. In fact, these approaches confound kinematic variability with time variation, and result in a loss of information when the experimental design structure is ignored and data is collapsed into a large rectangular matrix.

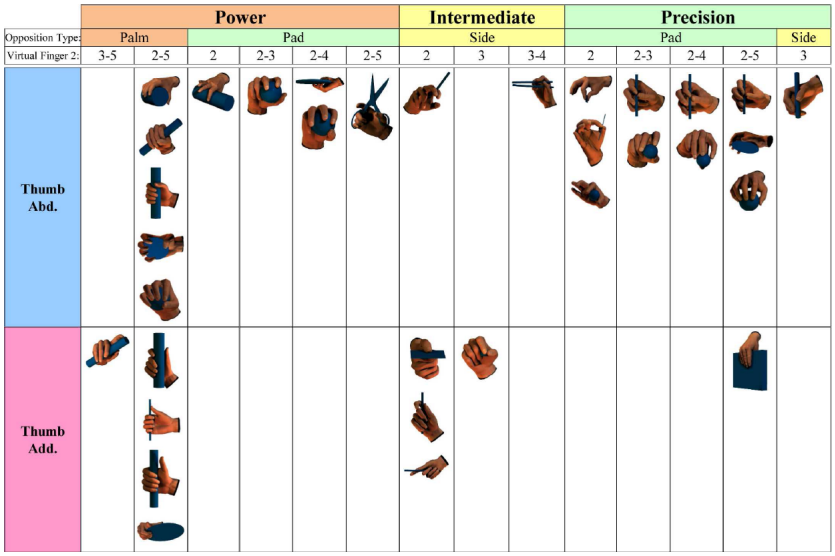
Other approaches in the literature have incorporated temporal information into the extraction of grasping synergies. For instance, Vinjamuri et al. (2007, 2010a,b) inspired in (d'Avella and Bizzi, 2005) proposed two convolved-mixture models, that use SVD and an optimization step in their core, to learn a dictionary of time varying *synergies*– while this approach is able to describe time varying phenomena, it does not provide a generative model of grasping. State-space models are generative models that have been used to model dynamics of general body motion (Wang et al., 2008) albeit not finger dynamics – in these models a Markovian assumption is made and thus long range time correlations are unable to be directly captured.

1.2.2 Our contribution modelling hand kinematics

We model finger kinematics during reach-to-grasp movements. In particular, in Chapter 3 we obtain static non linear kinematic synergies at relevant time landmarks and we show improved classification accuracy of objects being grasped at every landmark as compared to linear synergies. While the non linear synergies capture better the information of object being grasped, due to the focus on critical landmarks, they do not model the entire trajectory and moreover, are not interpretable.

In Section 5.1 we explicitly define interpretable hand engineered synergies that describe relevant features of grasping such as grasp opening, finger curling and finger spread. And in Chapter 4 we adapt the Gaussian Process Factor Model from Yu et al. (2009) to model finger kinematics. This Multivariate Gaussian Process Factor Model considers entire trajectories at once, and decomposes and reduces the dimensions

Example of a grasp taxonomy



Visualization of static linear eigengrasps

Model	DOFs	Eigengrasp 1			Eigengrasp 2		
		Description	min	max	Description	min	max
Gripper	4	Prox. joints flexion			Dist. joints flexion		
Barrett	4	Spread angle opening			Finger flexion		
DLR	12	Prox. joints flexion Finger abduction			Dist. joints flexion Thumb flexion		
Robonaut	14	Thumb flexion MCP flexion Index abduction			Thumb flexion MCP extension PIP flexion		
Human	20	Thumb rotation Thumb flexion MCP flexion Index abduction			Thumb flexion MCP extension PIP flexion		

Figure 1.3: *Top panel:* Figure from Bock and Feix (2008): A unifying Hand Taxonomy. *Bottom panel:* Figure from (Ciocarlie et al., 2007): Synergies or eigengrasps that were obtained as in (Santello et al., 1998) but from four robotic hand models and a human hand model.

of the variation of the data into interpretable elements that capture what is common between replications and what is different. Additionally, by applying an alignment procedure based on the total kinetic energy, we also address time variation.

1.3 Encoding and decoding problems

Neurons propagate signals through electrical pulses called action potentials or *spikes* that propagate down nerve fibers. These action potentials convey information through their timing. In order to understand and to reproduce (through prosthetic devices) a specific phenomena like volitional movement two reciprocal problems can be posed: neural encoding and decoding. Neural encoding refers to measuring and characterizing how stimulus attributes or motor actions are represented by action potentials; it corresponds to learning a mapping from stimulus to neural response such that the function explains the neural activity in terms of the motor action/stimulus.

The inverse problem of neural decoding consists of mapping a representation of the neural activity of an organism to a specific behavior or interaction with the environment. Some examples of neural decoding are: mapping thought to a specific word or meaning, mapping thought to limb movement, guiding the selection of a specific key on a virtual keyboard with thoughts (Kennedy et al., 2000) or moving a cursor or a robotic arm based on thoughts (Velliste et al., 2008; Hochberg et al., 2012). In the context of motor control, the goal of the decoding analysis is to predict the kinematic variables from neural activity originating in motor areas of the brain.

1.3.1 Encoding

The encoding problem can also be viewed as a search of all possible stimuli that are able to generate replicable and specific patterns in the neuron or population of neurons. But it is not possible to evaluate all possible stimuli, as the search space of all possible stimuli is too large. Hence, the encoding problem is essentially statistical: given a finite number of samples, noisy physiological data and all possible stimuli how do we estimate the neural code? (Paninski et al., 2006). In other words, how do we provide a model that explains the probabilities that different spike sequences are produced by specific stimuli?

Although in this work we follow this *parameter representational* approach of searching in the space of kinematic variables those that can consistently explain the firing rate of a neuron, recently there has also been work on a *dynamical systems* approach, where, instead of trying to estimate the firing rate of a neuron or population of neurons in terms of a function of kinematic variables, one tries to estimate the *change* (or derivative) of the firing rate in terms of the firing rate and external variables (Churchland et al., 2012). The dynamical systems view does not contradict the parameter representational approach (for a discussion refer to (Churchland et al., 2010; Shenoy et al., 2011)), but it opens up new perspectives and opportunities to understanding the neural code for voluntary movements.

Statistical models for neural encoding

The probability density for the occurrence of a spike is the *firing rate* of a neuron and there are different ways to estimate this statistic (Dayan and Abbott, 2001; Cunningham et al., 2009). The average firing rate of a neuron written as a function of the stimulus (or of the parameters that describe the stimulus) is the *tuning*

curve. One way to provide an encoding model is to show that the response of a neuron or a population of neurons to specific stimuli can consistently be described through a deterministic tuning curve. This is the case, for instance, for the cosine tuning curve found by (Georgopoulos et al., 1982) used to describe preferred direction in neurons in the primary motor cortex.

Tuning curves as described do not provide a model for the variability or noise in the neurons. However it is possible to describe the encoding model through a probabilistic model such as a standard multiple linear regression model or a point process. Even though standard regression methods are designed for the analysis of continuous valued data, after some pre-processing of the data, these methods have shown to reliably encode kinematic variables (Kettner et al., 1988; Schwartz, 1992, 1993; Ashe and Georgopoulos, 1994; Moran and Schwartz, 1999; Wang and Moran, 2007). Point processes are naturally suited to analyze spike trains as they are processes composed of a time series of binary events that occur in continuous time. These processes are defined through the conditional intensity function which, in the case of neural encoding, specifies the joint probability density of spike times based on the kinematic stimuli. This conditional intensity function corresponds to a deterministic estimate of the firing rate and from it, the model generates a stochastic spiking pattern. Truccolo et al. (2004); Paninski (2004) proposed this framework for neural encoding of kinematic data[1] where they modelled the conditional intensity function through parametric models (e.g. either through a Generalized Linear Model or through other general parametric models). This framework allows us to analyze the simultaneous effects and relative importance of spiking history, neural ensemble and extrinsic covariates. Furthermore it is flexible enough to allow the inclusion of non linear transformations and of latent terms that can help to better describe the anatomy/physiology of the system or model intracellular noise or the dynamical state of the network (Paninski et al., 2006).

There is a large body of work in encoding for voluntary movements; here we present an non exhaustive but representative survey.

Primary motor cortex (M1). Neural activity in M1 is associated with execution of movement. Georgopoulos et al. (1982) and Schwartz et al. (1988) showed that individual neurons in M1 fire preferentially when movements are executed in their *preferred direction* (in 2D and 3D, respectively), and that the tuning profiles of motor cortical cells can be described by a cosine tuning function. Georgopoulos et al. (1986a) also showed that at a population level, while cortical neurons with different preferred directions are active during movement in a particular direction, the activity of the whole population results in a *population vector* that corresponds to the direction of movement. Maynard et al. (1999) showed later, that interactions among groups of neurons in M1 improve population coding of movement direction. The population vector forms the basis for the first decoding algorithm in motor control that we will review in later sections.

While the existance of a particular preferred direction for single neurons in M1 is consistently present, studies have found systematic shifts in preferred direction on single units in M1 while subjects perform standard center-out tasks or random target pursuit tasks (Sergio et al., 2005; Hatsopoulos et al., 2007) and in PMd (Mason et al., 1998). But direction of movement is not the only kinematic variable encoded in neurons in M1. Neurons in M1 show linear neural encoding of 3D hand position and define a *position gradient* (Kettner et al., 1988). Speed of movement has also been shown to be linearly encoded in drawing movements (Schwartz, 1992, 1993) and in reaches (Moran and Schwartz, 1999). Furthermore, Wang and Moran (2007) linearly combined the models proposed in (Kettner et al., 1988) and (Moran and Schwartz,

[1] This paradigm was first applied to hippocampus data by (Brown et al., 1998).

1999) and showed that position and velocity are simultaneously encoded by single motor cortical neurons in an additive fashion, but the velocity signal is more salient during reaching than the position.

Wang et al. (2010) investigated the encoding of orientation and rotational velocity during a center-out with rotation task. They found that single M1 neurons are capable of simultaneously encoding hand position, translational velocity, orientation, and rotational velocity. They also found that hand rotational velocity is more strongly represented than hand orientation; and that when two or more of the mentioned kinematic variables are encoded by the same neuron their relationship can be additive (as in Wang and Moran (2007)) or multiplicative, such as between hand translation and rotation.

Paninski et al. (2004a) investigated the encoding of position and velocity by a single motor cortical neuron during a pursuit tracking task and they incorporated time into the model through the evaluation of individual time lags. They found that moto-cortical neurons linearly encoded a function of the full time varying hand trajectory and showed that although both position and velocity are encoded roughly to the same extent this encoding exhibits markedly different temporal dynamics for each of them.

Other kinematic parameters, such as force, have also shown to be linearly encoded in the firing patterns of neurons in M1 (Sergio and Kalaska, 1998; Sergio et al., 2005; Cabel et al., 2001).

All the studies mentioned above use standard linear models as the encoding paradigm. Paninski et al. (2004b) used a model that fits (Truccolo et al., 2004) to show that M1 cells encode a *superlinear* function of the full time varying hand trajectory in a continuous tracking task. They showed that the exponentiated linear combination of position and velocity constitute a suitable encoding model for M1 neurons (point process-GLM framework (Truccolo et al., 2004; Paninski, 2004)). This showed in a principled way, that the firing of neurons in M1 depends on the full hand trajectory and not just through position and velocity at a fixed time, and that there exist a non linear encoding performed by M1 cells. They also showed dependence of the firing rate on the activity of neighbouring cells.

Hatsopoulos et al. (2007) performed an analysis within the same statistical framework (Truccolo et al., 2004) and investigated the superlinear encoding of normalized velocity *trajectory*, mean speed and mean hand position. Their model can be reduced to cosine tuning (Georgopoulos et al., 1982) and linear tuning of speed (Moran and Schwartz, 1999). This study, which is similar to (Paninski et al., 2004b), provides an explicit trajectory encoding model or tuning function that characterizes the shape of the *preferred trajectories* of neurons in M1. Interestingly, in a similar analysis, they did not find strong encoding of torque trajectories.

Saleh et al. (2010, 2012) took a similar approach building trajectories describing hand shaping during a reach-grasp task. Trajectories were built taking joint angles, angular velocities of fingers, wrist and arm at multiple time lags, and applying principal components to these variables. Then, the components were modeled into the point process-GLM framework to show encoding of the kinematic features by neurons in M1. Their main conclusion was that neurons encode trajectories of the hand's joints during prehensile movements.

All the studies we mentioned in the last paragraphs were performed in rhesus monkeys; however there is at least one paper reporting GLM encoding of motor variables on spike train data in humans (Truccolo et al., 2008). In this work, two humans with tetraplegia were implanted with intracortical arrays in M1 and the encoding of intended movement (velocity and position) in a imagined visually guided pursuit tracking task and a center-out task was performed. A point process-GLM encoding model showed that the M1 spiking activity is strongly tuned to intended movement direction and velocity even though highly variable time lags were observed. In line with (Hatsopoulos et al., 2007; Saleh et al., 2010) the approach of querying

whether specific parameters are encoded in M1 is questioned by Reimer and Hatsopoulos (2009). They argue that inherent correlations between parameters of movement that happen during natural movement cause unsolvable confounded effects and that the highly variant time lag between neural activity and motor variable effect causes noise. They suggest an alternative approach of building a *hyper-volume* in a high-dimensional movement space (similar to the proposed trajectories in their mentioned works) that avoids these issues and that can potentially better explain the neural code in M1.

1.3.2 Decoding

The problem of neural decoding consists in mapping a representation of the neural activity of an organism to a specific behavior, interaction with the environment or variables representing either of them. We focus on the problem of decoding kinematic variables describing voluntary movement from spike trains collected from one of the motor areas on the cerebral cortex. Two good reviews of the main approaches are (Yu et al., 2010; Koyama et al., 2010b). In the Appendix we provide tables summarizing the approaches found in the literature to address the neural motor decoding problem in all its modalities (Tables 7.4, 7.5, 7.2 and 7.3).

On-line control versus Off-line reconstruction

When considering the decoding problem, two settings can be distinguished: on-line (*closed-loop*) control versus off-line (*open-loop*) reconstruction. The off-line setting corresponds to the application of decoding algorithms in pre-recorded data and involves an algorithmic, statistical and data analysis problem. There is a large body of work on off-line kinematic decoding algorithms (see Tables 7.4 and 7.5).

The on-line setting, on the other hand, includes a living subject, a specific task to be performed, a brain-computer interface (BCI) and the efficient implementation of a decoding algorithm that will assist the subject in the performance of the task through the BCI. The on-line setting involves scientific, algorithmic and technological challenges and has been performed in different organisms ranging from rats (Chapin et al., 1999), non human-primates (Wessberg et al., 2000; Serruya et al., 2002; Taylor et al., 2002; Carmena et al., 2003; Velliste et al., 2008), to even human beings (Hochberg et al., 2006; Kim et al., 2008; Hochberg et al., 2012). Refer to Tables 7.2 and 7.3 for more examples.

In our work we focus on the off-line decoding of kinematic variables. However, as a cautionary satement, we remark that it has been shown that decoding performance can highly vary when the same algorithm is applied in the on-line setting versus in the off-line setting (Chase et al., 2009; Koyama et al., 2010b). Some factors that may affect the disparity of performance of decoding algorithms in the off- and the on-line setting include the algorithm's statistical properties (Koyama et al., 2010b), the ability (or inability) for the subjects to compensate for certain biases (Chase et al., 2009; Koyama et al., 2010b), the change of properties of neurons (plasticity), and the ability of the brain to learn and adapt the mapping between neural activity and device movement (Orsborn et al., 2012).

Decoding continuous versus discrete variables: regression versus classification

Movement of an arm/hand or cursor can be described from moment-to-moment through the trajectory that it follows in time. This trajectory can be specified by a set of continuous variables like position, velocity, acceleration, joint angles or any function of these variables. Alternatively, the movement of a device can also be characterized by a discrete finite set of final target locations or configurations, such as, a specific

key on a virtual keyboard or whether a finger is flexed or extended. Mathematically, the distinction between these two types of decoding corresponds to predicting a time series of real numbers (*continuous decoding*) versus predicting a discrete finite number of possible outcomes (*discrete decoding*). In the machine learning community, these correspond to the two different problems of regression and classification.

In our work we provide an algorithm to efficiently solve the decoding of continuous variables in time (section 5.2). We also provide a discrete decoding framework that allows us to recover relevant grasping events from neural activity (section 5.1).

In the next sections we provide a survey of the main techniques for decoding kinematic continuous and discrete variables.

Decoding of continuous variables

There are several approaches for decoding kinematic continuous variables:

Population Vector Algorithm (PVA). First proposed by Georgopoulos et al. (1986b, 1988) the PVA method constructs a vector for each neuron representing the neuron's *preferred direction*. The algorithm linearly combines the vectors of all the neurons into a *population vector* that indicates the instantaneous movement of the hand/cursor at each point in time. The PVA method performs optimally when the tuning functions are linear, the set of preferred directions are uniformly distributed, and spike counts in adjacent time bins are conditionally uncorrelated (Koyama et al., 2010b), which is often not the case in real data. Additionally, since the PVA method is not designed to combine activity across multiple time steps, it frequently generates results that are not smooth. Regardless, the algorithm yields good results in the on-line setting in spite of its non-uniform assumption; the reason seems to be that subjects learn to compensate for the bias introduced by the assumption (Chase et al., 2009). There are various versions of this algorithm based on the same essential idea (Moran and Schwartz, 1999; Taylor et al., 2002; Velliste et al., 2008).

Optimal Linear Filter (OLE). The OLE method (Salinas and Abbott, 1994) also builds vectors of preferred directions for each neural unit. The difference with PVA is that OLE drops the assumption that the directions of the units are uniformly distributed (impacting decoding accuracy). PVA can thus be viewed as a special case of OLE.

Linear Filters (LF). Serruya et al. (2002) demonstrated 2D on-line cursor control (decoding position) with linear filters. This method estimates (each of the coordinates of) the kinematic variable at a particular time as a linear combination of the binned historical activity of the neurons. That is, the design matrix contains the binned firing rate history of each neuron for a period of time (Serruya et al. (2002) considered one second binned in 50mS periods). The design matrix is regressed onto the kinematic variables, and a filter is built for each coordinate. This method was introduced in the visual decoding context by Warland et al. (1997), and adapted to motor-decoding by Paninski et al. (2004a). Linear filters are able to incorporate time information, which results in smoother estimates.

Non-linear methods. Artificial neural networks have been used to perform continuous decoding of kinematic variables (Wessberg et al., 2000; Hatsopoulos et al., 2004; Aggarwal et al., 2008). The disadvantage of this approach is the lack of interpretability of the elements in the model.

Recursive Bayesian Filters (State-space methods). Bayesian filters provide ways of incorporating the uncertainty of the decoding through the use of probabilistic assumptions (Brockwell et al., 2004; Wu et al., 2004, 2006). In particular, two probabilistic models need to be defined: one that describes how the time-evolving kinematic variables relate to the neural activity (the *observation or encoding model*) and another that characterizes the way the motor commands vary from one time point to the other (the *state or trajectory model*). The problem can then be posed as obtaining the most likely sequence of kinematic variables given the observed neural activity; or, in other words, obtaining the *maximum a posteriori* (MAP) probability of the kinematics at a particular point in time given the neural activity observed up to that point. The estimation can be done by applying Bayes rule at each point and recursively thereafter. These models make optimal use of the observed data when the model assumptions are satisfied, yield smooth kinematic estimates (as they incorporate time varying information) and, importantly, provide a framework where it is relatively easy to relax or incorporate assumptions and information about the neural behavior (as modulated by the kinematics of the system) or about the evolution of kinematic variables. This flexibility is reflected in the ability to incorporate: hidden variables that can account for attentional states or other non observed variables (Wu et al., 2009; Lawhern et al., 2010), complex trajectory models (mixtures of trajectories (Yu et al., 2007) or switching states (Srinivasan et al., 2007)), switching observation models (Wu et al., 2004), models with parameters that change over time (Wu and Hatsopoulos, 2008) or feedback (Gilja et al., 2012).

The point process-GLM framework (Truccolo et al., 2004; Paninski, 2004) introduced in Section 1.3.1 naturally fits within the recursive Bayesian decoders. The conditional intensity function is the observation or encoding model and only the trajectory model needs to be specified.

The biggest drawback of Bayesian decoders is the computational complexity, in particular, the bottleneck is the algorithm for calculating or approximating the posterior. In the Gaussian case, exact solutions can be found. When non Gaussian distributions are considered, careful analysis is needed. While Yu et al. (2007) were able to incorporate their complex trajectory model without impacting computational cost, this is not the general case. Some alternative approximate inference methods include variational approximations (Jordan et al., 1999) or Monte Carlo methods such as Particle Filtering (PF) (Brockwell et al., 2004). Monte Carlo methods are computationally expensive. In Section 5.2 we propose an alternative approximation scheme, the Laplace-Gaussian Filter (LGF) – a deterministic method that gives fast and accurate state estimates with bounded error.

Decoding of discrete variables

In some cases, the state to be decoded is a target from a discrete and finite set, for instance, the state can be a key on a virtual keyboard (Kennedy et al., 2000), specific location in space (Musallam et al., 2004; Santhanam et al., 2006), a specific direction (Maynard et al., 1999; Shenoy et al., 2003), whether a single or multiple fingers are flexed or extended (Hamed et al., 2007; Aggarwal et al., 2008; Acharya et al., 2008; Baker et al., 2009; Shin et al., 2009a,b, 2010) or specific grasp types (Stark and Abeles, 2007; Carpaneto et al., 2011; Townsend et al., 2011; Hao et al., 2012). See Tables 7.3 and 7.5 for more examples.

In this context the problem to be solved is a classification task. The classification problem can be

solved discriminatively or generatively (Mitchell, 1997). Some works of discriminative kinematic variable decoding make use of support vector machines, k-nearest neighbors or artificial neural networks (Carpaneto et al., 2011; Hao et al., 2012; Xu et al., 2013). But the most widely used discrete decoders are probabilistic.

In the probabilistic setting the classification problem can be viewed as predicting the most likely target given the neural activity. Some studies use a maximum likelihood approach (Shenoy et al., 2003; Shin et al., 2009a,b, 2010). This approach is the *maximum a posteriori* technique but considering uniform priors. Some studies use the MAP approach (Musallam et al., 2004; Maynard et al., 1999; Baker et al., 2009; Townsend et al., 2011) where the problem can be solved by applying Bayes rule to obtain the most likely target given the data. The likelihood term specifies the probability of the neural activity given a possible target and makes assumptions about the neural behavior. The most commonly used probabilistic models for neural activity (represented as spike counts) are Gaussian (Maynard et al., 1999; Hatsopoulos et al., 2004; Santhanam et al., 2006; Yu et al., 2007) and Poisson (Shenoy et al., 2003; Hatsopoulos et al., 2004; Santhanam et al., 2006) models.

Traditionally, the units are assumed to be conditionally independent given the target, mainly to avoid overfitting by estimating the large number of parameters that comprise the full covariance matrix (in the Gaussian case) (Yu et al., 2010) and because in the multivariate Poisson case, estimation becomes cumbersome (Kawamura, 1979).

While the conditional independence assumption yields good results when using a large number of simultaneously recorded neurons, there have been studies showing that there exists systematic variation in the activity of a neural population, suggesting that the activity in the units is *not* conditionally independent. As pointed out by Yu et al. (2010) some of the factors that have been found to produce this systematic variation include: reach curvature, reach speed, force, type of grasp, attention and others. One way of exploiting the systematic variation include defining latent factors that account for any source of variation and then decomposing the neural data covariance matrix (through factor analysis) into a term that represents shared variability across the neural population and a term that represents what is independent across units. By defining a system that includes these terms, Santhanam et al. (2009) showed that the decoding error is significantly less than in the Gaussian and Poisson conditionally independent models in the off line setting. There are other methods for summarizing and reducing the dimensionality of the neural activity like (Yu et al., 2009) that could be used to leverage the systematic variation in neural data in order to increase decoding accuracy.

Highly accurate discrete decoding of binary kinematic variables, like decoding of single finger movements, requires at least 20-25 neurons (Shin et al., 2009b). Kennedy et al. (2000), on the other hand, showed rough decoding of one dimensional position with one or two units. In Section 5.1 we investigate how accurately we can decode grasping binary *events* with single neurons, and how robust is such decoding. We propose a framework where grasping information is intuitively summarized by interpretable variables. We also identify relevant events that describe grasping and show that a large proportion of neurons is able to decode specific grasping events consistently.

1.3.3 Our contributions in encoding and decoding

In Section 3.2 we show some evidence of improved encoding of non linear static synergies as compared to linear synergies using a multiple linear regression approach. In Section 5.1 we propose a discrete Bayesian classification approach to decode relevant interpretable grasping features. We show, in particular that there

are variables that are consistently decoded using just a single neuron from both hemispheres of the brain of a non-human primate. Finally, to deal with non linearities in the neural signals, we developed the Laplace Gaussian Filter (LGF) (Chapter 5.2) which mitigates the intractability of computing the posterior expectation for the decoding problem by using Laplaces approximation. We show in simulations that LGF outperforms particle filtering in accuracy while running in a fraction of the time and it provides better approximations than PVA in our off-line decoding center-out task.

1.4 Summary and main contributions

We now outline the remainder of the thesis and state the main contributions.

In Chapter 2 we describe of experimental design and the datasets we used in Chapters 3, 4 and Section 5.1. We mention the derivation of joint angles according to the specific experimental settings.

In Chapter 3, we first define relevant time landmarks based on the total kinetic energy, then we investigate static synergies of grasping via a number of different linear and non linear dimensionality reduction strategies. Using these reduced representations of grasping kinematics we evaluate the amount of information preserved in the synergies by predicting which objects are being grasped. We show that as the reach-to-grasp movement evolves, non linear synergies give significantly better results, suggesting that information from the kinematics can be better encoded via non linear features instead of linear features. We also present results from linear encodings of the static synergies on the firing rates of primary motor cortex neurons, yielding comparable results between linear and non linear synergies at population level, but showing evidence of better non linear synergy encoding in some neurons.

In Chapter 4, we go beyond static dimensionality reduction: we incorporate modelling of whole trajectories and account for time variation through statistical alignment of multivariate curves. We propose and fit the Multivariate Gaussian Process Factor Model to represent finger kinematics during reach-to-grasp movements. Our MGPFM model takes advantage of the repeated trial structure in the grasping experiment and separates the common structure across many trials from the specific structure within each trial by fitting interpretable parameters. We develop tractable algorithms for estimating parameters of the model, demonstrating its effectiveness in both simulated and real data. In terms of reconstruction error, we show that extracting structure using our model yields a better fit to the data compared to just using the mean or PCA.

In Chapter 5, we focus on both continuous and discrete formulations of the neural decoding problem. In the first part of Chapter 5, we propose a discrete decoding formulation, in which the goal is to recover kinematic events associated with interpretable hand engineering features that capture relevant aspects of grasping. The discrete on-off events in our event-based framework require much less data to reliably decode, allowing us to demonstrate consistent decoding of events from single neurons. For continuous decoding, our contribution is algorithmic — we develop the Laplace Gaussian Filter (Section 5.2) to deal with the non linear observation models that arise in typical neural decoding problems. Our algorithm yields fast, recursive, and deterministic estimates of the kinematic variables with an error that remains stable over time. In comparison with the (non deterministic) particle filter, our method delivers superior results in a simulation and off-line analysis of real data in a fraction of the time.

Chapter 2

Experimental design and datasets

In Chapters 3 and 4 and in Section 5.1 we analyze and propose models for datasets that come from the same experimental paradigm. In order to avoid repetition, we explain the experiment here and specify which portion of the datasets was used for each of the analysis in the ulterior chapters. In Chapter 5.2 we analyze another dataset corresponding to a center-out task, but we confine its description to that section as it is the only part of the thesis where it appears.

In this chapter we first describe the grasping experiment design and some data exploration, and then we point out what datasets we used in the different chapters of this thesis.

2.1 Reach-to-grasp experiment description

Three *Macaca mulatta* monkeys (*Baxter*, *Vinny* and *Esteban*) were trained to reach and grasp one of ten[1] different objects (Figure 2.2), presented in different orientations. The monkeys were positioned on a primate chair (as illustrated in Figure 2.1 right panel) , with one hand restrained and the other free to move to perform the task. During the task, two datasets were recorded: one corresponding to the hand and fingers position of the monkey and another corresponding to the activity of *some* neurons in the primary motor cortex of the monkeys.

The experiment design and data collection was done at the University of Pittsburgh's Motor Lab directed by Dr. Andrew Schwartz. The data were collected by Dr. Chance Spalding and Dr. Sagi Perel, and the first steps of preprocessing were performed by Dr. Samuel Clanton.

Experimental design. The experiment had a repeated trial structure where an industrial robot presented the objects (Figure 2.2) one at a time at randomly selected orientations. A trial is described in Figure 2.3. At the beginning of the trial, the monkey positioned its hand on the start pad. The industrial robot presented an object in a specific position to the monkey. A cue light turned on and the monkey reached and grasped the object. If the monkey exerted enough force for a randomized period of time, squeezing top- and bottom-mounted pressure sensors on the object as to surpass a threshold, then the trial was saved and the monkey was given a water reward.

[1] Different monkeys grasped different subsets of these objects.

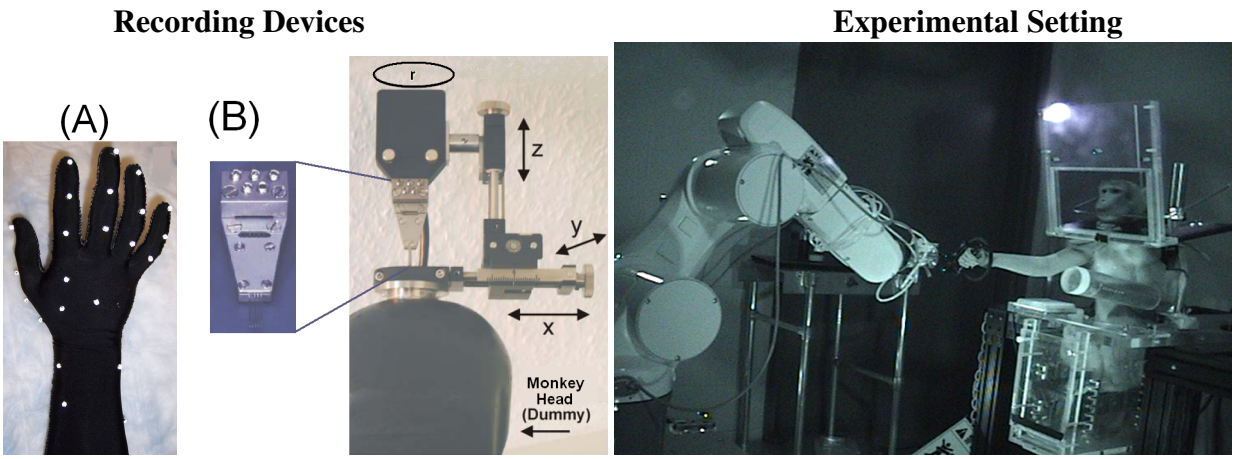


Figure 2.1: **Recording devices.** *(A)* Custom made glove with 23 reflective markers, whose position is tracked by a Vicon Inc. optical marker tracking system. *(B)* Picture of a 5-Channel Electrode Mini Matrix System from Thomas Recording GmbH. The magnified microdrive head shown on the left was inserted every session into the primary motor cortex of the macaques with the device shown at the right. **Experiment setting.** A picture of the monkey sitting on the primate chair, with the robot presenting the object to grasp. The light turns on as a *go* cue.

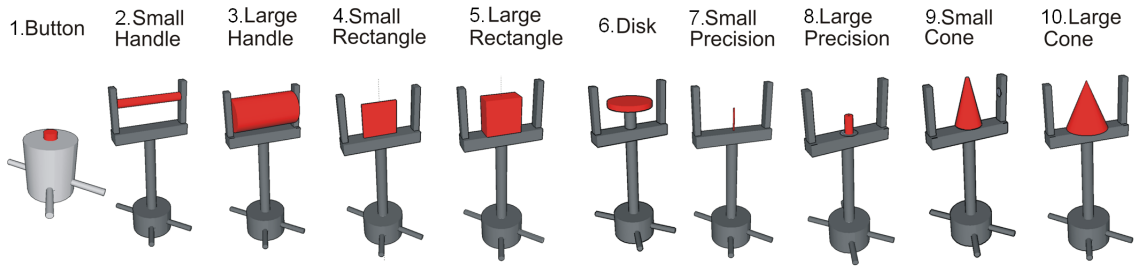


Figure 2.2: Objects presented to the monkeys. Each object was presented in different positions. Not all objects were grasped by all subjects – see text for details.

Vinny was trained to grasp all the objects shown in Figure 2.2, Baxter grasped nine objects (all except the large handle), and Esteban grasped six objects (all the small stimuli, the button and a bar not shown in the figure). All objects, except the button and the bar were presented in seven orientations: horizontal, 45° flexion of the wrist, 45° extension of the wrist, 45° abduction and 45° adduction, 45° to the left and 45° to the right (these last two orientations are a combination of abduction/adduction and flexion/extension). The button was only presented in the first five listed orientations, and the bar was presented in only two orientations: pronation and supination.

For Vinny and Baxter the goal was to record five successful trials for each (object, orientation) pair during each session; and for Esteban, twenty.

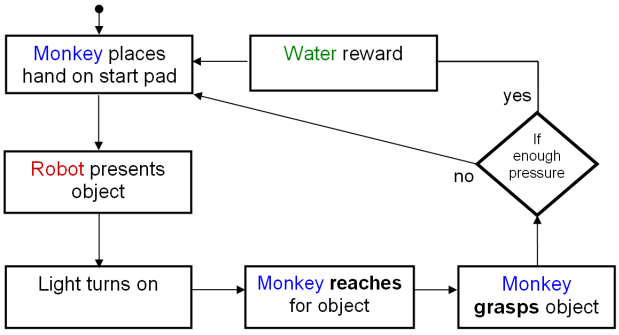


Figure 2.3: A trial of the grasping experiment.

Kinematic recording. A state-of-the-art motion tracking system (Vicon Inc.) was used to record the three-dimensional (3D) positions of passive markers placed on a thin custom made glove worn by the monkey, at a rate of 200Hz. The markers were positioned at the center of each of the fingers' phalanges, on the wrist and on the back of the hand. Each replication of the reach-to-grasp task corresponded to a specific condition (i.e. an object presented in a specific orientation) and constituted a multivariate time series of markers' position.

The orginal kinematic data consists of twenty-three 3D points representing the position of each of the reflective markers swen in the glove (Figure 2.1 left panel). Therefore, a total number of 69 numbers yields the hand configuration *per time point*. There were three markers for each of the following fingers: index, medium, ring and little; four markers for the thumb; three markers for the hand; and four markers for the wrist. The total number of variables *per time point* that correspond only to the finger configuration is, consequently, forty-eight. See Figure 2.5 left panel for a normalized visualization of several reach-and-grasp movements.

The 3D marker positions corresponding to the fingers were summarized into 20 joint angles per time point through a kinematic model used and applied in the Motor Lab. The joint angle dataset is described by four variables per finger (see Figure 2.4): the metacarpo-phalangeal joint (MCP) that is defined by two degrees of freedom the one that describes abduction-adduction (denoted in the figure with Ia where I stands for $index$) and the one that describes flexion-extension ($f1$ in the figure), the proximal interphalangeal joint (PIP) that moves in flexion-extension ($f2$ in the figure) and the distal interphalangeal joint (DIP) that performs a flexion-extension movement ($f3$ in the figure). In Figure 2.5 (right panel) we show the joint angles corresponding to the ring finger for five trials of Baxter grasping the large cone with 45 degrees of abduction (note that each trial has different length).

According to Chang and Matsuoka (2006) the human thumb requires five DOF to be fully described because the axes of rotation in the thumb are neither perpendicular nor parallel and are non intersecting (Veber and Bajd, 2006). This implies a loss of information for the thumb kinematics when represented by four variables. In Section 7.2 we provide our own mathematical conversion from the 3D datapoints to joint angles based of the specific marker configuration and we describe the thumb in spherical coordinates to tackle this loss of information. However, joint angles obtained by applying a kinematic model used in the Motor lab essentially correspond to the 3D original dataset.

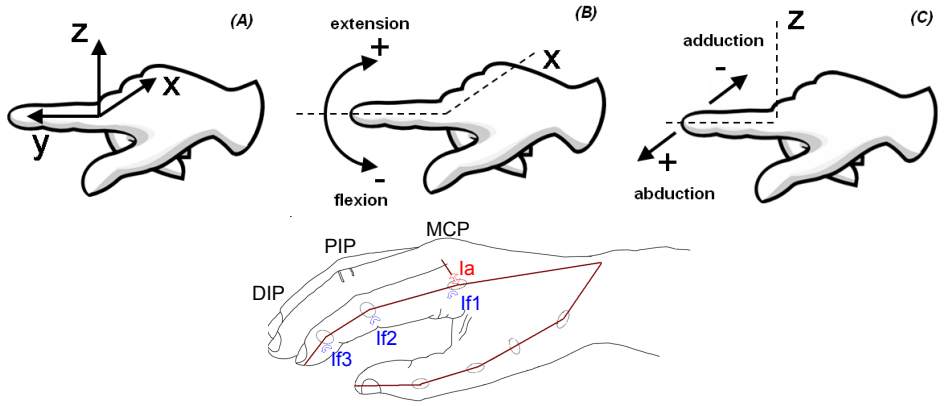


Figure 2.4: *Mechanics of digits (A)* Coordinate system defined for a particular joint. *(B)* The extension-flexion movement of a joint with respect to the x-axis. *(C)* The abduction-adduction movement with respect to the z-axis. (Figure extracted from the *Computational Bioengineering Laboratory*, *National University of Singapore website*) *(Right panel)* Example of the index finger joint angles we used. a stands for abduction, f stands for flexion. The flexion variables are enumerated in order, increasing as they are more distal from the wrist. (Figure adapted from (Veber et al., 2007))

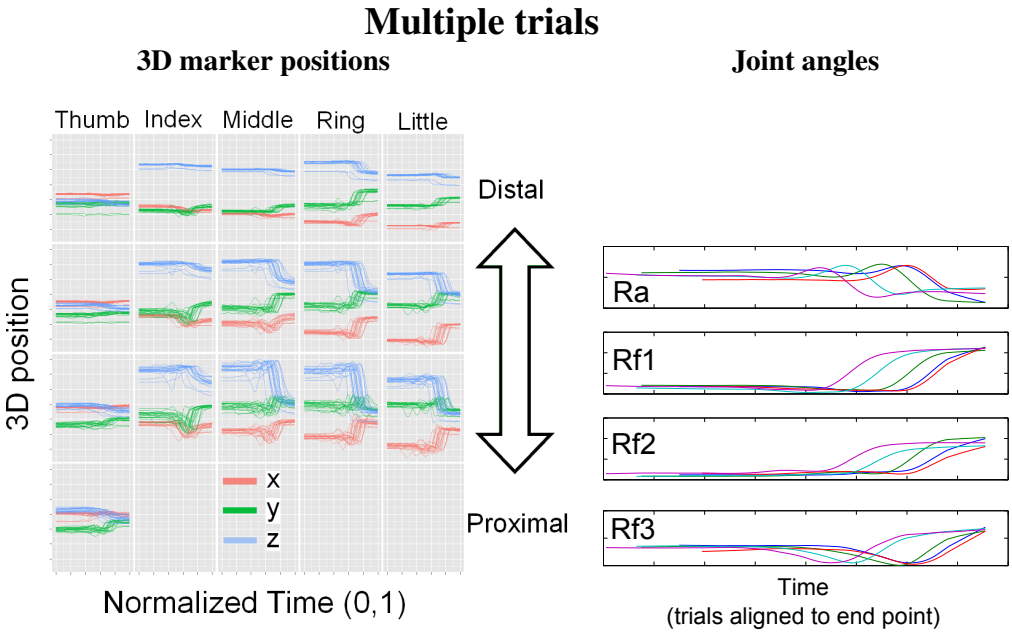


Figure 2.5: *Kinematic data of Baxter grasping the large cone (45 abduction)*: *(Left panel)* 3D marker positions, *(Right panel)* joint angles of ring digit. In the *(Left panel)* columns correspond to fingers, rows to markers, and each line represents a trial; data is normalized to time interval (0,1). In the *(Right panel)* each row is a different joint angle, a color corresponds to a trial (each trial is of different length); trials are aligned to end points.

In summary, we have two kinematic datasets (one for 3D marker position and one for joint angles) describing the grasping of objects from three monkeys.

In Figure 2.5 is evident that when considering the time component, the kinematic curves have a certain

consistency: that is, they present similar features such as peaks or valleys. However, these features occur at different times for each trial, and furthermore, each of the trials is of different length.

Neural recording. Single unit recordings were obtained every millisecond from a 5-Channel Electrode Mini Matrix System, from Thomas Recording GmbH (Figure 2.1 center panel). For every recording session five new electrodes were positioned in the ports in the Mini-Matrix and mounted on the recording chamber of the subject, spike waves were sorted on-line using two discriminator bars resulting in between two to eight isolated units per sessions. Sometimes electrodes moved within a session and not all trials in the session were associated with that unit's activity; and across sessions there are all different units recorded.

To visualize the activity of the units, we constructed raster plots and peristimulus time histograms (PSTH) for each of the studied neurons. These plots display aggregated activity and timing of discharge of the neurons in relation to the reach and grasp task. We built these type of plots aligning the trials in two ways: with respect to (a) the beginning of the trial, and (b) to the end of the trial. Two main qualitative observations resulted from this approach. First, within the population of studied neurons there exist neurons whose peak of activity is towards the begining of the trial, that is, towards the beginning of the reach; but there are also neurons whose peak of activity is towards the end of the trial, that is, during the grasp. Figure 2.6 shows an example of each. It is sensible to hypothesize that the former type of neurons modulate reach, and the latter, grasp. And second, there exist neurons whose discharge is object dependent. An example is shown in Figure 2.6 panel *(B)* this neuron fires preferentially to trials of all objects, except for small handle. In effect, as time progresses, the neuron becomes silent in all trials associated with the small handle. The silent period is emphasized in the graph with an orange striped rectangle.

Neural commands of voluntary movement are generated in the brain before the kinematic behavior is observed. That is, there exists a *lag* between the neural activity representing a command and the kinematic response. As an example, observe Figure 2.7 where neuronal activity is displayed simultaneously with a summary of the kinematic behavior (that will be introduced in Section 3.1.1, but that represents amount of motion of the fingers). Notice, for instance, neuron *spk003a* which consistently shows a higher frequency of spikes several dozens of mS before the largest peak in the energy function happens, one might hypothesize that this neuron is signaling finger motion in the displayed trials. A challenge is to identify the *time lag* for each neuron.

2.2 Datasets for obtaining static synergies (Chapter 3)

In Table 2.1 we provide a summary of how many sessions per monkey we analyzed, together with the total number of trials, the average number of trials per session, the average number of time samples across trials and the average trial length in milliseconds. In the lower panel of the table we show the total number of neurons recorded per session per monkey. In Appendix 7.3 for each monkey, we display the total number of successful recorded trials per (object, position) pair and the mean number of trials per session (Tables 7.6 and 7.7). The latter information will be relevant when considering the neural analysis. The important point of these tables is that for this section: (a) we have considerably more data for Vinny than for Baxter, and that (b) for each monkey there are different numbers of repeats for each (object, position) condition.

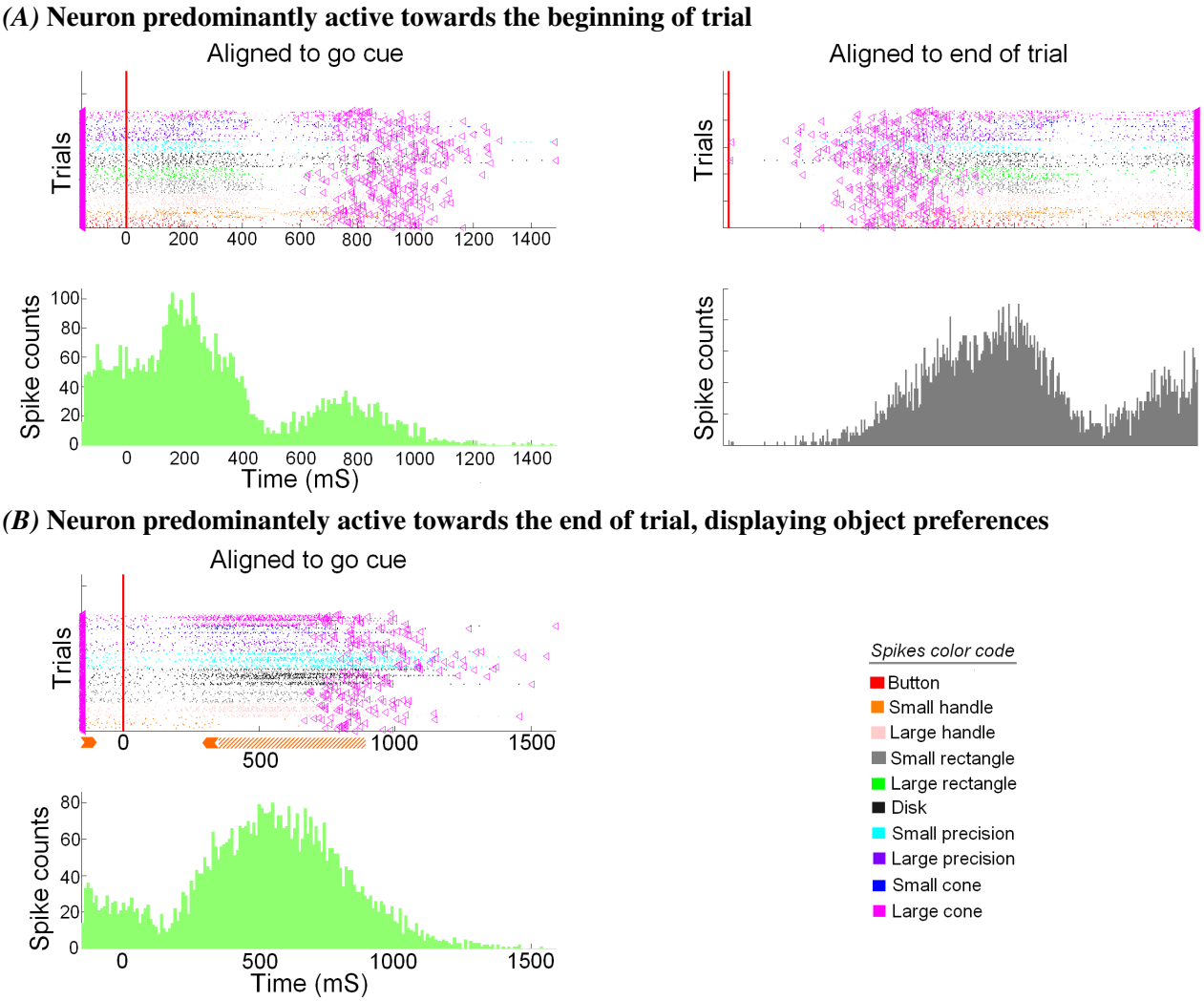


Figure 2.6: Raster plots and peristimulus time histograms showing the activity of the two neurons: Panel *(A)* Session Vinny 000661, neuron 5b. 292 trials. Panel *(B)* Session Vinny 000669, neuron 4a. 216 trials. On the (*left side*) the raster plot and the PSTH are aligned to the *go cue* (t=0), whereas the (*right side*) shows the plots aligned to the end of trials. The different colors of the spikes in the raster plot denote different objects, the spikes color code is shown in the graph. Magenta triangles denote beginning and end of trials. Panel *(A)* shows how the shape of the aggregated activity of the neuron changes as a function of alignment type. The PSTHs in panel *(A)* (and also in panel *(B)*) shows two peaks of activity: a major one, and a minor one. Observe that, regardless of the alignment, the order of peaks of activity in the PSTHs is preserved. That is, even though the shape of the aggregated neural activity does change as a function of alignment, the larger peak always precedes the lower peak in panel *(A)* regardless of the alignment. This is also the case for the neuron shown in panel *(B)* where the larger peak always follows the lower peak (alignment with respect to end of trial is not displayed). In a nutshell, regardless of alignment, we can observe the timing of the highest peak of activity of the neurons and hypothesize which part of the reach-grasp they are modulating.

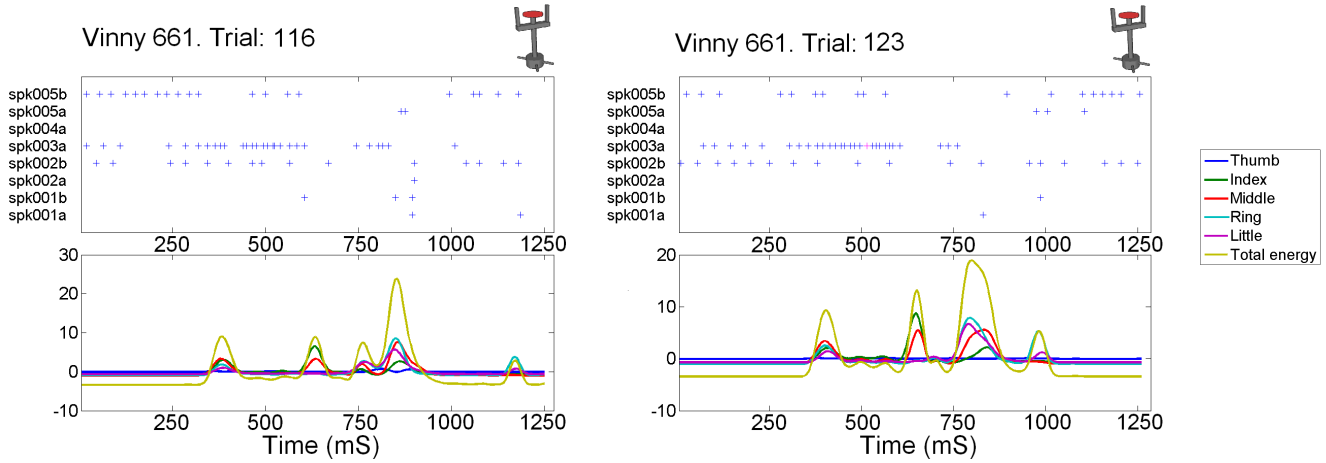


Figure 2.7: Spike trains and measure of movement of fingers for different trials. The kinematics of the fingers was summarized in the *energy function* that will be introduced in Section 3.1.1). The upper panels display a cross in the (time, neuron) position if the neuron presented an action potential at the given time. Neural activity was binned in 5mS intervals, to correspond to the frequency at which the kinematics data was sampled. Crosses in magenta denote that more than one action potential happened in the corresponding 5mS bin. In this case, all the energy functions were centered at zero, which explains why some energy values are displayed as negative.

2.3 Datasets for obtaining dynamic synergies (Chapter 4)

In Chapter 4 we propose a model for dynamic synergies. We used data from Esteban. In Table 2.2 we summarize the data we have for Esteban. In the analysis we analyzed the 48-dimensional finger motion captured data from all 23 sessions and focused on five conditions: small cone and small handle presented in positions 45° of flexion and 45° of adduction, and for the small cone also at 45° of abduction. In each condition we considered all trials across sessions, totalling in average 155 trials per condition.

Even though we did not pursue a full encoding or decoding analysis with this data, we did some preliminary exploration summarized in Appendix 7.4 where we show a (1) visual analysis of which sessions are suitable for analysis of simultaneous neurons based on the number of valid trials and simultaneous neural recordings Figure 7.5, and (2) a glimpse on the heterogeneity of the population of recorded neurons based on their modulation during the reach-to-grasp movement Figure 7.6.

2.4 Datasets for decoding grasping events (Section 5.1)

In Section 5.1 we pose a decoding task. Thus our focus is on neurons. In Table 2.3 we show the number of neurons we analyzed for Vinny. We considered the two hemispheres, and we required that a neuron had fired in (at least) 100 trials. If that condition was fullfilled we also required the neuron to spike in average at least 10 times during the whole reach and grasp movements. For the kinematics we obtained hand engineered interpretable variables that are explained in the corresponding section.

Dataset for static synergies

General sessions and stimuli information

	Vinny	Baxter
Num. sessions	16	4
Total num. trials	3,166	672
Trials per session (mean $\pm$ std)	198 $\pm$ 71	168 $\pm$ 25
Mean number of time samples ($\pm$ std)	211 $\pm$ 33	155 $\pm$ 35
Mean trial length [mS] ($\pm$ std)	1,052 $\pm$ 164	772 $\pm$ 176

Number of collected neurons

Session	Num. valid neurons	Session	Num. valid neurons
Vinny000639	5	Baxter000467	7
Vinny000658	6	Baxter000475	3
Vinny000661	8	Baxter000478	4
Vinny000669	4	Baxter000481	5
Vinny000673	4		
Vinny000676	1		
Vinny000678	4		
Vinny000680	3		
Vinny000682	4		
Vinny000683	4		
Vinny000687	1		
Vinny000690	5		
Vinny000691	5		
Vinny000693	5		
Vinny000694	5		
Vinny000695	3		
Total number	67		19
Neurons with at least 100 trials	37		13

Table 2.1: Dataset for static synergies. (*Top table*) General information of kinematic data collected from Vinny and Baxter. (*Bottom table*) Number of recorded neurons per session per monkey. In the last row we show the total number of neurons per monkey that fired in at least 100 trials.

Dataset for dynamic synergies

General sessions and stimuli information

Number of:	Total	Per session (total: 23 sessions) Mean	Min	Median	Max
Trials	9,465	411	34	495	717
Objects		5	1	6	7
Trials per object		70	22	82	119
(Object, orientation) pairs		29	7	32	39
Trials per (obj, orient.) pair		13	4	15	22
Neurons	152	6	3	6	10
Task related neurons					

Specific information per session

Session	Num trials	Num obj	Avg trials per obj	Num (obj,orient)	Avg trials per cond.	Num Neurons
Esteban000597	687	7	98.14	39	17.62	9
Esteban000612	527	6	87.83	32	16.47	4
Esteban000613	86	3	28.67	14	6.14	4
Esteban000615	717	6	119.50	32	22.41	6
Esteban000620	185	6	30.83	32	5.78	6
Esteban000623	487	6	81.17	32	15.22	3
Esteban000640	213	6	35.50	32	6.66	9
Esteban000642	34	1	34.00	7	4.86	5
Esteban000644	362	6	60.33	32	11.31	8
Esteban000645	45	2	22.50	11	4.09	6
Esteban000647	325	6	54.17	32	10.16	10
Esteban000648	120	4	30.00	25	4.80	6
Esteban000655	563	6	93.83	32	17.59	3
Esteban000659	527	6	87.83	32	16.47	6
Esteban000661	553	6	92.17	32	17.28	6
Esteban000663	495	6	82.50	32	15.47	7
Esteban000667	496	6	82.67	32	15.50	8
Esteban000669	639	6	106.50	32	19.97	10
Esteban000671	517	6	86.17	32	16.16	9
Esteban000675	362	6	60.33	32	11.31	7
Esteban000677	516	6	86.00	32	16.13	9
Esteban000679	517	6	86.17	32	16.16	6
Esteban000683	492	6	82.00	32	15.38	5

Table 2.2: Dataset for dynamic synergies. (*Top table*) General session information. (*Bottom table*) Specific information per session.

Dataset for decoding grasping events

Data set	Number of neurons $\geq$ 100 trials	AND $\geq$ 10 spikes/trial	Average number of spikes per trial	Mean $\pm$ std seconds per trial
Vinny right hemisphere (left hand)	83	37	20.57	1.07 ± 0.23
Vinny left hemisphere (right hand)	155	67	17.52	1.03 ± 0.28

Table 2.3: Dataset for decoding grasping events.

Chapter 3

Static synergies and their kinematic encoding

Given the repeated trial experiment structure described in Chapter 2 we can identify several sources of variation in the recorded reach-to-grasp movements: time, conditions, replications and hand components. In this chapter we study finger kinematics (joint angles) at specific relevant time slices, effectively removing time variability. The goals are to obtain synergies, which we understand as summaries of variation, that help to understand finger mechanics during grasping, and to investigate their encoding in neurons in the primary motor cortex.

This chapter is divided in two main sections. In the first part we define time landmarks during grasping motion based on the total kinetic energy of the trial, then obtain grasping synergies (at those landmarks) through linear and non linear dimensionality reduction methods, and finally evaluate the amount of information that these synergies contain through a classification task. In the second part we investigate linear encoding of the obtained grasping synergies through multiple linear regression, and we verify that our results are not due to chance by bootstrapping.

3.1 Learning static synergies

Previous work has applied matrix decomposition methods to similar grasping datasets (see Table 3.1). Our approach differs in two ways: we focus on specific relevant time landmarks and we apply non linear (kernelized) methods to extract synergies. With this strategy we compare features that happen at specific points in time, and enhance the possibility of extracting structure from data. However, by applying kernelized methods we lose interpretability and increase computational cost.

While the application of PCA or SVD capture linear interactions between observations, in our exploratory analysis we found evidence of non linearities between some variables in the dataset; in particular, we observed some parabola-shaped relationships between variables (Figure 3.1). We therefore explore the hypothesis that modelling non linearities in the kinematic data better captures the configuration of the digits during grasping and improves object prediction accuracy.

To explore this hypothesis we first obtain the synergies (linear and non linear) through dimensionality reduction methods. And then we evaluate the ability of these synergies to classify what object is being

Task	Subjects	Coordinate system	Method	Reference
Static synergies				
• hold imaginary object	human	JA	PCA	(Santello et al., 1998)
• specific manipulation of actual objects	human	JA	PCA	(Todorov and Ghahramani, 2004)
• reach-to-grasp	human	3D	SVD	(Mason et al., 2001)
• reach-to-grasp	monkey	3D	SVD	(Mason et al., 2004)
• skilled activity	human	JA	PCA	(Soechting and Flanders, 1997)
• unconstrained haptic exploration	human	JA	PCA	(Thakur et al., 2008)
Dynamic synergies				
• grasp and hold	human	JA	convoluted mixture model	(Vinjamuri et al., 2007, 2010a,b)

Table 3.1: Summary of related work in extraction of hand synergies. *Static* methods do not account for timing, instead they confound time variability. *Dynamic* synergies account for the variation in timing of features during movement (we defer discussion of these synergies to Chapter 4). JA stands for joint angles, and 3D for the three dimensional position of markers. (For more detailed information of the data they used and more details on the methods, refer to Table 7.1 of the Appendix.)

grasped.

The dimensionality reduction methods we use for synergy extraction are principal components analysis (PCA), kernelized versions of PCA (kPCA) with different kernel functions and linear discriminative analysis (LDA) [1]. We chose to use kPCA because of the type of curvature we observed in some plots (like in Figure 3.1), where we observe a data distributions that could be represented potentially with either a polynomial kernel of degree higher than one or a Gaussian kernel. We use LDA because it is a supervised method and thus can serve as a reference of how well the dimensionality reduction can be done in an unsupervised manner compared to the supervised manner. Since we only decided to use this method as a reference, we do not further explore other supervised methods or their non linear extensions, like kernelized discriminant analysis (Mika et al., 1999; Baudat and Anouar, 2000).

For the evaluation criterion, we use the following classifiers: naive Bayes and Multivariate Support Vector Machines. The Naive Bayes classifier is a supervised generative classifier based on Bayes rule. It makes the assumption that the considered variables are conditionally independent given the class to which the input data belongs. This assumption is made to reduce the complexity of general Bayesian classifiers from $O(2^n)$ to $O(n)$ (Mitchell, 1997). We selected this classifier because of its simplicity, low complexity and because it has been applied successfully in applications in spite of the conditionally independent assumption being invalid. We selected the discriminative SVM because it generally results in good classification accuracies in other applications.

In Appendix 7.5 we briefly survey the methods we use for the data analysis.

[1] We also tried graph based dimensionality reduction methods such as local linear embedding (LLE) and isomap but their performance was highly sensitive on the number of neighbors, so we omit the results.

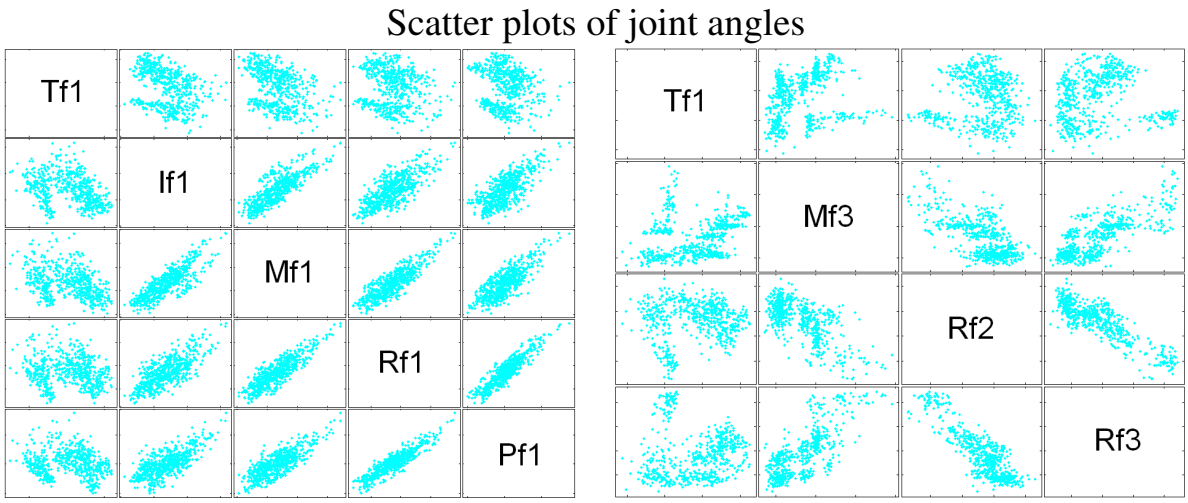


Figure 3.1: Evidence of linear and of non linear relationships between joint angles. Matrix of scatter plots of joint angles from Vinny at the end of the trial (the last landmark as defined in Section 3.1.1) when grasping the object labelled 'small precision'. Each plot shows the relationships of two joint angles (T= thumb, I= index, M= middle, R= ring, P= pinky; f= flexion; 1=proximal, ..., 3= distal). There is evidence of parabola-shaped relationships between variables.

3.1.1 Landmark definition

An important property of a reach-to-grasp movement is the magnitude of motion of the fingers during the trial, we can quantify this magnitude by aggregating the motion of each of the sixteen markers attached to the fingers. The amount of motion of the digits can be written as a function of the velocity of the markers along time, and we call it the *total energy signal*.

Let $\mathbf{Y}^r(t)$ denote the $K \times 3$ *matrix* containing the 3-dimensional position of the $K = 16$ markers attached to the fingers for replication r at time t. And let $\dot{\mathbf{Y}}^r(t)$ denote the corresponding velocities. Then $\mathbf{G}^r(t) = [\dot{\mathbf{Y}}^r(t)][\dot{\mathbf{Y}}^r(t)]^{\mathrm{T}}$ is the matrix of inner products of marker velocities for each replication r in a specific condition, and the sum of the squared magnitudes of the velocities across markers is:

$$\mathbf{E}^r(t) = \mathtt{tr}(\mathbf{G}^r(t)) = \mathtt{tr}([\dot{\mathbf{Y}}^r(t)][\dot{\mathbf{Y}}^r(t)]^{\mathrm{T}}). \tag{3.1}$$

$\mathbf{E}^r(t)$ is an important property of the trial because it summarizes the aggregated magnitude of motion of the fingers during a trial. In general, the energy profile of a specific trial starts close to zero, presents one or several peaks, and goes back to be close to zero at the end of the trial (see Figure 3.2 for two examples). In addition, in every case there is a well-defined maximum for the energy. Thus, we define the relevant time *landmarks* with respect to the time slice t^r_{max} corresponding to the point where the maximum of the energy of replication r is attained (this time slice is labelled c in the central panel of Figure 3.2).

In addition, we consider two time slices before t^r_{max} and two time slices after t^r_{max}. The first two landmarks correspond to the first landmark *before* the maximum and to the last landmark before the maximum, and thus they are denoted by $t^r_{firstBef}$ and by $t^r_{lastBef}$ respectively. Landmark $t^r_{firstBef}$ (labeled a in Figure 3.2) is important because it corresponds to the *beginning* of the trial, since it is defined as the first time that the amount of motion of the digits reaches a $\rho\%$ of the total motion of the trial, where ρ is

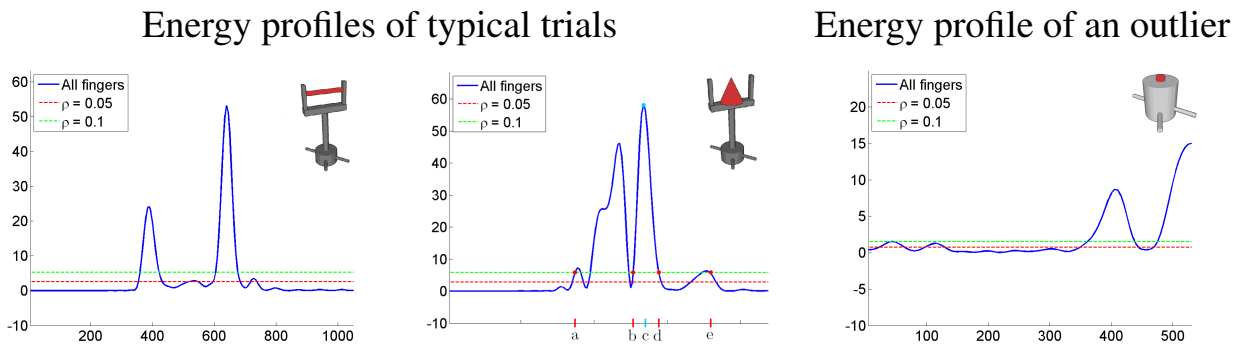


Figure 3.2: Energy profiles of different trials, landmark definition and profile of an outlier during the reach-grasp movement. Time landmarks are defined with respect to the amount of fingers' motion and make reference to the time t^r_{max} (time when the maximum of the energy of replication r is attained). In the plots two horizontal lines show $\eta^r(\rho)$ for $\rho = 0.05$ and $\rho = 0.1$. In the example in the center we show the definition of our times of interest: *(a)* $t^r_{firstBef}(\rho)$, *(b)* $t^r_{lastBef}(\rho)$, *(c)* t^r_{max}, *(d)* $t^r_{firstAft}(\rho)$, and *(e)* $t^r_{lastAft}(\rho)$. (*Left panel*) Vinny: session 673, trial 238; small handle at $-45°$ rotation. (*Center panel*) Baxter: session 475, trial 147; large cone at $45°$ abduction. (*Right panel*) Baxter: session 467, trial 25; button, $-45°$ rotation.

small. Landmark $t^r_{lastBef}$ (labeled b in Figure 3.2) denotes when the energy function is starting to climb the hill that will maximize it; formally, it corresponds to the last intersection of the energy function with line $y_\rho = \rho \cdot (max_t(\mathbf{E}^r(t)) - min_t(\mathbf{E}^r(t)))$ before t^r_{max}. The last two landmarks happen *after* the maximum and are labelled $t^r_{firstAft}$ and $t^r_{lastAft}$. The latter corresponds to the end of the trial, i.e. when the energy (or the total finger motion) is decreasing towards zero for the last time (last intersection with line y_ρ). Whereas $t^r_{firstAft}$ corresponds to the finger motion reaching zero for the first time after the energy function was maximized.

These landmarks are important because they closely correspond to crossings of zero of the aggregated finger velocity and to the peak of velocity, that is, they are critical points of the aggregated digits movement.

In Figure 3.2 *(center panel)* we show an example of energy profile together with the defined landmarks referring to the maximum amount of motion of the fingers (indicated by c in the figure) and the other landmarks defined by the ρ-percent of motion of the trial: a and e denoting the first and last intersection with the line $y_\rho = \rho \cdot (max_t(\mathbf{E}^r(t)) - min_t(\mathbf{E}^r(t)))$, where $\rho = 0.05$ or $\rho = 0.1$, and b and d denoting the closest intersections preceding and succeeding $max_t(\mathbf{E}^r(t))$. In this chapter, we use these landmarks as reference points to compare trials at meaningful moments in time.

3.1.2 Data analysis

For this section we considered data from the grasping experiment described in Chapter 2. We analyzed sixteen sessions from Vinny and four sessions of Baxter). We included all conditions and all recorded replications and we applied the following analysis.

Preprocessing

Three preprocessing steps were performed: outlier removal, selection of time of interest and data summarization.

(1) Outlier definition. The energy function (Equation 3.1) provides us with the means of defining an outlier. There are trials where the energy function is not close to zero at the end of the trial. This could mean either that *(a)* there was an error in defining the end of the trial, or *(b)* the configuration of the hand at the defined end of trial is not stable, and there is still a lot of finger movement.

We defined a trial as an outlier when the motion of the fingers as the end of the trial has not gotten *close* enough to zero, that is, when there is no $t^r_{lastAft}$ different than $t^r_{firstAft}$. Refer to Figure 3.2 for an example.

Under this definition Baxter had a 10% and Vinny 2% of outliers when $\rho = 0.05$, and 5% and 1.5% when $\rho = 0.10$. This indicates that Vinny reached and held a steadier hand configuration compared to Baxter at the end of the grasp.

(2) Selection of time of interest. In order to frame the problem as a classical dimensionality reduction task we need to build a matrix $\mathcal{Q}$ which contains in its rows the trials and in its columns the variables.

Many authors (Soechting and Flanders, 1997; Todorov and Ghahramani, 2004; Thakur et al., 2008) have built $\mathcal{Q}$ by considering each time point as an independent trial and have stacked the variables X^r_t corresponding to different time points t and different trials r all together as follows: $\mathcal{Q} = \left(X^1_1, \ldots, X^1_{T(1)} | \ldots | X^N_{T(N)}, \ldots, X^N_{T(N)}\right)^{\mathrm{T}}$. This approach *(a)* removes the temporal information in the data, and *(b)* treats each time point as independent from the others. In our approach, we select a specific time of interest (like in (Santello et al., 1998)) in order to make data comparable and remove time variability.

(3) Sampling versus averaging. Once a time point of interest is fixed (see 3.2 for a definition of relevant times of interest), and outliers are removed, we end up with a matrix of size $N \times p$ where p is the number of variables describing hand configuration, and N is the number of replications whose value is: $N_{Baxter} \approx 600$ and $N_{Vinny} \approx 3,100$.

In kernelized versions of PCA, the time complexity is cubic in the number of samples since we need to spectrally decompose a matrix of size $N \times N$. The analysis of Vinny's data takes a large amount of time. In order to be able to run several experiments with 10-fold cross validation (in a single machine), we considered ways of summarizing the data. In particular, we considered (a) sampling a specific number of trials, preserving the distribution of frequencies of (object, position) pairs; and (b) averaging across trials fixing an (object, position) pair, to obtain a representative trial from each condition.

In the case of sampling we considered all the trials for Baxter because the number of trials for him is not prohibitive for applying kernelized dimensionality reduction methods. However, for Vinny we sampled $1,000$ trials. Averaging, on the other hand, drastically summarized the data into a matrix that contained only one representative mean trial per (object,position) pair for each of the monkeys. To obtain the representative mean trial, we averaged across trials fixing (object,position). Figure 3.3 shows a schematic view of the analysis after outliers removal.

Experiments

We considered the joint angles data set for each monkey, that is, we obtained postural joint angle synergies. And performed the following steps:

1. Select:

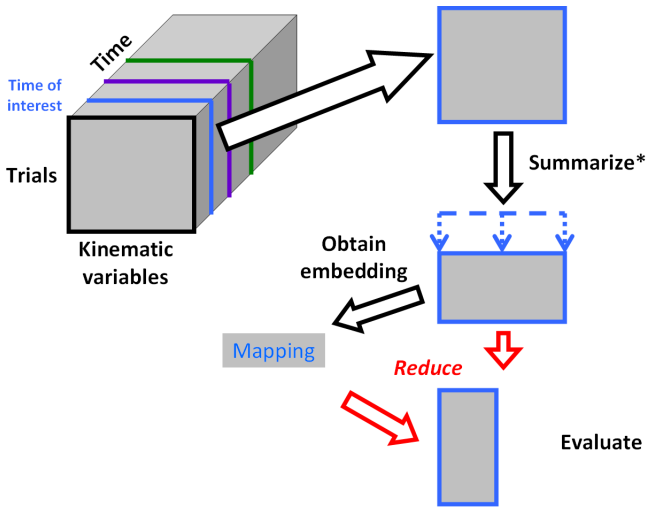


Figure 3.3: A schematic diagram of the analysis after outliers removal. *Summarizing* refers either to *(a) sampling* a certain number of trials preserving the distribution of number of trials that belong to a specific (object,position) condition; or *(b) averaging* across trials fixing (object,position). The latter yields a drastic reduction of the number of trials in the matrix that will be used to obtain the mapping. One of the objectives of the analysis is to determine which strategy yields better results according to the measures of goodness.

(a) A specific time of interest: to study the hand configuration when object is being grasped we considered $t_{lastAft}(\rho)$. And, to study the pre-shaping of the hand we considered $t_{firstBef}(\rho)$, $t_{lastBef}(\rho)$, t_{max} and $t_{firstAft}(\rho)$.
(b) An outlier definition: we tried $\rho = 0.05$, and $\rho = 0.10$.
(c) A summarizing strategy:
    i. Sampling:
        A. Baxter: we used all the data, since the total number of trials does not exceed 700.
        B. Vinny: we sampled 1,000 trials preserving the distribution of conditions.
    ii. Averaging across trials fixing (object, position) condition.

2. Build a matrix D of kinematic information with the options selected in the previous step. Each row corresponds to a particular trial at the selected landmark (in the case of sampling), or to a mean trial at the selected landmark (in the case of averaging); and each column corresponds to the kinematic variables that describe the hand configuration, in this case, the joint angles.
3. Use D to obtain a mapping to project the high dimensional data to a low dimensional data. The mapping was obtained using: Principal Components Analysis, Linear Discriminant Analysis, and kernel Principal Components Analysis with polynomial kernel (degree 2 and 3) and with Gaussian kernel ($\sigma = 0.3, 0.5, 0.8, 1, 1.5, 2, 2.5$)
4. Obtain the low dimensional embeddings of the original data through the mappings obtained in the previous step.
5. We applied the classification methods to: the original data, and the reduced data. We considered the following number of components for the reduced data:

(a) PCA, kPCA: 2, 3, 4 to 20 in steps of size two.

(b) LDA: 2, 3, 4, 6, 8.

6. Finally, to get a sense of the preservation of the data, we obtained the 10-fold cross validation accuracy from all the D matrices with:

   (a) Multi-class support vector machines

   (b) Naive Bayes classifiers

We report the main results in the following section.

3.1.3 Results

For the kPCA-Gaussian method we selected $\sigma^2 = 2$ after 10-fold cross validation because this value yielded the best results for both Baxter and Vinny. Also, we found that the performance of the synergies when using kPCA-polynomial degree 2 and 3 was very similar, so we focused on the results of kPCA (with polynomial kernels of degree 2 and 3). Local linear embedding and Isomap yielded solutions very sensitive to the number of neighbors, and they performed very poorly compared with the other methods so we exclude them from the results. We compared the accuracies obtained from classifying the original kinematic data (not reduced) versus the classification accuracies of all the obtained synergies. We focused mainly on unsupervised classification methods because the main goal is not to classify objects, but for comparison we show the results of Linear Discriminant Analysis which is a supervised linear dimensionality reduction method (see Figure 3.7).

Evaluation at different times of interest. The classification accuracy of all the methods increases as the reach-and-grasp evolves in time and the hand reaches the final hand configuration. We can conclusively show that the classification accuracy of the original data at the last landmark (signaling the end of grasping) was on average better than all the other time points we considered. The first four landmarks represent the *preshaping* of the hand. These results can be observed in Figure 3.2, where we show the performance of all the dimensionality reduced data at different time points along the evolution of the reaching. The plots correspond to the landmarks signaled in Figure 3.2 (center panel).

In our experiments we observed that starting from the point in time when the fingers start increasing their aggregated motion during the reach, kernel PCA with polynomial kernel of degree 2 consistently outperforms other dimensionality reduction methods when Support Vector Machines are used. In the same setting, the Gaussian kernel-reduced data consistently performs the worst, even compared against PCA.

Classification accuracy also depends on the specific classifier. The combination of PCA and Naive Bayes classifier yielded surprisingly good accuracies for lower numbers of components in Baxter (but not in Vinny). Nevertheless, the main result is that non linear methods can capture information from the grasping that linear methods cannot.

Variance explained with Linear Principal Components. We show the mean number of principal linear components needed to explain 85% and 95% of variance averaged across all the times of interest in Tables 3.3 (the details for each time of interest are in Table 7.8). The first fact to notice is that the mean number of components needed to explain a specific amount of variance is relatively robust to the definition of outliers

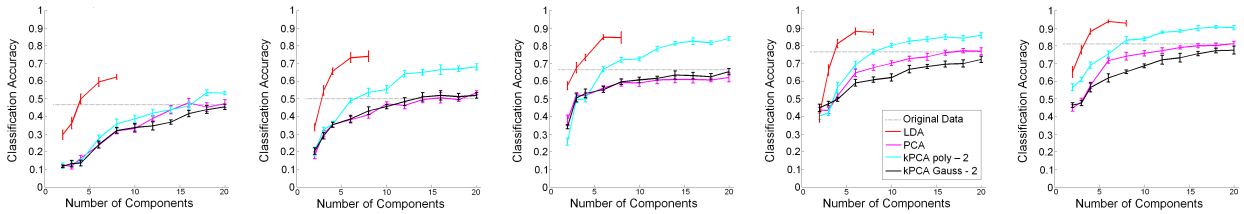


Table 3.2: Preshaping and final configuration of the hand: at the left the earliest time point in the reach is plotted; going further to the right, the hand is closer to the object to be grasped. The classification task is to predict the object that is being grasped using the obtained static synergies. The horizontal line in each plot represents the classification accuracy using the raw (unprojected) features. In this case chance level accuracy is 11%. The data is from Baxter with Support Vector Machine classification, and similar results are seen from Vinny with SVM.

through the threshold defined by ρ. Secondly, the number of principal components needed to explain the averaged data are lower than the number of principal components needed to explain the sampled data.

Various authors have reported the number of components needed to explain specific percentages of the variability of the data sets they considered. The number of components they reported are shown in Table 7.1 of the Appendix.

Our results are only comparable to those of Santello et al. (1998), since they also considered only one time point as opposed to all time points of the movement. The differences are that *(a)* our subjects are monkeys, and theirs, human beings; *(b)* they only considered 15 out of the 20 joint angles that describe the hand configuration, and we included all joint angles. The condition where our results were closest to them was in Baxter, when $\rho = 0.10$ and when the eigenvalues of the covariance matrix was obtained from the averaged data across trials fixing (object, pair) condition. But, in general, the number of principal components we report are slightly higher particularly for Vinny. A plausible explanation for this is that the number of variables we considered is larger than the number of variables (Santello et al., 1998) used.

The amount of variance explained by kernel PCA is not directly comparable to PCA, since the covariance that is being obtained is the covariance matrix in the feature space, not in the space of variables.

Robustness to outliers definition criterion. Not only is the mean number of components needed to explain 85% and 95% of the variance in the PCA reduced data practically the same when considering different outlier criteria. But also, the difference in accuracy given by removing outliers with $\rho = 0.05$ and with $\rho = 0.10$ is not meaningful. We found thus, that our results are relatively robust to the specific threshold for defining outliers, and we fixed $\rho = 0.05$.

Sampling versus averaging to obtain mappings. We contrasted the strategy of sampling versus averaging the data set to obtain the mappings, and found that as a whole the classification accuracies are not affected for any of the unsupervised dimensionality reduced methods. However, the performance of the embedding using Linear Discriminant Analysis is deteriorated when averaging is used as opposed to sampling. In Figure 3.4 we show the phenomenon: for all number of components, the accuracy of LDA obtained with

Baxter

Baxter	Sampling		Averaging	
	85%	95%	85%	95%
Outliers $\rho = 0.05$	4.2	7.6	2.5	5.1
Outliers $\rho = 0.10$	4.1	7.3	2.5	4.8

Vinny

Vinny	Sampling		Averaging	
	85%	95%	85%	95%
Outliers $\rho = 0.05$	4.8	8.1	3.1	5.8
Outliers $\rho = 0.10$	4.8	8.3	3.5	5.8

Table 3.3: Baxter and Vinny: mean number of components needed across all times of interest considered, for the different alternatives of the analysis.

the mappings constructed from averaged data is lower than the accuracy of LDA obtained with the mappings built from sampled data. This is a representative example of all the comparisons between averaging and sampling in all the different experimental setups.

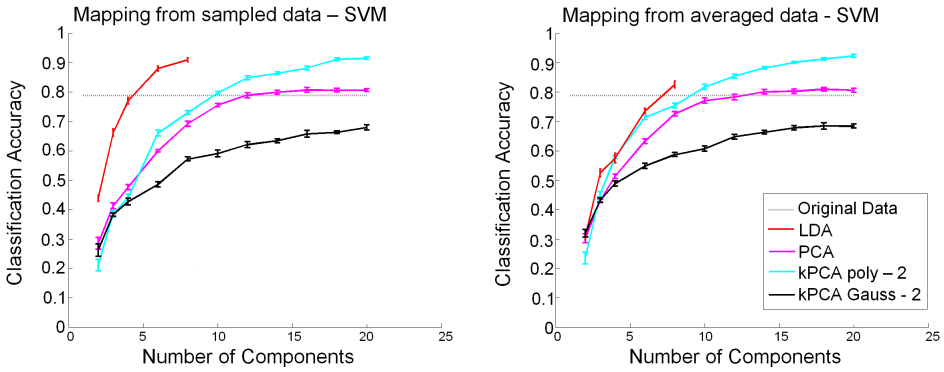


Figure 3.4: Contrasting summarizing strategies: sampling versus averaging. Using the averaged data across trials fixing (object, position) pair considerably deteriorates the performance of Linear Discriminant Analysis, whereas the performance on the classifiers remains relatively unchanged for the unsupervised dimensionality reduced embeddings. The performance of LDA when evaluated with Naive Bayes classifiers is also worse when using averaged data as opposed to sampled data to obtain the reduction mapping. In this plot: Vinny. Classification method: SVM. Time point: end of grasp (*lastAft*).

Classification accuracies of reduced data. The dimensionality reduced method naturally depends on the kernel. The two main options we tried were polynomial and Gaussian kernels, and they exhibit different behavior when evaluated with the classification methods. Classification accuracy also depends on the specific classifier. In our experiments we observed that kernel PCA with kernel polynomial of degree 2 performs consistently the best as compared to other dimensionality reduction methods, when Support Vector Machines are used. In the same setting, the Gaussian kernel-reduced data performs consistently the worst, even

compared against PCA.

With the SVM classifier, the PCA embeddings rarely go higher than the accuracy yielded by the Original Data (Figure 3.4). In contrast, the PCA embeddings coupled with a Naive Bayes Classifier yield higher accuracy. One possible explanation to why Principal Component Analysis perform better when trained with Naive Bayes as opposed to Support Vector Machines is that PCA might rotate the data in a way that it more closely satisfies the conditional independence assumptions.

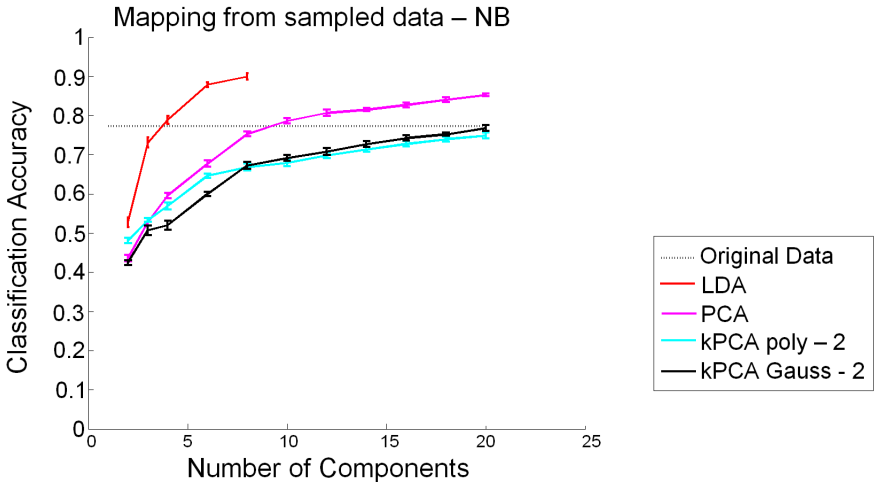


Figure 3.5: Contrasting classification methods: Support Vector Machines (Figure 3.4 Part *A*) versus Naive Bayes classifier. The accuracy obtained from linear and non linear reduced data evaluated through Naive Bayes classifiers is not meaningfully different. In this plot: Vinny. Classification method: Naive Bayes. Time point: end of grasp (*lastAft*).

Confusion matrices. We now show qualitative results that bring insight into which objects are easier to classify, and we elaborate on possible explanations for this. In Figure 3.6 we show graphical representations of confusion matrices for the three unsupervised dimensionality reduction methods coupled with a respective Support Vector Machines classifier for Baxter. We show (from left to right) an increasing number of components. A confusion matrix $\mathcal{C}$ contains in its (i, j) coordinate a count of observations known to be in group i but predicted to be in group j.

On the left most column we observe the confusion matrices obtained with only two components. The object that is easiest to classify by all classifiers, and only with *two* components, is the button. Through observation of movies representing the hand movements we observe that the pattern of *grasping* is indeed quite different from grasps of other objects. The monkeys use their middle finger to push the button, and the rest of the fingers are extended. The hand configuration for all the other objects is very different than that pattern, and more similar between the other objects in which more classic power or precision grips are observed. The clear differentiation of hand configuration from button to all other objects can also be observed in the scatter plots of the LDA projections of the data (see Figure 3.7).

From left to right there is an increasing organization of the matrices towards the diagonal. The method that brings the elements fastest to the diagonal is kPCA with polynomial kernel of degree two. Furthermore, observing the right-most column of matrices which corresponds to taking *all* the components, the method

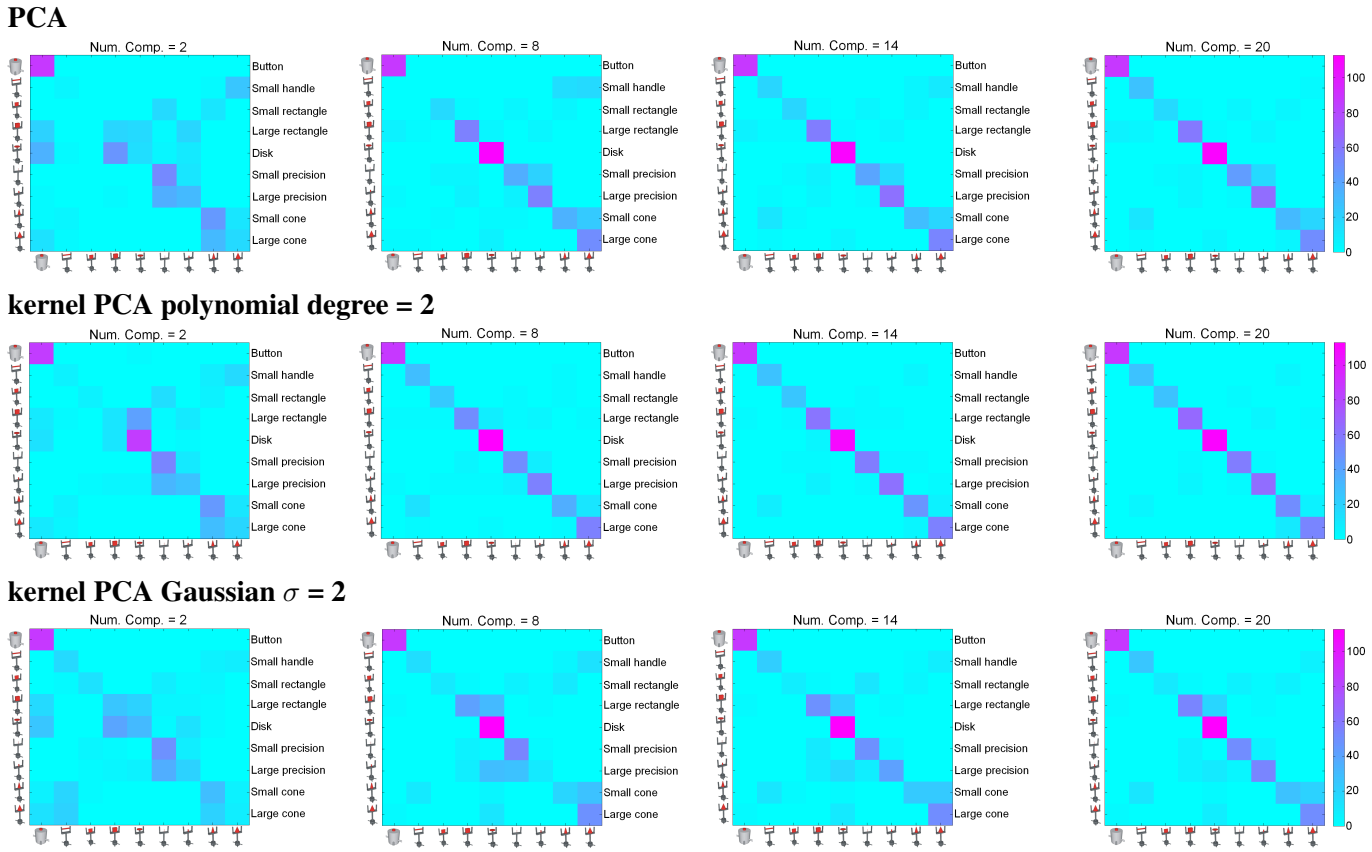


Figure 3.6: Confusion matrices obtained from SVM classification on Baxter data at the landmark at the end of the grasp from the two non-supervised dimensionality reduction methods. On the y-axis the true object is shown; on the x-axis the predicted object is shown. A sequence of increasing number of components considered is shown from left to right.

that presents the cleanest pattern of elements only in the diagonal is kPCA polynomial of degree 2 (also compared to kPCA with Gaussian kernel). The performance of kPCA with Gaussian kernel is the worst, and objects which PCA is not able to differentiate even with all the components (like the small precision versus the large precision) are successfully classified by kPCA with polynomial kernel (see Figure 3.6's last column of matrices, coordinates corresponding to (small precision, large precision)).

In the first column we also observe blobs of dark color outside the diagonal. Common troubles for all classifiers when only two components are considered are: *(a)* the large rectangle is classified as the disk (and vice versa, except for kPCA polynomial degree 2); *(b)* the large precision is classified as the small precision (and vice versa); *(c)* the large cone is classified as the small cone (and vice versa), and *(d)* surprisingly the small handle is classified as the large cone.

These qualitative results show evidence that unsupervised non linear dimensionality methods can extract more information per component than unsupervised linear methods, and also that non linear synergies extract information about the object to be grasped earlier in the reach.

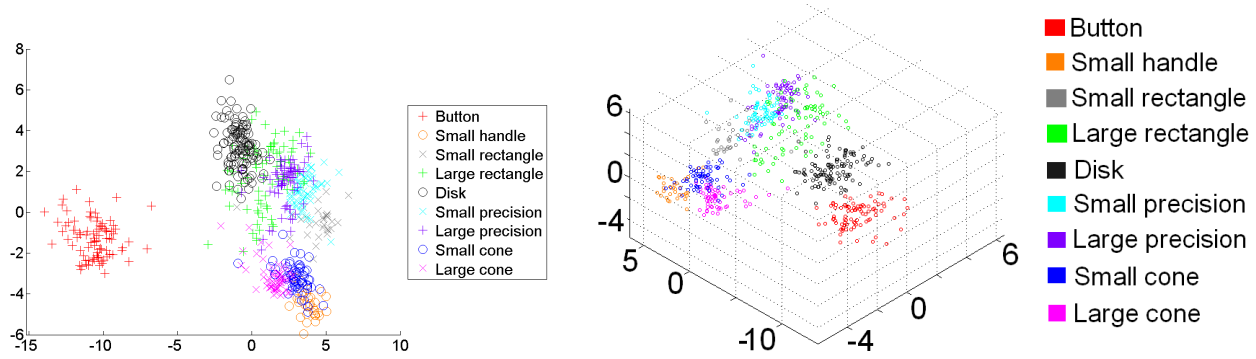


Figure 3.7: First two and three components obtained from the Linear Discriminant Analysis projection of Baxter at a time point where the grasp is fully configured to the object. Note the separation of objects in the 3-first components space suggesting that different objects elicit different hand configurations that can be distinguished from each other.

3.1.4 Conclusions

The performance of non linear versus linear dimensionality reduction depends on the specific classifier used, as we have seen from experiments. However, we have shown that modelling the non linear relationships in the data is helpful for classifier performance. Out of all the experimental settings for both monkeys, the setting which achieves the highest accuracy with 10 components is always the Support Vector Machine with polynomial kernel degree 2. Our main finding is that modelling the non linear relationships in the data resulted in better classifier performance, and this strongly supports our hypothesis that non linear relationships in the data can be captured with kernelized (non linear) dimensionality reduction methods.

3.2 Encoding of linear and non linear synergies

In the previous section we obtained joint angle linear and non linear synergies for grasping movements in the defined landmarks in time. In this section our goal is to investigate whether these linear and non liner synergies are encoded in the firing pattern of the collected neurons. That is, whether we can predict the neural activity (and to what extent) based on the synergies, through multivariate linear regressions.

In addition to the joint angle synergies we obtained joint angle velocity synergies, 3D postural synergies and 3D velocity synergies in an analogous fashion. We investigate their encoding too.

Related work. In Section 1.3.1 we provide a general literature review of models for motor encoding. Here, we mention some studies specifically related to grasping. Hendrix et al. (2009) studied standard multiple linear encoding of grasp force and dimension (size) during reach and grasp movements in neurons in PMd and M1. They found that these two parameters were encoded in both regions but the correlations with grasp force were stronger in the firing of M1, and across both regions the modulations with these parameters increased as reach to grasp proceeded. They also found that although neurons that signaled grasp force also signaled grasp dimension (not vice-versa) the two signals exhibit limited interactions suggesting that the

neural basis for control of these two parameters is independent. Chen et al. (2009) performed an analysis-of-variance type encoding analysis (essentially a linear model) associating the way of approaching to a grasped object and the shape/type of grasped objects to the firing rate of neurons in the posterior parietal cortex. They found that the way of approaching to the object was strongly correlated with firing rate and that the object shape/type was not correlated. Mason et al. (2006) performed an analysis-of-variance type encoding analysis of the size and shape of objects grasped and of force exerted in a reach-grasp task in simple[2] spike trains of Purkinje neurons in the cerebellum. They found that there was significant force and object related modulation in these neurons, but that the signals were not correlated between each other.

More closely related to our work is the work of Saleh et al. (2010, 2012) who built *trajectories* describing hand shaping during a reach-grasp task. Trajectories were built taking joint angles, angular velocities of fingers, wrist and arm at multiple time lags, and applying *linear* principal components to these variables. Then, the components were modeled into a point process-GLM framework (Truccolo et al., 2004) to show encoding of the kinematic features by neurons in M1. Their main conclusion was that neurons encode linear trajectories of the hand's joints during prehensile movements.

In our work we use multivariate linear regression to investigate the encoding of synergies, and it is different from others because we focus on specific landmarks and because, to our knowledge, no one has studied the encoding of synergies obtained with non linear dimensionality reduction methods.

3.2.1 Methodology

Pre-selection of neurons

We performed a pre-selection of neurons. Two criteria had to be fulfilled by a neuron in order for it to be considered eligible for the analysis: (a) the neuron had to be associated with at least 100 trials; (b) the neuron had to be *task-related*.

In order to define *task relatedness* we divided each trial into three epochs: *premovement*, *reach* and *grasp configuration*. *Premovement* extends from the beginning of the trial until landmark $t_{firstBef}$; *reaching or prehension* included the period between landmark $t_{firstBef}$ and landmark $t_{lastAft}$; and *grasp-hold* was the period from landmark $t_{lastAft}$ until the end of the trial. Roughly, these three epochs correspond to (1) the monkey having the hand on the starting pad, (2) the monkey moving and positioning the fingers on the object, and (3) the monkey holding the object. An analysis of variance was done of the firing rates during premovement, reach-prehension and grasp-hold. We tested the null hypothesis that states that the mean of the firing rates in each of the epochs are equal. We considered the neuron to be *task related* if the p-value of the F-test was lower than 0.001, which meant that the null hypothesis was rejected.

Linear model for kinematic synergies

We modeled the firing rate w_j of neuron j in terms of a linear relationship of the synergies $\mathbb{S}$ by

$$w_j = \mathbb{S} \cdot B_j + \epsilon_j \qquad \epsilon_j \sim \mathcal{N}(0, I \cdot \sigma_j^2). \tag{3.2}$$

For each monkey, we considered the firing rate of each neuron to be binned in non-overlapping time bins of length $\Delta = 70mS$ as in (Wu et al., 2003, 2002). The physical relationship between neural firing and

[2]Their analysis focused on the *simple* spikes of Purkinje cells as opposed to the *complex* spike type also characteristic of these types of neurons.

kinematic behavior implies the existence of a time lag between them (Moran and Schwartz, 1999). In order to address this neural-motor lag, we considered four non-overlapping bins backwards starting from the considered landmark (spanning a total period of $280mS$). Firing rate was obtained as sum of spike counts divided by the length of the bin. Then we fitted a model for each time lag using the static synergies obtained in the previous section at the two latest landmarks during the reach-and-grasp (namely, $t_{firstAft}$ and $t_{lastAft}$). In addition, we obtained static synergies on these same landmarks considering the velocity of the joint angles and the velocity of the 3D markers during the reach-and-grasp. For each synergy we selected the lag whose model yielded the largest mean 10-fold cross-validated *coefficient of determination* R^2. The larger this value, the better the synergy explains the firing rate.

Bootstrap to verify results are not due to chance

Finally, for each regression we performed a version of the bootstrap to verify that the results were not due to chance. Consider the matrix of kinematics $\mathbb{S} \in \mathbb{R}^{n \times p}$ that contains n trials, and perform the following procedure M times (in our case $M = 10$): draw a sample $\mathbb{S}_{res} \in \mathbb{R}^{n \times p}$, with replacement of size n from among the trials contained in $\mathbb{S}$; , perform the regression of w_j on $\mathbb{S}_{res}$, and save the resulting R^2. In this way, M coefficients of determination will be obtained, and the mean of them is an approximation of the R^2 obtained by chance. These values can be used to compare the R^2 obtained from the original regressions.

3.2.2 Data Analysis

Neural data. We analyzed sixteen sessions from Vinny and four from Baxter ; they contained 67 and 19 recorded neurons respectively. But from these neurons only 37 and 13 respectively fired in at least 100 trials and had, in average, at least ten spikes per replication. Therefore, we focused our analysis in those 37 and 13 neurons.

Static synergies. Table 3.4 shows the choices needed to build static synergies $\mathbb{S} \in \mathbb{R}^{n \times p}$. The number of columns p was determined by the number of components to be considered. When considering the low representation of the data we considered 2, 3, 5, 8 and 15 principal components for the unsupervised dimensionality dimension methods (PCA and kernel PCA), and the same number of components except for 15 for LDA. In the case of the original data, $p = 20$ for joint angles position and velocity, and $p = 48$ for 3D marker position and velocity. Ten fold cross validated multiple linear regression analysis was performed for each of the built synergies $\mathbb{S}$ paired with each of the vectors of neural activities. Since the neural activity was binned in intervals of $70mS$, the lag which yielded the largest 10-fold cross validated R^2 was selected.

Monkey	Kinematic variable	Time of interest	Synergy obtained with
• Vinny • Baxter	• joint angles • joint angles velocity* • 3D marker position • 3D marker velocity*	• *prehension* ($t_{firstAft}$) • *grasp-and-hold* ($t_{lastAft}$)	• PCA • kPCA polynomial kernel • kPCA Gauss kernel • LDA

Table 3.4: Summary of the choices made to build synergies $\mathbb{S}$ to investigate their linear encoding on neurons in M1. In the case of the Gaussian kernel, we considered $\sigma = 2$ as determined by the cross validated classification accuracy in Section 3.1.3. *Note that for velocity we considered a window of 40mS centered in the landmark and averaged the kinematic variables – otherwise the variables would have been zero due to the landmarks definition.

3.2.3 Results

Pre-selection of neurons. We pre-selected the neurons through the analysis described in Section 3.2.1. Thirty-one neurons from Vinny (83.78% of the total analyzed) were found to be *task related* ($p < 0.001$) and four neurons from Baxter (30.77% of the total analyzed). Table 3.5 shows the complete results.

Monkey	$p < 0.05$	$p < 0.01$	$p < 0.001$
Vinny	91.89% (34/37)	83.78% (31/37)	83.78% (31/37)
Baxter	38.46% (5/13)	30.77% (4/13)	30.77% (4/13)

Table 3.5: Number of task related neurons. We found evidence to reject the hypothesis that the firing rates in each epoch (premovement, reach-prehension and grasp-hold) are the same, with p-value of the F-test in the ANOVA being less than the specified value in the columns. Note that the two last columns look the same, but refer to different thresholds for the p-values.

Linear encoding of static synergies. We focused on the results from Vinny because there is a population of 31 neurons as opposed to only four from Baxter.

Linear encoding of joint angle kinematics. In Table 3.6 we show the mean R^2 values corresponding to the regression of selected neurons and the obtained synergies. We find that for these neurons (and many others) the non linear learned synergies better explain the firing rate of the collected neurons, especially for the variables corresponding to velocity. For some neurons, the joint angle synergies present a suggestive pattern: when the number of components is small, the linear and non linear learned synergies explain the firing rate very similarly; however, when considering more components, the non linear synergies are clearly better than the linear synergies.

We compared the linear coding of joint angles versus joint angles velocities. The number of neurons (out of 31) that yielded in average an R^2 higher than 0.15 are in Table 3.7. We contrasted synergies corresponding to two landmarks: $t_{firstAft}$ (prehension), and $t_{lastAft}$ (grasp-and-hold).

The number of neurons that yielded R^2 above threshold for joint angles was always higher than that for joint angle velocity except in three cases: LDA with 2 components at time prehenshion, kernel PCA

3D marker velocity

Baxter000467 - Neuron: spk002a		
Num. comp.	PCA	kPCA poly-2
2	0.19	0.28
3	0.23	0.32
5	0.25	0.34
8	0.27	0.37
15	0.31	0.54

Vinny000661 - Neuron: spk005a		
Num. comp.	PCA	kPCA poly-2
2	0.19	0.29
3	0.20	0.30
5	0.22	0.30
8	0.24	0.32
15	0.28	0.41

Joint Angles

Vinny000658 - Neuron: spk005b		
Num. comp.	PCA	kPCA poly-2
2	0.05	0.06
3	0.08	0.08
5	0.10	0.13
8	0.13	0.17
15	0.18	0.27

Vinny000693 - Neuron: spk001a		
Num. comp.	PCA	kPCA poly-2
2	0.16	0.18
3	0.20	0.21
5	0.20	0.23
8	0.23	0.25
15	0.30	0.36

Table 3.6: Examples of neurons and the mean R^2 from their 10-fold cross validated regression during prehension. In these examples non linear learned synergies explain better the firing rate of the collected neurons, especially for the variables corresponding to velocity. For some neurons, the joint angle synergies present a suggestive pattern: when the number of components is small, the linear and non linear learned synergies explain the firing rate very similarly; however, when considering more components, the non linear synergies are clearly better than the linear synergies.

(polynomial) with 15 components, and kernel PCA (gaussian) with 15 components. However, in these three cases the difference of number of neurons was only one. To contrast joint angles against joint angle velocities with respect to the number of neurons whose R^2 surpassed threshold, we calculated the average of the absolute value of the difference between the number of neurons. This value was 7.2 for grasp-and-hold, and 1.3 for prehension. The last value indicates that no matter what kinematic variable we consider (joint angle or joint angle velocity) the number of neurons whose firing rate can be explained (given our threshold) by a linear model of the kinematics is the same. In other words, at prehension there is the same number of neurons in our population that code for joint angle and for joint angle velocity.

We also contrasted the different dimensionality reduction methods (Table 3.7). We concluded that when considering joint angles, the behavior of *all* dimensionality reduction methods is very similar; such is also the case when considering joint angle velocities at prehension. However, when considering joint angle velocities at grasp-and-hold, LDA performs better, perhaps because at that point, the monkey is already grasping the object and so the information of the object that is definitely coded in the kinematics (since LDA is supervised) implies some specific firing pattern of the neurons.

Linear coding of 3D markers kinematics. Table 3.8 display the encoding results using 3D markers positions and velocities. Note that in this case, the design matrices consisted of 48 variables, as opposed to the 20 variables used in the joint angles case.

With 3D marker positions and velocities, the number of neurons whose regression yielded an R^2 higher than threshold were higher using 3D marker positions than 3D marker velocity except where a Gaussian kernel was used for the dimensionality reduction. There were also three other exceptions, but the difference

Prehension

JA position

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	11	(35.5)	10	(32.3)	12	(38.7)	7	(22.6)
3	11	(35.5)	11	(35.5)	12	(38.7)	9	(29)
5	12	(38.7)	12	(38.7)	14	(45.2)	12	(38.7)
8	17	(54.8)	15	(48.4)	18	(58.1)	17	(54.8)
15	24	(77.4)	24	(77.4)	23	(74.2)		

JA velocity

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	9	(29)	9	(29)	9	(29)	7	(22.6)
3	10	(32.3)	10	(32.3)	10	(32.3)	8	(25.8)
5	12	(38.7)	12	(38.7)	12	(38.7)	11	(35.5)
8	15	(48.4)	15	(48.4)	15	(48.4)	16	(51.6)
15	23	(74.2)	25	(80.6)	24	(77.4)		

Grasp-Hold

JA position

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	11	(35.5)	11	(35.5)	9	(29)	14	(45.2)
3	14	(45.2)	15	(48.4)	10	(32.3)	19	(61.3)
5	21	(67.7)	18	(58.1)	12	(38.7)	26	(83.9)
8	26	(83.9)	24	(77.4)	15	(48.4)	26	(83.9)
15	29	(93.5)	29	(93.5)	24	(77.4)		

JA velocity

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	0	(0)	0	(0)	0	(0)	7	(22.6)
3	2	(6.5)	2	(6.5)	7	(22.6)	13	(41.9)
5	10	(32.3)	10	(32.3)	10	(32.3)	18	(58.1)
8	21	(67.7)	20	(64.5)	21	(67.7)	24	(77.4)
15	27	(87.1)	26	(83.9)	26	(83.9)		

Table 3.7: Encoding of kinematic JA synergies. Number (and percentage) of neurons whose regression yields an $R^2 \geq 0.15$ considering joint angles and joint angles velocities as regressors. Each entry in the table (composed of two numbers – the count of neurons and the percentage) corresponds to one kinematic data set, a dimensionality reduction method, a specific number of components, and a specific point in time. For each dataset 10-folded cross validated regression was run on the firing rate of a specific neuron on different lags, the lag that yielded the highest R^2 in average was selected and its corresponding R^2 recorded.

was only of one neuron: LDA with 8 components at grasp-and-hold, and LDA with 8 components and PCA with 3 components at prehension. The average absolute difference in number of neurons was 7.55 for grasp-and-hold and 5.6 at prehension. Most of the difference was concentrated in the experiments where the Gaussian kernel was used to reduce the data.

Are these results due to chance? To verify whether the results were due to chance or not we performed a version of the bootstrap as explained in 3.2.1. The number of regressions to perform was large: $M \times numberNeurons \times numberConditions \times numberTimePoints \times numberLags \times numberKinematicVariables$ where M denotes the number of iterations of the bootstrap; $numberNeurons = 31$ for Vinny; $numberConditions = 19$ referring to dimensionality reduction method with specific number of components; $numberTimePoints = 2$ for prehension ($t_{firstAft}$), and grasp-and-hold ($t_{lastAft}$); $numberLags = 4$ for the different lags tried; and $numberKinematicVariables = 4$ referring to joint angles, joint angles velocity, 3D marker position and 3D marker velocity.

We defined $M = 10$. Thus, for each neuron, we obtained M values of R^2 (and for each of them we took the maximum R^2 across the four lags) and took its average. Then, we counted how many neurons yielded an R^2 higher than the defined threshold. Our bootstrap results show that it was never the case that the number of neurons yielding an $R^2 \geq 0.15$ through this procedure was higher than the number of neurons reported in Tables 3.7 and 3.8, and consequently, these results are not due to chance.

Prehension

3D position

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	9	(29)	10	(32.3)	0	(0)	10	(32.3)
3	9	(29)	13	(41.9)	1	(3.2)	10	(32.3)
5	17	(54.8)	17	(54.8)	2	(6.5)	12	(38.7)
8	21	(67.7)	21	(67.7)	3	(9.7)	15	(48.4)
15	26	(83.9)	24	(77.4)	7	(22.6)		

3D velocity

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	9	(29)	3	(9.7)	7	(22.6)	8	(25.8)
3	10	(32.3)	4	(12.9)	9	(29)	8	(25.8)
5	13	(41.9)	9	(29)	14	(45.2)	10	(32.3)
8	17	(54.8)	11	(35.5)	17	(54.8)	16	(51.6)
15	24	(77.4)	18	(58.1)	20	(64.5)		

Grasp-Hold

3D position

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	15	(48.4)	14	(45.2)	7	(22.6)	14	(45.2)
3	16	(51.6)	15	(48.4)	9	(29)	20	(64.5)
5	24	(77.4)	24	(77.4)	14	(45.2)	24	(77.4)
8	27	(87.1)	27	(87.1)	17	(54.8)	26	(83.9)
15	31	(100)	31	(100)	20	(64.5)		

3D velocity

Num. comp.	PCA		kPCA poly		kPCA Gauss		LDA	
2	9	(29)	3	(9.7)	3	(9.7)	10	(32.3)
3	14	(45.2)	6	(19.4)	8	(25.8)	14	(45.2)
5	18	(58.1)	9	(29)	16	(51.6)	24	(77.4)
8	21	(67.7)	18	(58.1)	23	(74.2)	27	(87.1)
15	27	(87.1)	23	(74.2)	29	(93.5)		

Table 3.8: Encoding of kinematic 3D synergies. Number (and percentage) of neurons whose regression yields an $R^2 \geq 0.15$ considering 3D position and 3D velocities as regressors. Each entry in the table (composed of two numbers – the count of neurons and the percentage) corresponds to one kinematic data set, a dimensionality reduction method, a specific number of components, and a specific point in time. For each dataset 10-folded cross validated regression was run on the firing rate of a specific neuron on different lags, the lag that yielded the highest R^2 in average was selected and its corresponding R^2 recorded.

3.2.4 Conclusions

The number of neurons to be analyzed is greatly reduced by the imposed requirements of having at least some minimum number of trials and of being *task related*. We argue that these constraints are necessary to ensure: (a) the reliability of the results of the linear regressions, and (b) that the signal we are analyzing is, in fact, related to the reach-to-grasp task. However, at the same time the requirements do drastically reduce the data we can analyze. For instance, for Baxter four neurons are actually insufficient to make any reliable inference of primary motor cortex neuron behavior. Nevertheless, the main result of this analysis is that we found some evidence that non linear synergies explain the firing pattern of some neurons in a better way than the linear synergies assuming a linear model.

3.3 Chapter summary and main contributions

In this work we analyzed the kinematics of the fingers of two monkeys (species *Macaca mulatta*) during a reach-to-grasp task. We considered two data sets: joint angles and 3D marker positions that describe the movement of the hand along time. We defined an energy function summarizing the amount of motion of the fingers. This function was useful to derive definitions for outliers, and to define time points of interest during the trials. We investigated low dimensional representations of the hand configurations at those specific time points during the reach-to-grasp movement. In order to perform the dimensionality reduction

we tried linear supervised (Linear Discriminant Analysis) and unsupervised (Principal Components Analysis) methods, and non linear unsupervised methods (kernel Principal Components Analysis with various choices of kernels). The low dimensional representations were then evaluated according to a classification task: predicting what object was being grasped at a specific trial. Two classifiers were trained and tested: a generative one (Naive Bayes) and a discriminative one (Support Vector Machines). The main finding was that under certain conditions, modelling the non linear relationships in the data resulted in better classifier performance.

The analysis of the low dimensional representation of the hand configurations was extended to investigate whether these representations were encoded in neurons recorded from the primary motor cortex (M1) of the monkeys. In order to perform the neural analysis, neurons were required to have a specific number of trials associated with them, and to be task related, as defined through differentiated firing rate patterns in different epochs during the reach-to-grasp trial (analysis of variance). We found that 84% of neurons of Vinny and 31% of neurons of Baxter were task related.

A classical multiple linear regression model was proposed to explain the firing rate of the neurons. The explanatory variables were the low dimensional representation of the kinematic variables of the fingers. The measure-of-goodness was defined to be the number of neurons that yielded an R^2 higher than a specific threshold. Using a variation of bootstrap in regression, results were verified to not be due to chance. We found that although the number of neurons that yield $R^2 \geq 0.15$ is similar in PCA and in kPCA with polynomial kernel, the actual value of R^2 is higher in the encoding of kPCA in some neurons. Therefore we found some evidence of better encoding of non linear synergies versus linear synergies.

Chapter 4

Dynamic synergies

In Chapter 3 we accounted for (or removed) time variation in the grasping dataset by identifying relevant time slices and analyzing the reach-to-grasp movements in those pre-defined landmarks. This approach allowed us to focus on finger and conditions variability, however it came at the cost of limiting the analysis to particular landmarks and disregarded finger variability in the rest of the trajectories.

An alternative strategy is to treat each observation as a multivariate functional observation and decompose the variation in a principled way. In this chapter we propose to explain out time variation of multivariate grasping trajectories by aligning (registering) the curves through a functional data analysis procedure. Then we show how to decompose and reduce the dimensionality of variation by adapting a Multivariate Gaussian Process (MGP) model, also known as the Gaussian Process Factor Analysis model (Yu et al., 2009). We set our model to decompose finger motion into two terms: a term that is shared among all replications of the same reach-and-grasp task and a term that is particular to each replication and that is modelled with a MGP. By fitting this model we also estimate dynamic and interpretable lower-dimensional representations of finger motion. In this chapter we discuss variants of our model, estimation algorithms, and we evaluate its performance in simulations and in data from the monkey Esteban introduced in Chapter 2. We also show that by taking advantage of the repeated trial structure of the experiments, our model yields an intuitive way to interpret the time and replication variation in our kinematic dataset.

The two main methodological contributions of our work include the alignment (or registering) of the collected multivariate grasping data and the decomposition and reduction of the dimensionality of the variation of the multivariate functional data according to the experimental structure: time, replication, and condition through the fitting of our Multivariate Gaussian Process Factor Model.

4.1 Introduction

In reach-to-grasp experiments, part of the variability may be understood as a result of the constraints among the fingers and this has led to the use of lower-dimensional representations known as *synergies* . Standard matrix factorization approaches, like principal components analysis (PCA), can go far in this direction (Santello et al., 1998; Todorov and Ghahramani, 2004; Mason et al., 2001, 2004; Soechting and Flanders, 1997; Pesyna et al., 2011; Thakur et al., 2008) but do not conform to the repeated-trial structure of most experiments and, furthermore, confound temporal variability with experimental condition and kinematic

variability.

There have been other approaches in the literature to obtain temporal grasping synergies. For instance, Vinjamuri et al. (2007, 2010a,b) inspired in (d'Avella and Bizzi, 2005) proposed two convolved-mixture models, which use SVD and an optimization step in their core, to learn a dictionary of time varying synergies. While the approach is able to describe time varying phenomena, it does not provide a generative model of grasping. State-space models are generative models that have been used to model dynamics of general body motion (Wang et al., 2008) albeit not finger dynamics. In these models, a Markovian assumption is posited and thus time correlations are unable to be directly captured. In contrast, the model we present in this chapter is not a state-space model — instead, we assume that the observed trajectories are generated via low-dimensional latent factor trajectories that are drawn directly from a GP, allowing for longer range correlations to be captured. Fyshe et al. (2012) also used this idea of modeling latent factor trajectories with GPs to analyze brain imaging data (MEG). Fyshe et al. (2012) assumed a loading matrix which changes over time as well as latent factors which are correlated through a hierarchical Bayesian prior, while our model follows the more traditional setting of latent factor analysis by assuming that the factors are independent and a stationary loading matrix.

We propose a Multivariate Gaussian Process (MGP) based model (similar to Yu et al. (2009)) which captures long range correlations and provides a generative framework for finger motion data. We set the MGP as a prior for a dynamic factor analysis model that allows for the learning of low dimensional spatio-temporal patterns (or *synergies*). In our model we decompose motion data into two terms: a term that is shared among all replications of the same reach-and-grasp task and a term that is particular to each replication and that is modelled with a MGP. We first derive an efficient inference algorithm that finds the *maximum a posteriori* trajectory in the latent low dimensional space from an observed multivariate 3D trajectory. And secondly, we derive an EM-based parameter estimation algorithm which, given 3D trajectories, outputs interpretable model parameters that can be used to explain the variability in the grasping task.

4.2 Model

Consider the $p-$dimensional observed dataset: $\{\ Y_i^r(t) \mid i = 1, \ldots, p;\ t = 1, \ldots, T;\ r = 1, \ldots, R\ \}$. $Y_i^r(t)$ is the i^{th} coordinate at time t of the p-dimensional trajectory that is the r^{th} replication of an event. Note that for simplicity of exposition, we consider only finitely many time points, but our model is based on Gaussian Processes and thus it applies to the continuous setting. R is the number of repeated trials, T the number of time slices and p the number of observed variables. In our application the observed variables describe the hand kinematics – they could be position, velocity, acceleration, joint angles or any function or representation of hand kinematics.

We define the Multivariate Gaussian Process Factor Model (MGPFM), which assumes:

$$\begin{bmatrix} Y_1^r(t) \\ \vdots \\ Y_p^r(t) \end{bmatrix} = \begin{bmatrix} \mu_1(t) \\ \vdots \\ \mu_p(t) \end{bmatrix} + \begin{bmatrix} \sum_{j=1}^d b_{1j} X_j^r(t) \\ \vdots \\ \sum_{j=1}^d b_{pj} X_j^r(t) \end{bmatrix} + \begin{bmatrix} \epsilon_1^r(t) \\ \vdots \\ \epsilon_p^r(t) \end{bmatrix}, \tag{4.1}$$

where $\mu_i(t) \quad i = 1, \ldots, p$ are deterministic mean functions, $\mathbf{B} = (b_{ij}) \in \mathbb{R}^{p \times d}$ is a deterministic factor loadings matrix, whose columns correspond to the d latent factors and rows correspond to the p observed

variables. Each latent factor trajectory X_j^r is drawn iid from an MGP with mean function $\mathbf{0}$ and covariance function $\sum(t_1, t_2)$ defined by $\sum(\cdot;\cdot) : [0,1] \times [0,1] \rightarrow \mathbb{R}$. And, finally, $\epsilon_i(t) \quad i = 1, \ldots, p$ are iid stationary MGP draws with covariance function $\Psi(t_1, t_2)$, which we assume to be diagonal in this work.

Letting $\mathbf{Y}^r(t) = \left[Y_1^r(t) \ldots Y_p^r(t)\right]^{\mathrm{T}}$, $\boldsymbol{\mu}(t) = [\mu_1(t) \ldots \mu_p(t)]^{\mathrm{T}}$, $\mathbf{X}^r(t) = [X_1^r(t) \ldots X_q^r(t)]^{\mathrm{T}}$ and $\boldsymbol{\epsilon}^r(t) = \left[\epsilon_1^r(t) \ldots \epsilon_p^r(t)\right]^{\mathrm{T}}$ we can write Equation 4.1 as:

$$\mathbf{Y}^r(t) = \boldsymbol{\mu}(t) + \mathbf{B}\mathbf{X}^r(t) + \boldsymbol{\epsilon}^r(t). \tag{4.2}$$

To ensure identifiability of the model we assume that the columns of $\mathbf{B}$ are orthogonal, that is: $\mathbf{B}^{\mathrm{T}}\mathbf{B} = \mathbf{I}_d$. Our model (Equation 4.2) decomposes each kinematic trial into a term $\boldsymbol{\mu}(t)$ that is common among replications and a term $\mathbf{X}^r(t)$ that is specific to the replication. The observed variables can represent any kinematic variables, such as 3D positions, velocities, accelerations or the corresponding joint angles. In our data analysis we model 3D marker velocities.

The parameter $\boldsymbol{\mu}(t)$ does not depend on the specific trial and can be modelled in two ways: invariant in time as a $p-$dimensional constant vector, and as a $p-$dimensional varying function in time. In the latter case $\boldsymbol{\mu}$ can be represented as a $p \times T$ matrix, or more efficiently, through a B-spline basis. The number of parameters to estimate for $\boldsymbol{\mu}$ is p when $\boldsymbol{\mu}$ is assumed to be constant, $p \cdot T$ when $\boldsymbol{\mu}$ is allowed to vary freely as $\boldsymbol{\mu} = \boldsymbol{\mu}(t) \in \mathbb{R}^{p \times T}$ and $\mathcal{O}(c \cdot T)$ when $\boldsymbol{\mu}$ is described through a B-spline basis with c the number of basis functions where $c << T$. This formulation drastically reduces the number of parameters to be estimated, while imposing a smoothing constraint on the learned functions, and it can be written as follows:

$$\mathbf{Y}^r(t) = \boldsymbol{\mu}^{\mathrm{S}}(t) + \mathbf{B}\mathbf{X}^r(t) + \boldsymbol{\epsilon}^r(t), \qquad \boldsymbol{\mu}^{\mathrm{S}}(t) = \boldsymbol{\alpha} \cdot (\mathcal{S}(t))^{\mathrm{T}} \in \mathbb{R}^{p \times 1}, \tag{4.3}$$

where $\mathcal{S} \in \mathbb{R}^{T \times c}$ is a matrix that holds the c spline basis functions, $\mathcal{S}(t)$ corresponds to one row of $\mathcal{S}$ and $\boldsymbol{\alpha} \in \mathbb{R}^{p \times c}$ contains the coefficients for each of the spline basis functions.

The parameter Σ corresponds to the covariance matrix of the MGP. The form of Σ determines the properties of the low dimensional kinematic representation. Estimating $\Sigma \in \mathbb{R}^{T \times T}$ implies learning $\mathcal{O}(T^2)$ parameters. While it is possible to estimate Σ *free* (which we do in some of our analysis), this procedure is prone to overfitting when there is not much data available or when the observed data was not exactly drawn from the model. One way to overcome this problem is to impose structure to Σ by assuming that it takes a parametric form. In this work, we use an stationary exponential covariance function: $\Sigma(i,j) = \exp\left(\frac{-(i-j)^2}{\theta_\Sigma}\right)$ where θ_Σ controls the width of the diagonal that decays exponentially. But other functions such as the Matern covariance function are also possible (Rasmussen and Williams, 2006). In our case, the exponential covariance function effectively imposes a prior belief that the latent trajectories are smooth, where θ_Σ controls how fast the function varies within a certain window of time.

4.2.1 Estimation

Our estimation algorithms are EM-based. We iterate between parameter estimation (learning) and inference on the latent variables as we explain below.

Loglikelihood

Denote $\vec{\boldsymbol{\mu}} = \{\boldsymbol{\mu}(t)\}_{t=1}^{T}$, $\mathbb{Y} = \left\{\{\mathbf{Y}^r(t)\}_{r=1}^{R}\right\}_{t=1}^{T}$ and $\mathbb{X} = \left\{\{\mathbf{X}^r(t)\}_{r=1}^{R}\right\}_{t=1}^{T}$. Then the joint distribution can be written as:

$$\mathbb{P}(\mathbb{Y}, \mathbb{X}|\vec{\boldsymbol{\mu}}, \mathbf{B}, \Psi, \Sigma) = \mathbb{P}(\mathbb{Y}|\mathbb{X}; \vec{\boldsymbol{\mu}}, \mathbf{B}, \Psi) \cdot \mathbb{P}(\mathbb{X}|\Sigma).$$

The loglikelihood consists of two terms:

$$log\,\mathbb{P}(\mathbb{Y}, \mathbb{X}|\vec{\boldsymbol{\mu}}, \mathbf{B}, \Psi, \Sigma) = log\,\mathbb{P}(\mathbb{Y}|\mathbb{X}; \vec{\boldsymbol{\mu}}, \mathbf{B}, \Psi) \; + \; log\,\mathbb{P}(\mathbb{X}|\Sigma). \tag{4.4}$$

If we simplify the model by defining $\Psi = \rho \cdot \mathbf{I}_{p\times p}$ with $\rho > 0$ then the first term corresponds to:

$$\begin{gathered} log\,\mathbb{P}(\mathbb{Y}|\mathbb{X}; \vec{\boldsymbol{\mu}}, \mathbf{B}, \rho) \\ = -\tfrac{1}{2}\sum_{r=1}^{R}\sum_{k=1}^{p}\sum_{i=1}^{T}\tfrac{1}{\rho}\left(\mathbf{Y}_k^r(t_i) - \left[\mu_k(t_i) + \sum_{w=1}^{d} b_{k,w}X_w^r(t_i)\right]\right)^2 \\ -\tfrac{R\cdot p}{2}\,T\cdot log\,\rho \; - \tfrac{1}{2}\,p\cdot R\cdot T\,log\,2\pi. \end{gathered} \tag{4.5}$$

The second term of the loglikelihood corresponds to the distribution of the d iid MGPs indexed by s (denoted $\mathbf{X}_s \in \mathbb{R}^{T\times 1}$) given the covariance function $\Sigma(t_i, t_j)$:

$$log\,\mathbb{P}(\mathbb{X}|\Sigma) = -\frac{1}{2}\sum_{r=1}^{R}\sum_{s=1}^{d} X_s^{r\,\mathrm{T}}\Sigma^{-1}X_s^r - \frac{1}{2}\,d\cdot R\,log|\Sigma| - \frac{1}{2}\,d\cdot R\cdot T\,log\,2\pi. \tag{4.6}$$

In sum, if we consider the covariance simplifications then the loglikelihood of the model is given by the sum of expressions in Equation 4.5 and 4.6. We can take Equation 4.4 and use it as a loss function to learn the components of the model. Our approach is EM-based, in which we iterate between:

1. *Estimation (learning) problem*: Assuming that $\mathbb{X}$ are known, learn parameters $\vec{\boldsymbol{\mu}} \in \mathbb{R}^{p\times T}, \mathbf{B} \in \mathbb{R}^{p\times d}, \rho \in \mathbb{R}, \Sigma \in \mathbb{R}^{T\times T}$ which jointly constitute the parameter space.
2. *Inference problem*: Assuming that $\mathbb{Y}$ and all the parameters known, estimate the latent variables $\mathbb{X}$.

Iterations are stopped either by convergence of the loglikelihood or by number of iterations. In the experiments we noted that 50 iterations sufficed to reach convergence. Our approach can be thought of as *Hard EM* – in conventional EM, one computes a *soft* posterior distribution in the E-step; in hard EM, we simply maximize the posterior. For example, K-means can be seen as the Hard EM based algorithm for fitting Gaussian Mixture Models.

Learning problem

In the first problem we assume that the latent space trajectories $\mathbb{X}$ are known and we estimate the parameters $\vec{\boldsymbol{\mu}}, \mathbf{B}, \rho, \Sigma$. Here we will be maximizing the loglikelihood with respect to the parameters.

Note that to learn $\vec{\boldsymbol{\mu}} \in \mathbb{R}^{p\times T}, \mathbf{B} \in \mathbb{R}^{p\times d}$, and $\rho \in \mathbb{R}$ we only need the first term of Equation 4.4, that is, Equation 4.5. And to estimate Σ we only need the second term of Equation 4.4, namely Equation (4.6).

To estimate Σ we consider Equation (4.6), let $\Omega = \Sigma^{-1}$ and perform standard optimization to learn the covariance function of the multivariate normal, obtaining:

$$\hat{\Sigma} = \frac{\sum_{r=1}^{R}\sum_{s=1}^{d} X_s^r \cdot X_s^{r\,\mathrm{T}}}{d\,R}. \tag{4.7}$$

Estimating ρ from Equation 4.5 is independent from estimating $\vec{\boldsymbol{\mu}}$ and $\mathbf{B}$. Differentiating Equation 4.5 with respect to ρ and equating to zero we obtain: $\hat{\rho} = \frac{\sum_{r=1}^{R}\sum_{k=1}^{p}\sum_{i=1}^{T}\left(\mathbf{Y}_k^r(t_i) - \left[\hat{\mu}_k(t_i) + \sum_{w=1}^{d}\hat{b}_{k,w}X_w^r(t_i)\right]\right)^2}{RpT}$.

Maximizing Equation 4.5 with respect to $\vec{\boldsymbol{\mu}}$ and $\mathbf{B}$ is equivalent to separately maximizing each term for a fixed $k = 1, \ldots, p$ and, in fact, corresponds to performing p multiple linear regressions. Consider the data vector $\mathcal{Y}_k$, the design matrix $\mathcal{W}$, and the variables to learn β_k defined as follows: ${\mathcal{Y}_k}^{\mathrm{T}} = \left[\left[Y_k^1(t_1), \ldots, Y_k^1(t_T)\right]^{\mathrm{T}}, \ldots, \left[Y_k^R(t_1), \ldots, Y_k^R(t_T)\right]^{\mathrm{T}}\right] \in \mathbb{R}^{1\times(R\cdot T)}$,

$$\mathcal{W} = \begin{bmatrix} \begin{bmatrix} X_1^1(t_1) & \ldots & X_d^1(t_1) \\ \vdots & \ddots & \vdots \\ X_1^1(t_T) & \ldots & X_d^1(t_T) \end{bmatrix} & I_{T\times T} \\ & \vdots \\ \begin{bmatrix} X_1^R(t_1) & \ldots & X_d^R(t_1) \\ \vdots & \ddots & \vdots \\ X_1^R(t_T) & \ldots & X_d^R(t_T) \end{bmatrix} & I_{T\times T} \end{bmatrix} \in \mathbb{R}^{R\cdot T\times(d+T)},$$

and

$$ {\beta_k}^{\mathrm{T}} = [b_{k,1}, \ldots b_{k,d}\,,\ \mu_k(t_1), \ldots, \mu_k(t_T)]^{\mathrm{T}} \in \mathbb{R}^{1\times(d+T)}. \tag{4.8}$$

Then, we can consider the p independent linear regression models:

$$\mathcal{Y}_k = \mathcal{W}\cdot\beta_k, \quad k = 1, \ldots,\ p. \tag{4.9}$$

By solving these p linear regressions we can estimate the vectors β_k, from which we can read off the desired model parameters $\vec{\boldsymbol{\mu}} \in \mathbb{R}^{p\times T}$ and $\mathbf{B} \in \mathbb{R}^{p\times d}$. In the case in which μ is assumed constant along time, the estimate corresponds to the mean across time of provided estimate.

Modelling $\boldsymbol{\mu}$ with splines. We have that the mean trajectory $\boldsymbol{\mu}$ can be written in a B-spline basis as follows:

$$\boldsymbol{\mu}^{\mathrm{S}}(t) = \boldsymbol{\alpha}\cdot(\mathcal{S}(t))^{\mathrm{T}} \in \mathbb{R}^{p\times 1}, \tag{4.10}$$

where $\mathcal{S} \in \mathbb{R}^{T\times c}$ is a matrix that holds the c spline basis functions (one in each column). Vector $\mathcal{S}(t)$ corresponds to one row of $\mathcal{S}$. We are interested in learning $\boldsymbol{\alpha} \in \mathbb{R}^{p\times c}$ which contains the coefficients for each of the spline basis functions and describes the data.

Our goal is thus to formulate similar models to Equation 4.9 for each variable $k = 1, \ldots, p$ but solving for the coefficients of the B-spline basis. Instead of defining the auxiliary variable β_k as in Equation 4.8, we define: $\phi_k = [b_{k,1}, \ldots, b_{k,d}, \alpha_k(1), \ldots, \alpha_k(c)]^{\mathrm{T}} \in \mathbb{R}^{(d+c)\times 1}$, where k denotes the index of an observed variable $k = 1, \ldots, p$ and $\alpha_k(i)$ denotes the coefficient of the i^{th} spline basis. Equation 4.10 can be written as $[\mu_k(t_1), \ldots, \mu_k(t_T)]^{\mathrm{T}} = [\alpha_k(1), \ldots, \alpha_k(c)]^{\mathrm{T}}\cdot\mathcal{S}^{\mathrm{T}} \in \mathbb{R}^{1\times T}$ and we want to determine the values for $\{\alpha_k(i)\}_{i=1}^{c}$ for each $k \in \{1, \ldots, p\}$. Note that $\beta_k = b_{k,1:d} \oplus \mu_k$ and $\phi_k = b_{k,1:d} \oplus \alpha_k$ where $\oplus$ denotes the stacking operation for vectors. Also $b_{k,1:d} = [b_{k,1}, \ldots, b_{k,d}]^{\mathrm{T}} \in \mathbb{R}^{d\times 1}$ corresponds to the k^{th} transposed row of the loading matrix $\boldsymbol{B} \in \mathbb{R}^{p\times d}$. Then: $\beta_k = (I_{d\times d} * b_{k,1:d}) \oplus (\mathcal{S}\cdot\alpha_k) = (I_{d\times d} \oplus \mathcal{S})\cdot(b_{k,1:d} \oplus \alpha_k) =$

$(I_{d\times d} \oplus \mathcal{S}) \cdot \phi_k$, and: $\mathcal{Y}_k = \mathcal{W} \cdot \beta_k = \mathcal{W} \cdot (I_{d\times d} \oplus \mathcal{S}) \cdot \phi_k$. Consequently, we have written an analogous problem as in Equations 4.9 to solve for the coefficients of the B-spline basis, namely:

$$\mathcal{Y}_k = \mathcal{W}^{\mathrm{S}} \cdot \phi_k, \qquad \mathcal{W}^{\mathrm{S}} = \mathcal{W} \cdot (I_{d\times d} \oplus \mathcal{S}) \tag{4.11}$$

for $k = 1, \ldots, p$. And the problem reduces to solve for ϕ_k in a similar way as before.

Constraining Σ. In the estimation procedure we can either learn Σ *free* as in Equation 4.7, or learn θ_Σ, the univariate parameter that determines the covariance function of the MGP: $\Sigma(i,j) = \exp\left(\frac{-(i-j)^2}{\theta_\Sigma}\right)$. The latter can be done by gradient descent (numerically maximizing the loglikelihood) or through a one dimensional search over a space of reasonable values for θ_Σ. In our implementation we follow the last strategy.

Inference problem

For the second problem (inference) we assume that the parameters $\mathbf{\Theta} = \{\vec{\boldsymbol{\mu}}, \mathbf{B}, \Psi, \Sigma\}$ are now known and we learn the hidden variables $\mathbb{X}$ by maximizing the posterior probability of $\mathbb{X}$ given $\mathbb{Y}$ and $\mathbf{\Theta}$. We observe that the vectorized elements of the latent factors for replication r (denoted by $\mathtt{vecX}^r \in \mathbb{R}^{(d\cdot T)\times 1}$) are distributed as $\mathcal{N}(\mathbf{0}, \Sigma \otimes I_{d\times d})$ where $\otimes$ denotes the Kronecker product of two matrices. Also, the vectorized difference of observed trajectory Y^r and mean μ given the latent factors of that replication are normally distributed:

$$(\mathtt{vecY}^r - \mathtt{vec}\mu)|\mathtt{vecX}^r \sim \mathcal{N}\left((I_{T\times T} \otimes \mathbf{B}) \cdot \mathtt{vecX}^r, \Psi \otimes I_{p\times p}\right).$$

Using standard properties of normal distributions we conclude that the posterior distribution of the latent factors given $\mathbb{Y}$ and $\mathbf{\Theta}$ is

$$\mathtt{vecX}^r|(\mathtt{vecY}^r - \mathtt{vec}\mu) \sim \mathcal{N}\left(\eta, \Lambda\right),$$

with:

$$\eta = \mathbf{0} + [(I_{T\times T} \otimes \mathbf{B}) \cdot (\Sigma \otimes I_{d\times d})]^{\mathrm{T}} \cdot \tag{4.12}$$

$$\left[(I_{T\times T} \otimes \mathbf{B}) \cdot (\Sigma \otimes I_{d\times d}) \cdot (I_{T\times T} \otimes \mathbf{B})^{\mathrm{T}} + (\Psi \otimes I_{p\times p})\right]^{-1} \cdot$$

$$(\mathtt{vecY}^r - \mathtt{vec}\mu - \mathbf{0}),$$

and

$$\Lambda = [\Sigma \otimes I_{d\times d}] -$$

$$[(I_{T\times T} \otimes \mathbf{B}) \cdot (\Sigma \otimes I_{d\times d})] \cdot \left[(I_{T\times T} \otimes \mathbf{B}) \cdot (\Sigma \otimes I_{d\times d}) \cdot (I_{T\times T} \otimes \mathbf{B})^{\mathrm{T}} + (\Psi \otimes I_{p\times p})\right]^{-1}$$

$$\cdot [(I_{T\times T} \otimes \mathbf{B}) \cdot (\Sigma \otimes I_{d\times d})]^{\mathrm{T}}.$$

The mean of a normal distribution maximizes the loglikelihood, therefore we set: $\hat{X} = \eta$. Note that the matrix we need to invert in this step is sparse and contains a lot of structure that we can exploit to make computation efficiently. In particular, in Equation 4.12, η is the product of two big matrices U and V and a vector w. Both U and V are sparse and we do not need to fully invert V, we only need to compute $V^{-1} \cdot w$. Hence, we can use sparse matrices to represent U and V, and we can efficiently calculate $V^{-1} \cdot w$ without explicitly inverting the matrix using a sparse linear solver. In addition, matrices U and V are Kronecker products with the identity matrix, which is itself sparse, and thus we can represent it efficiently computationally.

Initialization

Since the learning algorithm is an EM procedure, it is susceptible to local optima. To avoid getting stuck in local optima we propose two initialization methods. The first one (*MLE*) involves estimating the MLE for a matrix normal distribution (Dawid, 1981; Dutilleul, 1999), and the second (*down-projection*) estimating the MGPFM parameters assuming a higher latent dimension and projecting down to the desired dimension.

Matrix normal MLE based initialization. A matrix $U \in \mathbb{R}^{p\times T}$ is said to be sampled from a matrix normal distribution $\mathcal{N}_{p,T}(M, F, G)$ with mean $M \in \mathbb{R}^{p\times T}$, among-row covariance matrix $F \in \mathbb{R}^{p\times p}$ and among-column covariance matrix $G \in \mathbb{R}^{T\times T}$ if its vectorized form $\texttt{vecU}$ is distributed as the multivariate normal: $\mathcal{N}_{p\cdot T}(\texttt{vecM}, F \otimes G)$. Conceivably the observed data generated with the MGPFM can be close to a matrix normal distribution or, at least, we can use this distribution for initialization purposes (for more details of the matrix normal distribution see (Dawid, 1981)). There are no analytical solutions for the MLE for the among-row and among-column covariance matrices of the matrix normal distribution. However, Dutilleul (1999) presents an iterative algorithm (also called *flip-flop* algorithm) to obtain the MLE of its three parameters (M, F, G). We propose to initialize the parameters of the MGPFM as follows: $\mu_0 = \tilde{M}$ and $\Sigma_0 = \tilde{G}$, where $\tilde{}$ denotes the MLE. To initialize $\mathbf{B}_0$ and ρ_0 we obtain the spectral decomposition of $\tilde{F}$. Intuitively, $\mathbf{B}_0$ contains the first d normalized eigenvectors of $\tilde{F}$ and ρ is the residual obtained by subtracting $\mathbf{B}_0 \cdot \mathbf{B}_0^{\mathrm{T}}$ from $\tilde{F}$. Let $D \in \mathbb{R}^{p\times p}$ be the diagonal matrix containing the decreasing eigenvalues of $\tilde{F}$ and let $E \in \mathbb{R}^{p\times p}$ contain in its columns the corresponding eigenvectors. We set $\mathbf{B}_0 = E_{:,1:d} \cdot \sqrt{D_{1:d,1:d}}$, and $\rho_0 = \sqrt{\frac{\sum_i \sum_j \check{e}_{i,j}^2}{p}}$ where $\check{e}_{i,j}$ is the (i,j) element of the matrix $\check{E}$ defined as $E_{:,d+1:p} \cdot \sqrt{D_{d+1:p,d+1:p}}$.

Down-projection based initialization. In this second initialization approach we want to learn the model parameters when the latent dimension is d_{goal}, but begin with a higher dimensional problem.

1. We first run the MGPFM learning algorithm (as before) for a latent dimension d_{high} higher than desired i.e. $d_{goal} < d_{high}$ and obtain as an output the estimates: $\hat{\mu}_h \in \mathbb{R}^{p\times T}, \hat{\rho}_h \in \mathbb{R}, \hat{\Sigma}_h \in \mathbb{R}^{T\times T}$, and $\hat{\mathbf{B}}_h \in \mathbb{R}^{p\times d_{high}}$.
2. Project the estimated parameters to the target latent dimension d_{goal}. We note that only $\hat{\mathbf{B}}_h$ needs to be projected. We use the SVD decomposition of the matrix: $\hat{\mathbf{B}}_h = U \cdot S \cdot V^{\mathrm{T}}$ and define $\mathbf{B}_{proj} = \hat{\mathbf{B}}_h \cdot V_{:,1:d_{goal}}$.
3. Use the projected estimates as initial values for a second run of the MGPFM learning algorithm, that is, set: $\mu_0 = \hat{\mu}_h$, $\rho_0 = \hat{\rho}_h$, $\Sigma_0 = \hat{\Sigma}_h$ and $\mathbf{B}_0 = \mathbf{B}_{proj}$.

We find in practice that this second initialization method is often effective but also significantly more computationally expensive since it requires optimization in a higher dimension first.

4.3 Alignment procedure

In functional data there are two types of variability: amplitude variation and phase variation (Ramsay et al., 2009; Ramsay and Silverman, 2005). Amplitude variation describes the variability of the sizes of the features of the kinematic curves; features such as the height of peak velocities of different markers recording finger

movement during a grasping task. On the other hand, phase variation describes the variability of the timings of the features of the kinematic curves, such as, the variation between the timing of the opening and closing or between the peak velocities of the fingers.

We are interested in studying amplitude variations in order to understand how the movement of the fingers relate to one another. To isolate the amplitude variation from the phase variation we can transform the time-axis for each trial so that phase variation between kinematic curves is minimized.

The original kinematic marker positions can be represented as $\mathbf{Y}^r(t) \in \mathbb{R}^p$ where $p = 3 * K$ and $K = 16$ is the number of markers placed on the fingers (K is multiplied by the three because of the 3-dimensional positions of the markers). Alignment of the curves is accomplished by estimating monotonically increasing *time warping functions* $h^r(t)$ such that the phase variation of $t \to \mathbf{Y}^r(h^r(t))$ is minimized across trials r. Note that it is important that the same function be used across kinematic variables for a fixed condition and trial, because we want to preserve the relationships between kinematic variables at fixed times.

Total energy signal. Alignment of multivariate curves is greatly simplified by summarizing each multivariate curve by a univariate curve. We will summarize the trials based on the *total energy signal* (already introduced in Chapter 3). The basic idea is that the velocity of the markers typically has clear peaks, valleys and zero crossings – features that are easily identified. In this subsection we slightly overload notation, letting $\mathbf{Y}^r(t)$ denote the $K \times 3$ *matrix* containing the 3-dimensional measurements of the K kinematic variables and $\dot{\mathbf{Y}}^r(t)$ the corresponding velocities. Thus $\mathbf{G}^r(t) = [\dot{\mathbf{Y}}^r(t)][\dot{\mathbf{Y}}^r(t)]^{\mathtt{T}}$ is the matrix of inner products of marker velocities for each replication r in a specific condition, and the sum of the squared magnitudes of the velocities across markers is:

$$\mathbf{E}^r(t) = \mathtt{tr}(\mathbf{G}^r(t)) = \mathtt{tr}([\dot{\mathbf{Y}}^r(t)][\dot{\mathbf{Y}}^r(t)]^{\mathtt{T}}). \tag{4.13}$$

$\mathbf{E}^r(t)$ is an important property of the trial because it summarizes the magnitude of motion during a trial and condition. Our goal is to estimate time warping functions $h^r(t)$ such that the phase variation of $t \to \mathbf{Y}^r(h^r(t))$ is minimized across trials. One of the benefits of this signal is that it is invariant under rotations of the 3-dimensional variables.

We estimate the time warping functions by minimizing the `MINEIG` criterion iteratively. The main idea is to choose a warping function for each trial such that the shape of the warped energy is close to the shape of the mean energy across replications of the same condition. In Figure 4.1 we show raw energy profiles and their aligned versions.

The MINEIG criterion. Estimation of the time warping function is explained in (Ramsay and Silverman, 2005) and is based on the minimizing criterion

$$\mathtt{MINEIG}(h) = \alpha_2 \cdot \det \begin{pmatrix} \int E_0(t)^2 dt & \int E_0(t) E^r(h(t)) dt \\ \int E_0(t) E^r(h(t)) dt & \int E^r(h(t))^2 dt \end{pmatrix}, \tag{4.14}$$

where $E_0(t)$ is the target and α_2 is the size of the second smallest eigenvalue of the enclosed matrix. The basic idea is that the matrix is like the covariance matrix of $(\{E_0(t) : t\}, \{E^r(h(t)) : t\})$. If one of the curves is exactly proportional to the other then the matrix is singular and so $\mathtt{MINEIG}(h) = 0$. The advantage of using this criterion is that there are well developed R and Matlab packages (`fda`, Ramsay et al. (2009)) for minimizing the roughness penalized criterion:

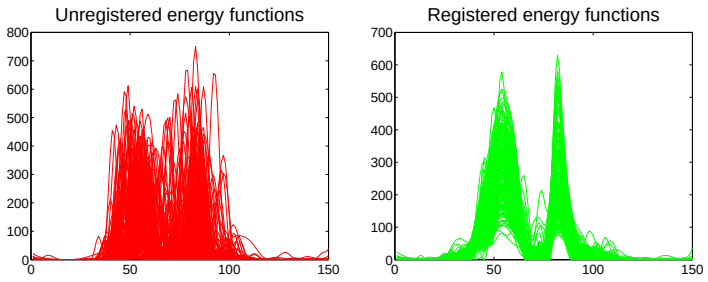


Figure 4.1: Example of energy profiles raw and aligned (Small cone, 45° adduction). On the x-axis we show time and on the y-axis the value of the energy functions.

$$\texttt{MINEIG}(h) + \lambda \int \{W^{(m)}(t)\}^2 dt, \tag{4.15}$$

when h is of the form: $h(t) = C_0 + C_1 \int_0^t \exp W(u)du$, with W expanded into the B-spline basis. The basic strategy for alignment then follows the iterative procedure known as *Procrustes method*:

1. Initialize the target $E_0(t) \leftarrow E^r(t)$ to some $E^r(t)$.
2. Repeat until convergence:
   (a) For each trial $r = 1, ..., R$ fit a time warping function $h^r(t)$ using the criterion given in Equation (4.15)
   (b) Update the target $E_0(t) \leftarrow \frac{1}{R}\sum_r E^r(h^r(t))$.

Recovering the aligned kinematic curves. Having estimated $h^r(t)$, the aligned velocity curves are $\dot{\mathbf{Y}}^r(h^r(t))$, while the positional curves can be obtained by integration:

$$\mathbf{Y}_0 + \int_0^t \dot{\mathbf{Y}}^r(h^r(u))du. \tag{4.16}$$

4.4 Simulation studies

We performed several simulations to study the behavior of the MGPFM when varying the number of samples, the size of the latent dimension and the various learning settings. Note that in the simulation studies there is no need of alignment.

We generated $p = 50$ dimensional data from a latent process of dimension $d_{true} = 4$. The dimensionality p of the observed simulated data roughly corresponds to the grasping data analyzed in further sections. We considered $T = 51$ time points and set $\mu \in \mathbb{R}^{p\times T}$ deterministically as a sinusoidal function of time with different amplitudes per coordinate: $\mu^k(t) = \sin\left(\frac{2\cdot k}{p} \cdot (t-k)\right), \quad k = 1, \ldots, p$; the entries of $\mathbf{B} \in \mathbb{R}^{p\times d}$ were drawn iid from $\mathcal{U}(0,1)$; we set $\rho = 0.25$; and assumed $\Sigma(i,j) = \exp\left(\frac{-(i-j)^2}{\theta_\Sigma}\right)$ with $\theta_\Sigma = 0.01$. Note that in the simulation studies no alignment is necessary.

Measures of goodness of fit. We summarize the mean square error (MSE) of observation r at a particular time slice t as:

$$e_r(t) = \frac{1}{p}\sum_{i=1}^{p}\left(Y_i^r(t) - \hat{Y}_i^r(t)\right)^2, \tag{4.17}$$

and the mean integrated square error (MISE) of observation r along time as:

$$\mathrm{e}_r = \frac{1}{T}\sum_{t=1}^{T} e_r(t). \tag{4.18}$$

These statistics summarize the reconstruction error. Also, in a specific simulation S we obtain the mean error in the simulation as:

$$\mathcal{E}_S = \frac{1}{R_S}\sum_{r=1}^{R_S}\mathrm{e}_r \tag{4.19}$$

where R_S is the number of observations in simulation S. We report the average of $\mathcal{E}_S$ across independent simulations S, and we also report its corresponding standard error.

Number of required training samples. We explore the question of how many training examples are required to achieve a certain performance and we compare the differences between modelling μ free and with splines with initialization through down-projection, keeping Σ free. We kept the latent dimension d fixed at $d_{true} = 4$, we generated a single test set of size 500, and ten different training sets for each specific number of training examples. Figure 4.2 shows that in every case, performance improved in terms of reconstruction error as the amount of training data increased. We also notice that after 40 examples the performance levels off. There is no clear difference between modelling μ *free* (as a matrix in $\mathbb{R}^{p\times T}$) and μ with splines, but considering the number of parameters to be estimated, the best performance is achieved with modelling μ as splines.

Latent dimension and reconstruction error. In the second simulation we study the behavior of different models when varying the value of the latent dimension. We consider three ways of modelling μ: as a *constant* across dimensions but varying along time; *free* (as a matrix in $\mathbb{R}^{p\times T}$) and modelling it with $B-$splines. We also consider two types of initialization: the matrix-normal MLE and the down-projection of a solution from a higher dimension. We investigate the performance in terms of reconstruction error, and we study whether the model and learning procedure are able to recover the true dimensionality of the data. We set apart a single test set of 500 samples and 10 training sets of size 20 for each value of d. We considered a training set of size 20 because this number corresponds to the number of samples of a specific condition in a session of the grasping dataset for the third monkey (see Chapter 2).

In Figure 4.3 we display the log-likelihood, the average MISE on the test set and the Bayesian Information Criterion (BIC) on the test set for each of the considered models, learning settings and for various values of the latent dimension.

In terms of MISE, modelling μ as a constant results in the worst performance. In contrast, modelling μ free yields much better results in both initialization regimes. We expected that modelling μ with B-splines

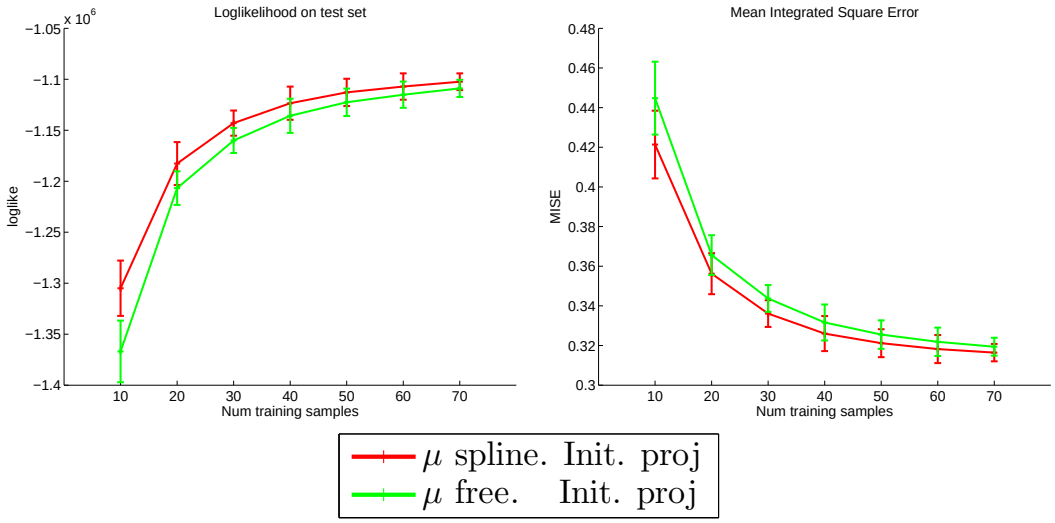


Figure 4.2: MGPFM: Loglikelihood and average MISE in the test set as a function of number of training samples. The figures show the results on a single train set of 500 samples using ten independent training sets of size 80. We report the mean error in the test set for the ten simulations $\left(\frac{1}{10}\sum_{i=1}^{10}\mathcal{E}_{S_i}\right)$ and its standard error. The dimensionality of the observed data was $p = 50$ and the latent dimension $d_{true} = 4$. We modelled Σ free, the μ with splines and free, and initialized the learning algorithm with the down-projection.

was as good as modelling μ free or even better, because constraining the number of parameters to learn would help when there is not a lot of training data. And indeed, when μ is modelled with B-splines and initialized through the projection strategy we get the best results in terms of MISE. However, the initialization regime played a bigger role than in the free setting (Figure 4.3) suggesting that in the more constrained case the algorithm is more susceptible to local optima and requires smarter initialization.

The true latent dimension is recovered through the BIC whenever the learning method is initialized through the projecting procedure, and sometimes with the MLE initialization. In all cases, the BIC is characterized by a very fast drop until reaching the true value of the latent dimension d, and the steep decrease is followed by either a slower decrease or slight increase in the BIC. The clearest case occurs when modelling μ with splines and in performing initialization through the projection procedure. In the supplementary material we show how the model works in one run of the simulation.

In conclusion, we showed that for data simulated from the model, we are able to recover the latent dimensionality, and that best performance comes from modelling μ with splines and initializing with the down-projecting strategy. However, given the computational cost, and that not much accuracy is lost, initializing the algorithm with the MLE of the matrix normal distribution is a recommendable strategy.

4.5 Data analysis

We analyzed the 48-dimensional finger motion captured data from 23 sessions of a rhesus monkey performing the reach-and-grasp task explained in Section 4.1. For the analysis we considered five conditions: small

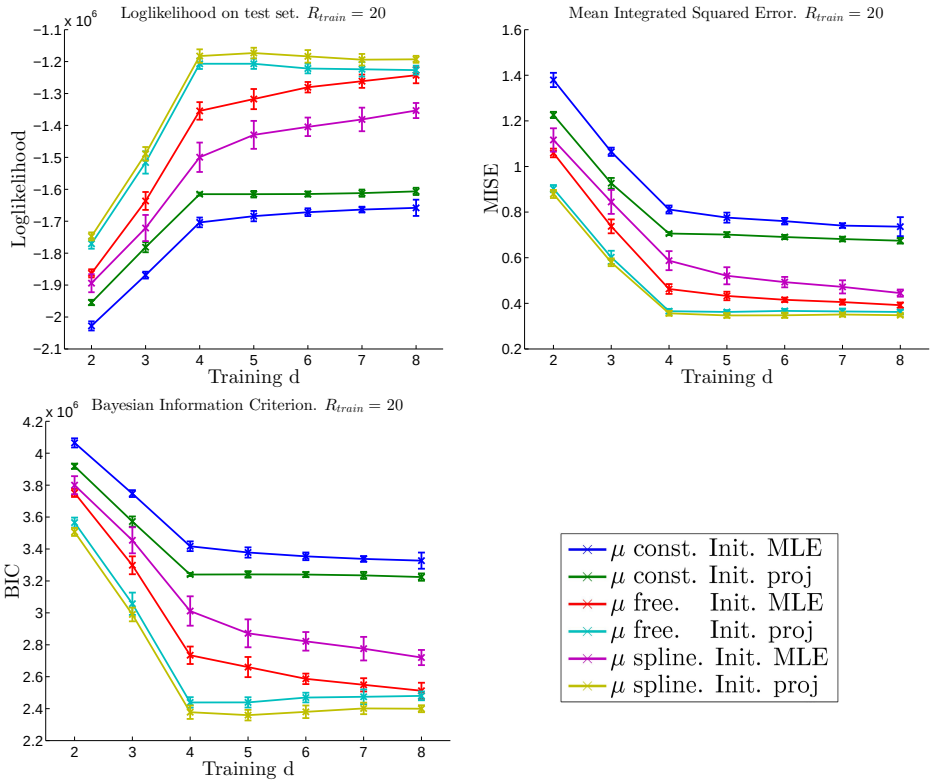


Figure 4.3: MGPFM: Average MISE in the test set, likelihood and BIC for different values of the latent dimensionality when varying the initialization regimes and the ways of modelling μ. The true latent dimension $d = 4$ is most obviously recovered when μ is modelled with splines and the initialization is through the projection procedure. When μ is modelled with splines the initialization regime has a large impact in performance.

cone and small handle presented in positions 45° of flexion and 45° of adduction, and for the small cone also at 45° of abduction. In each condition we considered all trials across sessions, totalling in average 155 trials per condition.

We performed the analysis on the 3D velocity profiles, because velocity and speed of hand movements are thought to be encoded in the activity of single motor cortical neurons (Moran and Schwartz, 1999; Wang and Moran, 2007). Note that performing similar analysis with joint angle velocity is also possible (Vinjamuri et al., 2007, 2010a,b).

We denote the observed velocity profiles as $\dot{\mathbf{Y}}^r(t)$ and fit the following model where $\boldsymbol{\mu}$ is modelled through splines:

$$\dot{\mathbf{Y}}^r(t) = \boldsymbol{\mu}^{\mathbb{S}}(t) + \mathbf{B}\dot{\mathbf{X}}^r(t) + \boldsymbol{\epsilon}^r(t). \tag{4.20}$$

Each reach-and-grasp replication lasted an average of 1.13 seconds, but each trial was of different length. In order to make it comparable, we smoothed data with a B-spline basis, resampling all trials onto 150 time slices.

For outlier removal we summarized each trial with its *energy function* (Equation 4.13) and clustered trials of a specific condition using the same function via $k-$means. We applied the clustering algorithm for several values of k (2, 3 and 4) and aggregated the resulting clusters removing the smallest group that contained at most 10% of trials and whose removal yielded the most visually uniform set of energy profiles. The MGPFM was applied to the preprocessed raw data and to aligned data (as explained in Section 4.3). Figure 4.1 shows an example of the same data in these two states.

Analysis of performance varying models, alignment and latent dimensionality. We compared the performance of the MGPFM to two different baselines in terms of the MISE (Equation 4.18). We first compared our model against a *simple* baseline, namely modelling the data as a time varying mean; and secondly, we compared MGPFM against PCA, the prevailing approach in literature. In addition, we investigated whether aligning the data had an impact on the performance of the models in terms of MISE. Finally, we investigated the impact of varying the size of the latent dimension. We modelled μ with B-splines, Σ constrained and we initialize the model with matrix normal MLE.

Note that to apply PCA, we stacked the training trajectories into a 2D matrix of size $(R \cdot T) \times p$ where R is the number of training samples, $T = 150$ and $p = 48$ denotes the number of kinematic variables. In applying this methodology we disregard time correlations (as is usual in conventional PCA), and for this reason it does not make sense to align data before applying PCA.

We considered different conditions: each condition defined by the object and the orientation in which it was presented. We obtained the 10-fold cross validated MISE for the preprocessed raw data modelled with the mean, with PCA and with the MGPFM. We did the same with the aligned data modelled with the mean and with the MGPFM.

Figure 4.4 shows a representative example of the results of this experiment. Regardless of alignment, PCA or the MGPFM considerably outperform the mean. However the alignment helps in every setting including the mean, PCA and MGPFM, and it gives a significant improvement in MISE particularly when modelling the data only with the mean. Finally, MISE decreases in every case as the latent dimensionality increases. And we notice that the impact that alignment has on the reduction of MISE is greater when the latent dimensionality is lower.

Results of a specific condition. We now discuss in some detail the results of estimating the model parameters for the small cone presented with 45° of abduction. We use the pre-processed and aligned data from all sessions (165 trials) and present the results of one of the ten folds (with 149 replications for training, and 16 for test). We modelled $\boldsymbol{\mu}$ with B-splines, Σ constrained and we initialized the model with the matrix normal MLE. For simplicity of interpretation, we set the latent dimensionality to be two.

In Figure 4.5 we plot the error in two ways: first, as a function of time, and second, integrated across time for each replication. We compare the MSE of MGPFM against a baseline of modelling only the mean. The top two plots show that there is much more variation during certain time periods in the trials and that the baseline is unable to capture this variation. The MGPFM significantly reduces the error (by approximately an order of magnitude) in those time periods by capturing the variation between trials.

In Figure 4.6 we show the observed velocity profiles, the MGPFM estimates and the residuals decomposed per marker and finger for a specific replication for the small cone presented at 45° abduction. These plots, which are representative of the grasping behavior in the dataset, show that the thumb's amount of

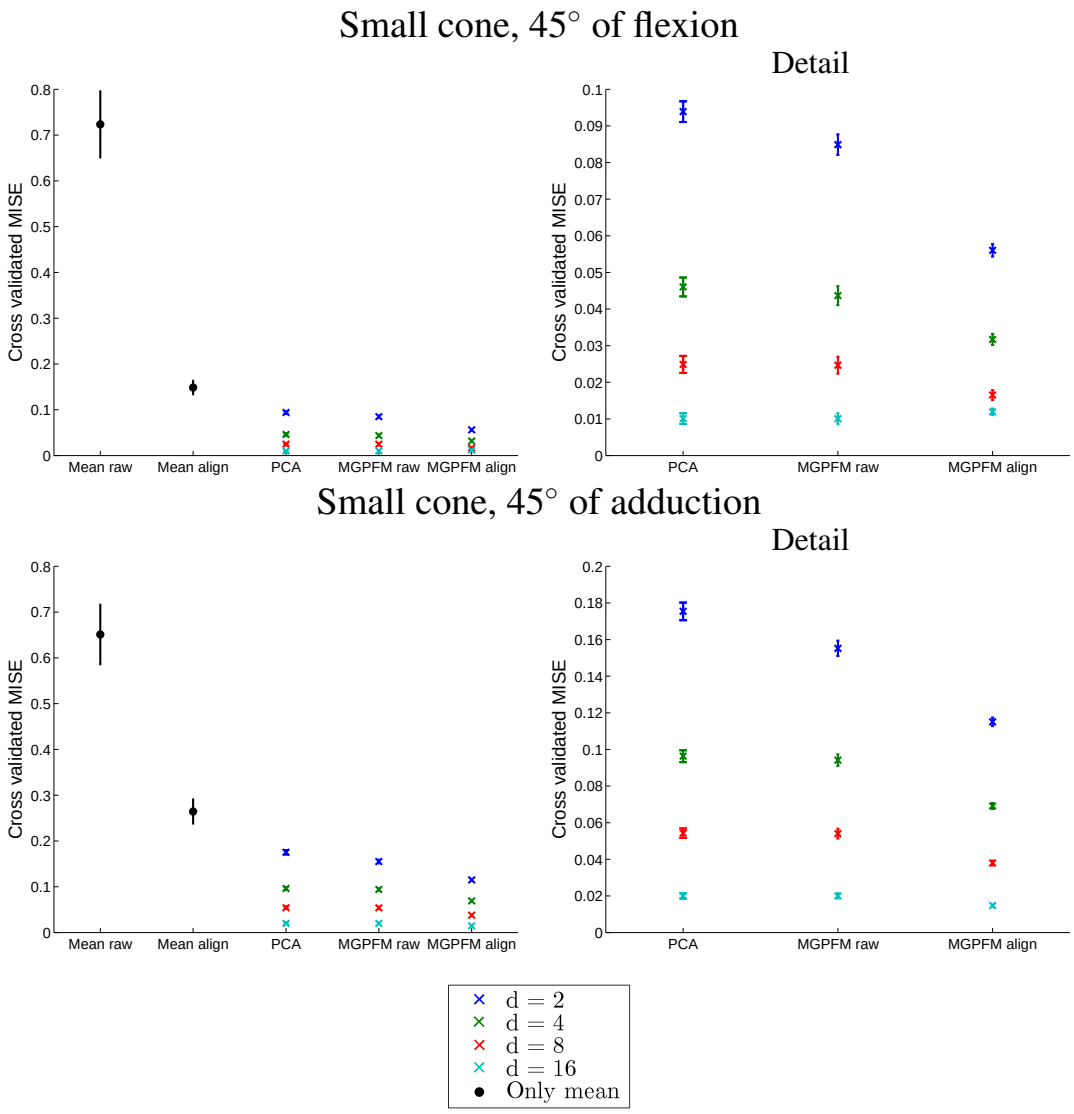


Figure 4.4: MGPFM: we show 10-fold cross validated MISE in grasping data for two conditions (small cone flexion and adduction). Note that we are not plotting standard errors as we are not dealing with independent replications of the experiment, but with folds; we show the average standard deviation as an indication of the spread of the error in different folds. We compare the baseline model considering only the mean, PCA, MGPFM on raw pre-processed data and MGPFM on aligned data. Experiments were run for various sizes of latent dimensionality d. The MGPFM was applied modelling $\boldsymbol{\mu}$ with splines, Σ constrained, initializing with the MLE of the matrix normal distribution and 50 iterations of the learning procedure. Observe that the MGPFM applied on aligned data achieves better results than other methods, but its advantage decreases as the size of the latent dimension increases.

movement is very small as compared to the amount of movement by all the other fingers. In other conditions (like the small handle) this is also the case, but the contrast is particularly prominent with the middle,

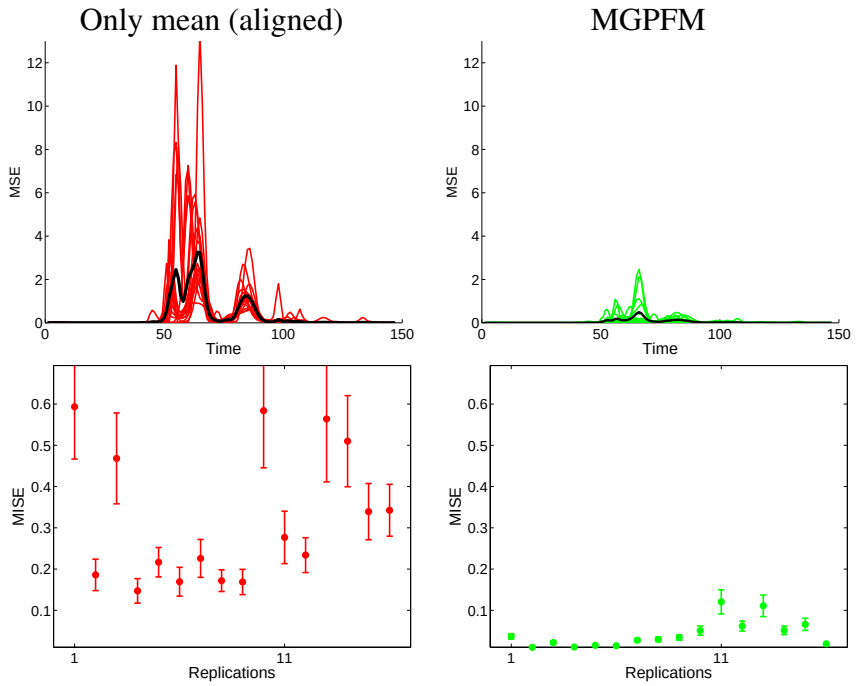


Figure 4.5: Error profiles in real data (small cone, 45° abduction). In the upper panel, each line corresponds to a replication and the black line is the mean value. The lower panel displays the mean integrated square error per replication ($\pm$ standard deviation of the error along time on the ten folds). The MGPFM reduces the error significantly, as compared to the baseline of modelling only the mean on aligned data (by approximately an order of magnitude).

ring and index fingers. The MGPFM captures most of the variation leaving residuals close to zero.

Interpretation of learned parameters. One of the main features of the MGPFM is that its parameters can be interpreted. Parameter $\boldsymbol{\mu}$ is a trajectory in the velocity space, and through Equation 4.16 we are able to obtain corresponding postures in the position space. In Figure 4.7 we show time slices of $\boldsymbol{\mu}$ projected onto the hand space that summarize the main features of the $\boldsymbol{\mu}$ trajectory. The mean parameter $\boldsymbol{\mu}$ captures the shared behavior across all trials. In this particular condition, this behavior consists of five *epochs* that correspond to the hand starting in a neutral position, followed by a slight opening of the fingers in a synchronized fashion, back to a neutral position, after which the subject spreads his fingers slightly before going back to neutral position. All trials in this condition show this pattern.

In contrast to the parameter $\boldsymbol{\mu}$ that encodes behavior shared among all replications, the term that includes the latent factors X and the loading matrix $\mathbf{B}$ corresponds to the ways in which a replication differentiates itself from the other replications. The factors X encode what is specific to each trial. Figure 4.8 shows X for two example replications superposed on the distribution of factors for all replications. The factors for these two representative replications differ, and to understand how these differences are manifested in the grasp trajectory we can visualize the columns of the loading matrix $\mathbf{B}$ which encode the marker movement as a function of the latent factors. Figure 4.9 shows both columns of $\mathbf{B}$ as a set of $K = p/3 = 16$ vectors

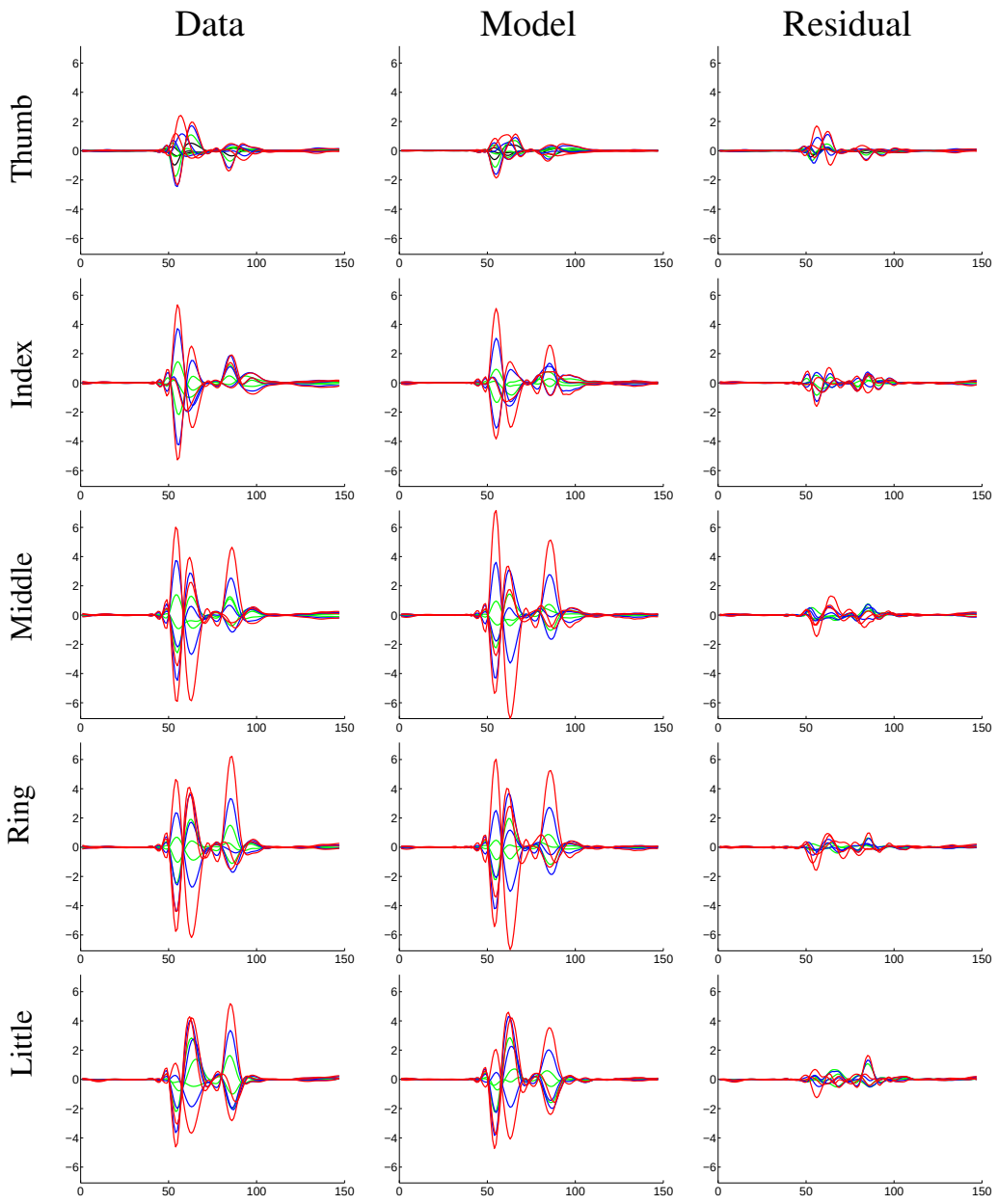


Figure 4.6: Observed velocity profiles, MGPFM fit and residuals for one replication of the subject grasping the small cone at 45° of abduction. Each row represents a finger and the trajectories corresponding to a specific marker are plotted in the same color: red for the most distal marker, green for the most proximal marker and blue for the maker in the middle. The thumb has a fourth marker plotted in black. The magnitude of motion in the thumb markers is very small as compared to the other fingers. The MGPFM estimates are very close to the observed data yielding residuals close to zero.

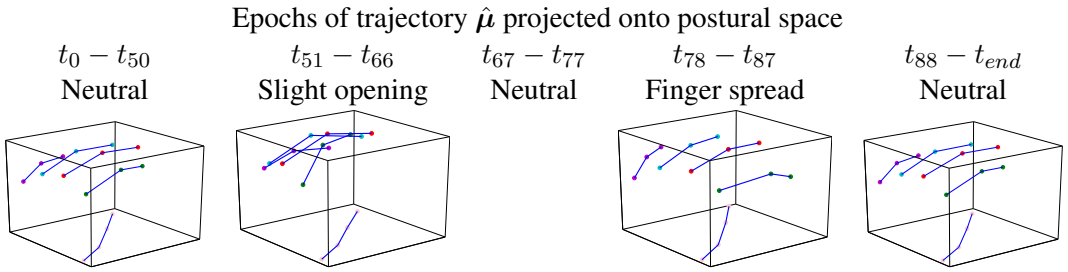


Figure 4.7: The trajectory $\hat{\boldsymbol{\mu}}$ (projected onto the position space) represents what is shared by all replications of a condition. Here we show the visualization of $\hat{\boldsymbol{\mu}}$ for the small cone at 45° of abduction. The trajectory presents five *epochs* corresponding to hand postures. It begins at a neutral position, followed by a slight opening of the grasp through a synchronized movement of fingers and a slight rotation, then back to the neutral hand configuration after which the fingers spread slightly (after the subject releases the object) and back to a neutral position. All replications for this condition follow this pattern; and they differentiate among themselves with the movement modelled through the loading matrix and the MGP term.

that correspond to the direction and relative magnitude of change for each of the p markers.

The first loading, or first column, of **B** (left panel) encodes a synchronized closing-opening motion of the fingers; whereas the second loading, or second column, of **B** (right panel) encodes a movement that happens at a somewhat different rate and direction per finger, whose net effect captures a wrapping (or curling) of the fingers (as if around the cone). These two loadings represent the ways in which trials are different among each other in the dataset, namely that trials differ amongst each other in the magnitude of the grasp opening and the emphasis of the curling motion. In other words, the subject can change the amount by which it opens its hand or the amount by which it wraps its fingers from trial to trial.

Whereas $\hat{X}$ is estimated in the velocity space, it is more intuitive to visualize the differences between trials on the position space (Figure 4.10) by integrating the latent factors along time and adding the corresponding initial hand posture (as in Equation 4.16). In this way we are able to compare the two replications in the positional space at specific time periods. For instance, we can choose (for illustrative purposes) the period between time points 50 and 58 (notice that it is valid to compare the two replications at the same time period because through the alignment procedure we accounted for the phase variation). We observe that whereas the first factor corresponding to replication 1 transitions from $X_1(t = 50) = +54.19$ to $X_1(t = 58) = -684.8$ (with a net change of -738.99), the first factor of replication 2 changes from $X_1(t = 50) = +73.21$ to $X_1(t = 58) = -178.5$ (with a net change of -251.71). While the net change is not meaningful by itself, the relative change is. The first/corresponding column in the loading matrix **B** suggests that these changes should result in an exaggerated opening of the hand in replication 1 as compared to replication 2. And, indeed by visualizing $\hat{Y}$ (and the observed data Y) we verify that the hand in replication 1 opens further than in replication 2 (see Figure 4.10).

This example that we just presented illustrates one of the highlights of our approach, namely, the ability to capture and display interpretable differences between replications of the same condition (through the interaction of the learned loading matrix and the latent factors) after having removed what is common among replications of the same condition. In other words, we are able to provide understanding of the variation between replications with the use of only a few components.

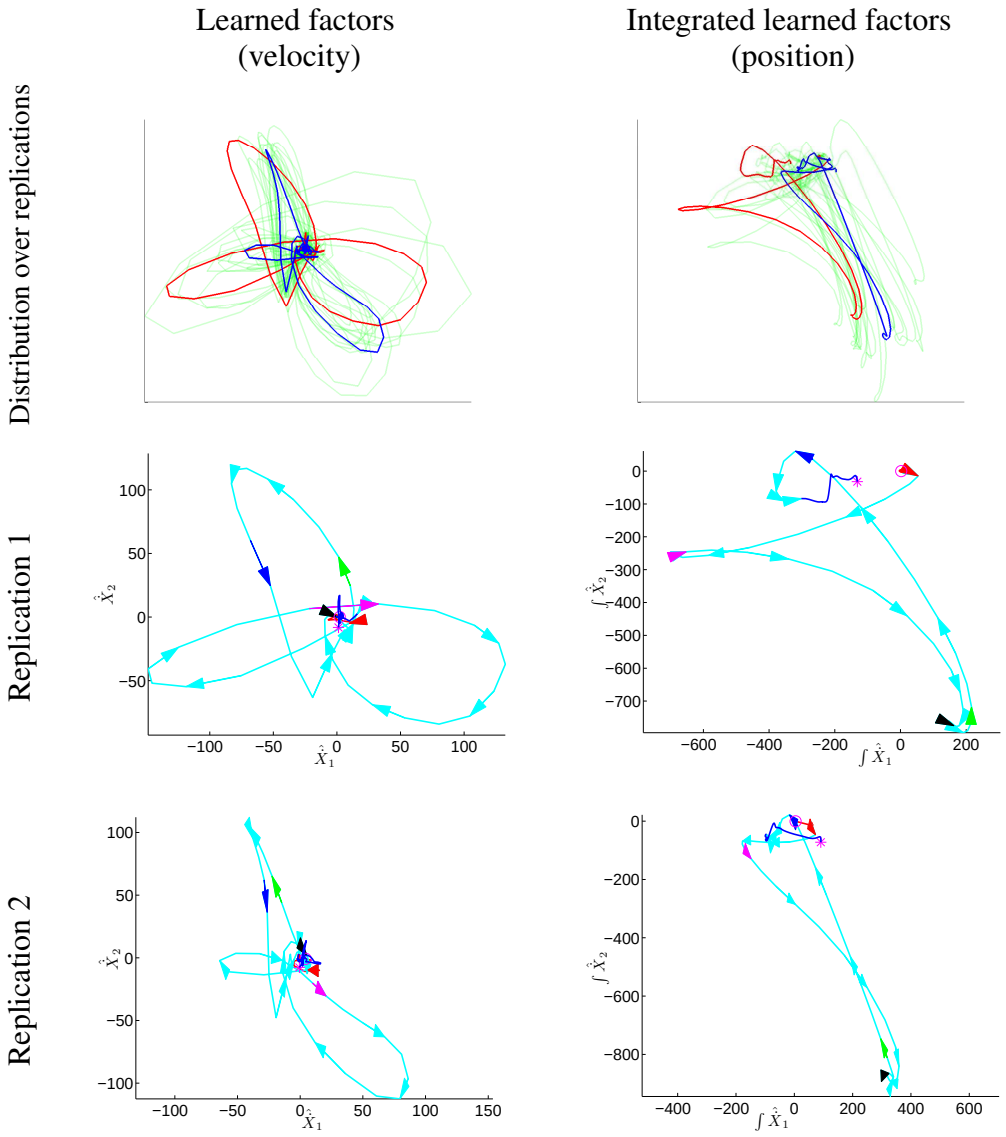


Figure 4.8: Learned factors $\hat{X}$ for condition: small cone, 45° abduction. In the top panel we show (in green) the distribution of learned factors in the velocity space (left) and their integrated version on the positional space (right). This figure depicts differences between trials in the space of learned factors. On this plot we overlap two exemplary trials: replication 1 (in red) and replication 2 (in blue). In the middle and lower panel we show details of these replications: the shape and values they display are different. The starting point of the trial is denoted by an open circle, the end position, by a star. There are arrows along the trajectory show the direction of movement. Colored arrows represent a percentage of the total time that the movement took and allow for comparison between trajectories: red arrow (33%), magenta (40%), black (50%), green (54%), blue (60%). In Figure 4.10 we show how difference between the integrated learned factors in these two trials manifest on hand posture.

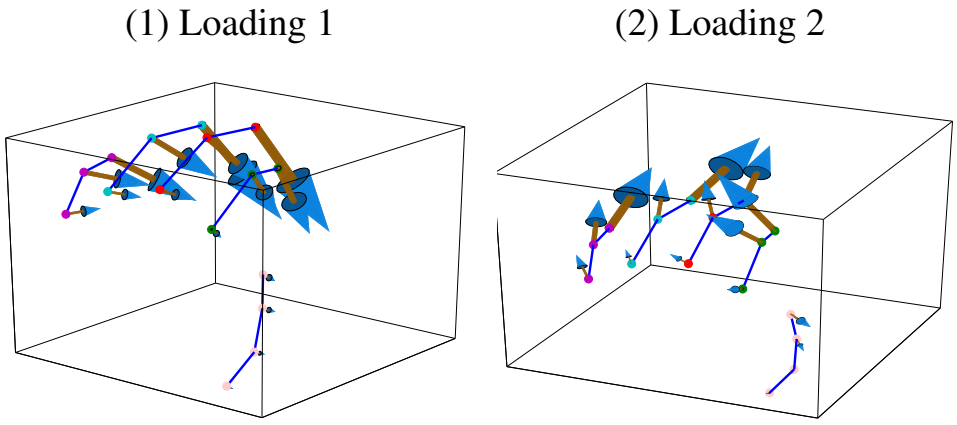


Figure 4.9: Visualization of loadings encoded in $\hat{\mathbf{B}}$ for the small cone presented at 45° abduction. The first loading corresponds to synchronized opening-closing of the hand; the second loading to curling of the fingers wrapping around the cone.

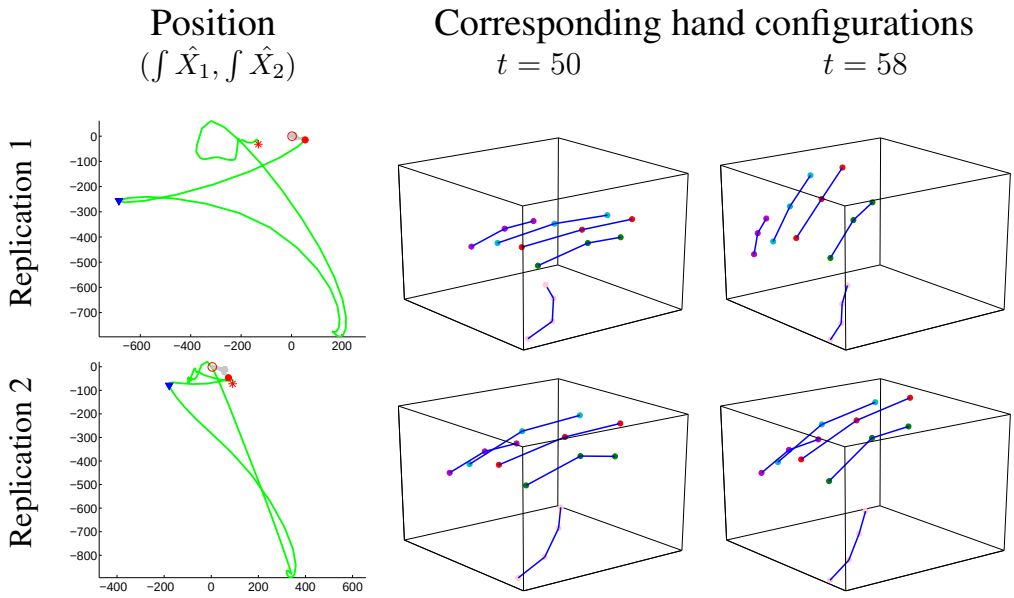


Figure 4.10: Interpretation of latent factors showing differences between replications. On the left we plot $(\int \hat{X}_1(t), \int \hat{X}_2(t))$ as a function of t. The start of the trial is at the red open circle (filled in with gray), the red circle corresponds to $t = 50$, the blue triangle to $t = 58$ and the star to the end of the trial. Middle and right panels: hand configurations corresponding to those time points. The interaction between the first latent factor (moving negatively) and the corresponding loading (Figure 4.9 panel 1) corresponds to an opening of the fingers in a synchronized manner – this movement differs between the two replications and leads to an exaggerated opening of the hand in replication 1 (top panel).

We have shown that each of the elements of the MGPFM have direct physical and intuitive interpretation.

Furthermore, we can differentiate the variability that corresponds to a specific replication from the variability shared among all trials from a specific condition. Finally, we are also able to accurately recover (in terms of reconstruction error) hand configurations from the estimated parameters and factors.

4.6 Chapter summary and main contributions

In this chapter we formulated a dynamic factor model based on Multivariate Gaussian Processes to study grasping kinematics. We developed an algorithm for inference and parameter estimation for the MGPFM and discussed variations of the model and algorithm. We showed in simulations that our model outperforms PCA in the reconstruction of error when alignment is applied. But, more importantly, in contrast with PCA or SVD we are able to differentiate sources of variation that can potentially be used to design robotic primitives for grasping devices. Our MGPFM can be extended by assuming prior distributions in the parameters (for instance, in the loading matrix), and can capture long range correlations, which can be good for coordinated dexterous hand motions. It is also easy to adapt to new settings — we can add sensors, change the representation to joint angles, and the same algorithms would be applicable in principle.

Our core methodological contribution is a strategy to decompose and reduce the dimensionality of the variation of the data according to the experimental structure (time, condition and replications). The decomposition of variance in the grasping datasets relied on application of a multivariate functional alignment procedure. A major product of this approach is the decomposition of variability between what is common in replications and what is specific for each trial; it also provides clear interpretation in the space of grasp postures. In particular, visualizations of the shared mean trajectory $\boldsymbol{\mu}$, of the axis of variation in replications encoded in the loading matrix $\mathbf{B}$ and of the specific differences in particular trials summarized in the latent factors X helped to explain variability in grasping movements.

Chapter 5

Decoding: algorithms and grasping features

We turn our attention now to the decoding problem. The problem of kinematic-neural decoding consists in mapping a representation of the neural activity to variables representing movement. Movement can be described with continuous variables, such as position, velocity or joint angles over time. Alternatively, movement can be represented by discrete events or states, like determining whether a particular finger is flexed or extended, or selecting a grasp configuration (among a discrete and finite set of types of grasps). In this chapter we deal with these two types of decoding: continuous and discrete, in two different ways.

First, we provide a framework to study the ability of single neurons to decode discrete interpretable grasping events (section 5.1). In particular, we define intuitive and interpretable variables that characterize grasping, identify relevant events such as maximum aperture, minimum speed in movement and others, and define a discrete decoding formulation of grasping events based on *maximum a posteriori* distributions. We demonstrate the ability of single neurons to decode these relevant events of grasping in a reliable and consistent way achieving decoding accuracies of up to 96% (where chance accuracy is 50%).

Second, we provide an algorithm for solving the continuous decoding problem efficiently in the context of recursive Bayesian decoders (section 5.2). In particular, we develop the *Laplace Gaussian Filter*, an algorithm to solve the filtering problem through an approximation of the posterior distribution in a deterministic way. Our algorithm consists in using Laplace's method to obtain estimates of the mean and variance of the posterior density, approximating that density by a Gaussian with the estimated mean and variance and recursively updating this distribution when the next observation is taken. Our algorithm provides an alternative to sequential Monte Carlo, provides superior results in a short amount of time and yields accurate results in simulations and real data.

In both cases, our approach is probabilistic and at the core of our reasonings lie Bayes rule and the computation of the most likely kinematic representation given the neural data.

5.1 Decoding of grasping features from primary motor cortex

To get continuous decoding in real data we typically need simultaneous recordings of many neurons (often at least ~ 25). For example, in our analysis of real data in Section 5.2.8 we use simultaneous recordings of 78 neurons in order to decode. But for now we focus on the question of how much information of *grasping* can we robustly extract from just one single neuron. In this section we describe an approach where we define meaningful interpretable variables that characterize grasping and identify relevant kinematic events associated with these variables. We then propose a discrete decoding formulation to investigate whether (and to what extent) individual neurons collected in our reach-and-grasp experiment are able to decode these events.

While natural, smooth and precise control of hand prosthesis cannot be achieved with the information of one single neuron, our study is scientifically illuminating, because it shows what information of grasping can be decoded from a single neuron. In addition, our approach can be useful for limited clinically viable systems where grasping is performed via discrete operations instead of being continuously controlled. Kennedy et al. (2000), for example, created a limited, but useful, clinically viable system when they demonstrated that a human patient, who suffered a brainstem stroke and only had residual facial movements, was able to control the horizontal coordinate of a cursor on a 2D screen using signals from only one or two neurons.[1] The patient was able to select letters on the screen to build words and sentences and communicate. Kennedy et al. (2000) was able to provide control over one horizontal degree of freedom. With our approach, we decode events based on the aggregated motion of all the fingers of the hand, which can be thought of as a form of dimensionality reduction.

5.1.1 Related work

In literature there are some works that have shown off-line decoding of continuous grasp aperture from spiking activity (Artemiadis et al., 2007; Zhuang et al., 2010; Bansal et al., 2011, 2012) and from ECoG data (Fifer et al., 2011) or finger position during a grasping task (Aggarwal et al., 2009). Some other works have shown off-line discrete decoding of grasp type (Stark and Abeles, 2007; Carpaneto et al., 2011, 2012; Hao et al., 2012; Xu et al., 2013) or of single or multiple finger flexion or extensions (Hamed et al., 2007; Aggarwal et al., 2008; Acharya et al., 2008; Baker et al., 2009; Shin et al., 2009a,b, 2010).

In the on-line setting (Velliste et al., 2008; Hochberg et al., 2012) controlled the open-and-close of a gripper and a robotic hand respectively, and Townsend et al. (2011) was able to decode discrete grasp types. All these studies used at least twenty units for decoding. We aimed at extracting relevant grasping information from a single neuron as in (Kennedy et al., 2000), who extracted information for moving a cursor on the screen. We also aimed at confirming the robustness of decoding the grasping events in many of the neurons that were sampled.

A first step to understand the grasping information is to summarize the information from the multiple markers on the fingers. One way to do it is by extracting synergies as in Chapters 3 or 4; or, alternatively, by defining some intuitive variables that describe the grasping, which can be thought of as a type of dimensionality reduction.

[1] This patient was implanted with a neurotrophic electrode (Kennedy and Bakay, 1998) on the primary motor cortex.

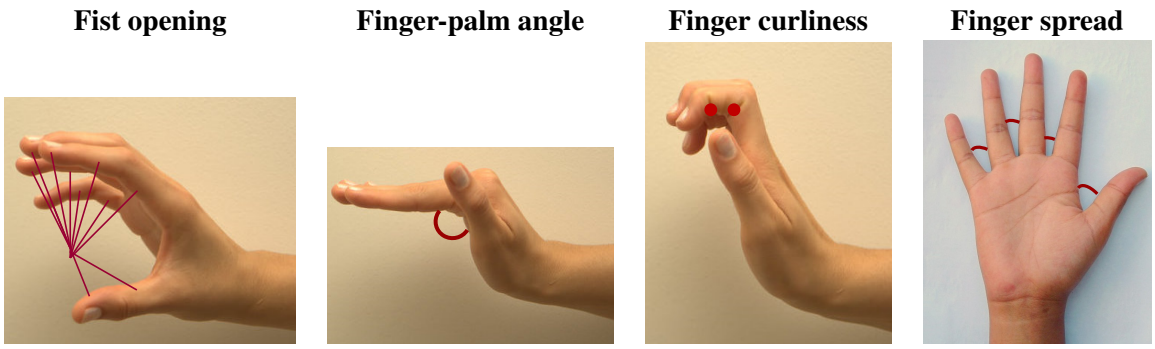


Figure 5.1: Definition of interpretable variables.

5.1.2 Interpretable variables

A problem when studying encoding of learned synergies is their interpretability. Synergies obtained with non-linear dimensionality reduction methods such as kPCA can be difficult to interpret. We define a set of intuitive and interpretable variables that summarize and describe the aggregated digits motion. The objective is to understanding the encoding of aggregated digit movement in an intuitive manner.

We propose to use the following interpretable variables:

- *fist opening*, or the sum of the Euclidean distances from each finger marker to its centroid $\mathcal{I}_1(t) = \sum_{j=1}^{16} \left\| Z_j^r(t) - c^r(t) \right\|_2^2$ where $Z_j^r(t) \in \mathbb{R}^{1\times 3}$ and $c^r(t)$ is the centroid of the markers position at time t of trial r, that is, of the rows of matrix $Z^r(t)$;
- *palm-finger angle* $\mathcal{I}_2(t)$, or aggregated sum of the metacarpo-phalangeal (MCP) joints of the digits;
- *finger curliness* $\mathcal{I}_3(t)$, or aggregated sum of the distal and proximal interphalangeal (DIP and PIP) joints of all the digits; and
- *finger spread* $\mathcal{I}_4(t)$, or aggregated sum of the angles between digits.

The last three variables can be defined as $\sum_{j\in\mathcal{W}} X_j^r(t)$ where $\mathcal{W}$ corresponds to the set of MCP joint flexions of the fingers in the first case, to the set of PIP and DIP joint flexions of the digits in the second case, and to the set of angles between fingers in the third case. Note that the only non-linear variable is the *fist opening*. This variable is also only one that includes information about the thumb. For an illustration, see Figure 5.1.

We now proceed to investigate whether some information of these interpretable variables is signaled by single neurons.

5.1.3 Decoding events framework

Our grasping datasets have very few neurons recorded simultaneously (refer to Chapter 2), so in this section we do not aim at reconstructing the whole grasping movement. Instead, we investigate ways to describe grasping through a set of relevant kinematic "events" that capture the relevant features of grasping, and we ask whether there is signal in the neurons that consistently decodes for these kinematics events.

We hypothesize that single neurons recorded from the primary motor cortex are sufficient to reliably decode relevant features of grasping that potentially can be used to control a basic prosthetic device with discretized commands. In order to verify our hypothesis we propose the following framework.

For a specific trial we consider a unidimensional variable along time $w(t)$ that describes grasping. For instance, let $w(t)$ be the interpretable variable hand aperture. Then we identify critical points of the variable, such as maximum, minimum, and change of sign (if any exist), and we call these critical points *kinematic events*. These kinematic events effectively discretize an aspect of the hand configuration. For instance, maximum aperture and minimum aperture represent two informative and opposite configurations of the digits. In effect, in the simplest case a grasp performed by a robotic hand could be controlled by a binary command: *maximize hand aperture* or *minimize hand aperture*. In addition to the *raw* variable $w(t)$ we can consider its rate of change $w'(t) = \frac{d}{dt}\, w(t)$ and its acceleration $w''(t) = \frac{d^2}{dt^2}\, w(t)$, and we can find the *kinematic events* of each of these functions. We can label two opposite kinematic events as x_i where $x_i \in \{+1, -1\}$ corresponding respectively to maximum and minimum. Each kinematic event is associated with some neural activity that precedes it. Our goal is to investigate the association between the neural signal and the kinematic event. If given a specific single neuron, we consider the neural activity z_i consisting of $280\, mS$ preceding a kinematic event, and we bin the activity in four $70\, mS$ windows as before (Section 3.2). The neural data together with all the labeled kinematic events associated with it form a data set. $\mathcal{D} = \{(z_i, x_i)\}_{i=1}^{n}$, where $z_i \in \mathbb{R}^4$, $x_i \in \{+1, -1\}$, and n is the number of kinematic events in the data set. In the case of considering the events to be the maximum and the minimum of a specific variable $w(t)$, n is equal to twice as many trials as are associated with a specific neuron. For a graphical description of how to build the data set, see Figure 5.2.

Decoding and encoding problems

Consider a kinematic event x (a binary variable in this case). Denote the probability distribution of the event by $P(x)$. Let $\boldsymbol{z}$ be the neural information as described above, and $P(\boldsymbol{z}|x)$ the conditional probability of the neural data given the specific event. Estimating $P(\boldsymbol{z}|x)$ from data constitutes the encoding problem within the event driven framework. The corresponding decoding problem is concerned with obtaining the *maximum a posteriori (MAP)* estimate of the kinematic event: $max_x\ P(x|\boldsymbol{z}) \propto P(\boldsymbol{z}|x)P(x)$, where we observe the firing rate $\boldsymbol{z}$ and we want to predict the discrete variable x representing the kinematic event.

Evaluation criteria

We evaluate the ability to predict the kinematic event correctly, performing k-fold cross validation and reporting the mean accuracy and standard deviation across folds. In order to make sure the results are correct, we use permutation tests for classification (Ojala and Garriga, 2010) and false discovery rates (Benjamini and Yekutieli, 2001) to control for Type I errors.

Permutation Tests for Classification (Ojala and Garriga, 2010). For each $(neuron,\ variable)$ pair we performed a permutation test where the null hypothesis H_0 is: $P(x, \boldsymbol{z}) = P(x)P(\boldsymbol{z})$, or equivalently, that the two conditional probability distributions are identical: $P(\boldsymbol{z}|x = 1) = P(\boldsymbol{z}|x = -1)$. We tested this hypothesis as follows:

1. Define $B = 10^3$, $k = 10$

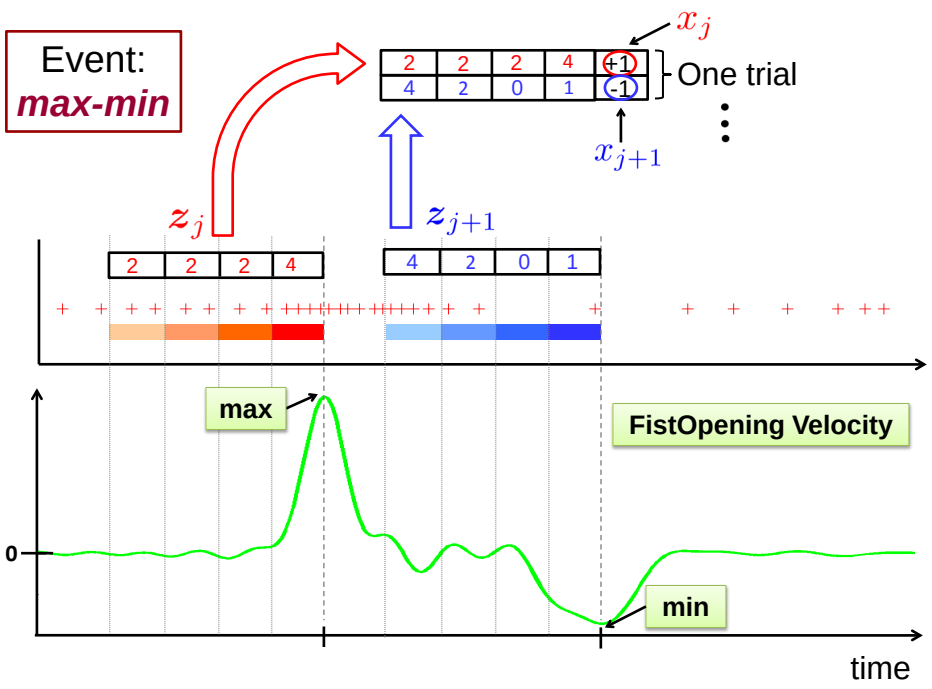


Figure 5.2: Definition of *kinematic event*: *maximum* vs *minimum*. (Top panel) Activity of neuron depicted with crosses. (Bottom panel) Fist opening velocity along time. In order to define kinematic graspig events, we identify the timing of critical points (like maximum and minimum) in the interpretable variables. We associate a label with these time points, for example, maximum as $+1$ and minimum as -1. We also associate the 280mS prior neural activity (binned in 70mS slots) to the corresponding time slice. We build one sample with the binned neural activity (as explanatory variables) and the class label associated with the kinematic event. (Session: Vinny 661, neuron: 0003a, trial: 16.)

2. Repeat B times:
   (a) permute labels
   (b) perform k-fold cross validated classification as explained above
   (c) record accuracy
3. Obtain histogram of B accuracies
4. Obtain k-fold cross validated accuracy of non-permuted (original) data set
5. Obtain the empirical p-value by:

$$p = \frac{\left|\{acc(f, \mathcal{D}') \geq acc(f, \mathcal{D}) : \mathcal{D}' \in \hat{\mathcal{D}}\}\right| + 1}{B + 1}$$

where $\mathcal{D}$ is the set of labeled data $\mathcal{D} = \{(z_i, x_i)\}_{i=1}^n$; n is the number of trials; π is a permutation of n elements; $\mathcal{D}'$ is a randomized version of $\mathcal{D}$ obtained by applying the permutation π on the labels i.e. $\mathcal{D}' = \{(z_i, \pi(x)_i)\}_{i=1}^n$; $\hat{\mathcal{D}}$ is a set of B randomized versions $\mathcal{D}'$ of the original data $\mathcal{D}$; f is our classifier which assigns a row of z_i to a label; and we denote the k-fold cross validation accuracy of the classifier f on the data $\mathcal{D}$ as $acc(f, \mathcal{D})$.

False Discovery Rate. We obtained (per data set) $numNeurons \times numVariables$ permutation tests, and thus we have as many p-values. We are interested in controlling the expected proportion of rejected hypotheses that are mistakenly rejected. FDR is that proportion – the proportion of tests (over all performed tests) where the null hypothesis is actually true. The FDR controls Error Type I. In our problem we apply the version of FDR testing that correspond to test statistics that are not independent (Benjamini and Yekutieli, 2001). This procedure is based on (Benjamini and Hochberg, 1995) and is as follows.

Consider m null hypotheses $\{H_0^1, H_0^2, \ldots, H_0^m\}$ and their test statistics. Let m_0 of these hypothesis be true null hypotheses. Consider that it is unknown which hypothesis are the true ones, and m_0 is also unknown. After testing these hypotheses there are m observed p-values: $\{p_1, \ldots, p_m\}$ where $p_i = 1 - F_{H_0^i}(S_i)$ and S_i is the test statistic of the i-th test. Then apply the following steps:

1. Sort the m observed p-values: $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(m)}$
2. Define $\kappa = max\left\{i : p_{(i)} \leq \frac{i}{m \cdot \sum_{i=1}^{m} \frac{1}{i}} \cdot q\right\}$ where $q = 0.05$ is our defined desired false discovery rate.
3. Reject $H_0^{(1)}, H_0^{(2)}, \ldots, H_0^{(\kappa)}$. If no such i exists, reject no hypotheses.

5.1.4 Experiments

We defined binary kinematic events related to all the interpretable variables defined in Section 5.1.2, and tried the proposed framework that decodes the kinematic events from neural data. We analyzed 83 neurons from the right hemisphere of Vinny and 155 neurons from the left hemisphere (see Table 2.3). From these neurons only 37 and 67 respectively, had fired in at least 100 trials and had an average number of spikes per trial higher than 10. We focused on these neurons.

In our framework, the prior $P(x)$ can be estimated as a proportion. We assumed the encoding model $P(z|x) \sim \mathcal{F}$, and we considered three models for $\mathcal{F}$:

1. The firing rates in different bins are correlated and the firing rate is distributed as a Multivariate Normal $N(\hat{\boldsymbol{\mu}}, \hat{\Sigma})$, $\hat{\boldsymbol{\mu}} \in \mathbb{R}^4, \hat{\Sigma} \in \mathbb{R}^{4\times 4}$.

2. The firing rates in different bins are independent and the firing rate is distributed as a Normal, thus: $P(\boldsymbol{z}|x) = \prod_{j=1}^{4} P(z^j|x) = \prod_{j=1}^{4} N(\hat{\mu^j}, \hat{\sigma^j})$, $\hat{\mu}, \hat{\sigma} \in \mathbb{R}$.

3. The firing rates in different bins are independent and the firing rate is distributed as a Poisson, thus: $P(\boldsymbol{z}|x) = \prod_{j=1}^{4} P(z^j|x) = \prod_{j=1}^{4} Poisson(\hat{\lambda^j})$, $\hat{\lambda^j} \in \mathbb{R}$.

5.1.5 Results

We focused on the set of interpretable variables defined in Section 5.1.2 and performed the analysis on data from Vinny. The first observation is that the three models we tried for $P(z|x)$ yielded very similar results. In Figure 5.3 we present the results of the multivariate normal model. We observe that three variables are decoded with accuracy larger than 75% consistently over a large proportion of sampled neurons on both hemispheres: finger curliness, velocity of palm-finger angle, and fist opening. Remarkably, some single neurons could be decoded with accuracy as high as 96%. Also some variables are decoded better by one

hemisphere than by the other; we hypothesize that this is due to the location of the area where neurons were recorded. Finally, events corresponding to finger spread do not appear to be decoded by either side of the brain from the neurons we sampled.

In order to make sure that we decoded digit related events versus arm events, we considered arm speed and defined arm events in an analogous fashion as before. We then decoded within our defined framework. Our results show that for most neurons whose grasp event decoding accuracy was high, those neurons decoded arm events accurately as well. That is, there is overlap between arm and finger movement signaling. However, there are some neurons that simultaneously (and significantly) decode digit events with high accuracy and arm events with low accuracy (see Table 5.1). This provides evidence for the existence of neurons that contain information of digit kinematics as opposed to arm movement.

We investigated further the correlation between the interpretable variables and arm speed. We found that the rate of change of the interpretable variables and arm speed yield an average Pearson's correlation coefficient larger than 0.6. This correlation suggests that arm speed and interpretable variables rate of change are confounded in the experiment design. This coupling may hamper the conclusions we can make about digit control; however, (a) these variables are correlated naturally in common reach-and-grasp tasks, and (b) the correlation coefficient is large, but not one. We hypothesize that the reason why these variables are not completely correlated is the actual grasping action; that is, the preshaping that varies according to object during the reach (see Figure 3.2) and the actual grasp where most of the movement corresponds to the digits grasping the objects.

Left hand (right hemisphere)		
	Finger curliness	Arm speed
Vinny000665spk004a	$\mathbf{95.97} \pm 3.07$	$\mathbf{64.15} \pm 7.42$
Vinny000667spk004a	$\mathbf{88.45} \pm 6.63$	$\mathbf{65.28} \pm 9.24$
	Palm-finger angle velocity	Arm speed
Vinny000667spk004a	$\mathbf{82.48} \pm 3.55$	$\mathbf{65.28} \pm 9.24$
Vinny000665spk004a	$\mathbf{76.63} \pm 4.31$	$\mathbf{64.15} \pm 7.42$
	Fist opening	Arm speed
Vinny000646spk001a	$\mathbf{77.28} \pm 6.1$	$\mathbf{59.85} \pm 6.13$
Vinny000667spk004a	$\mathbf{81.88} \pm 5.19$	$\mathbf{65.28} \pm 9.24$
Right hand (left hemisphere)		
	Palm finger angle velocity	Arm speed
Vinny000449spk001a	$\mathbf{82.22} \pm 6.44$	$\mathbf{62.78} \pm 11.21$

Table 5.1: Vinny. Decoding accuracy of interpretable variables (digit-related) and arm related kinematic events: decoded event max-min. Examples of neurons whose digit events decoding accuracy is high, and whose arm event decoding accuracy is low.

5.1.6 Conclusions

The choice of the neural activity model did not significantly affect the results. We found sound evidence that in a large proportion of the *single* recorded neurons in the hand/finger region of the primary motor cortex there is enough signal to reliably decode kinematic events related to the digits. Most of these neurons

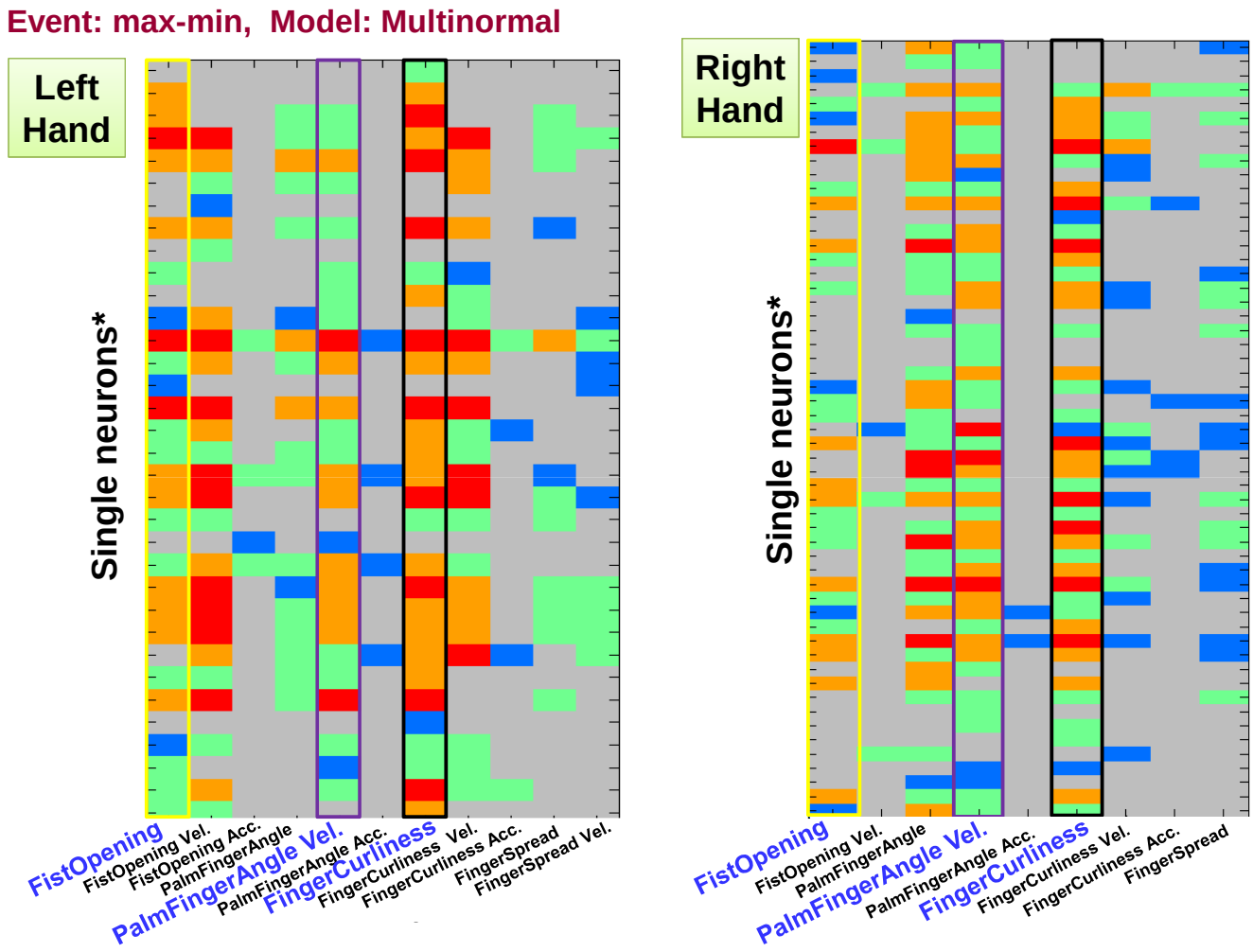


Figure 5.3: Matrix of results for Vinny with binary event maximum-minimum, using multinormal model: each row corresponds to a single neuron, each column to an interpretable variable. All neurons whose color code is not gray yielded classification accuracy higher than chance and passed the FDR test. Color coded cross-validated classification accuracy: Red: $> 85\%$, Orange: $75 - 85\%$, Green: $65 - 55\%$, Blue: $< 65\%$ but significantly higher than chance.

also decode kinematic events related to the arm; however, there exist neurons that irrefutably preferentially decode digit-related events as opposed to arm events. This confirms our hypothesis that we can reliably decode kinematic events related to digits configurations from single neurons, and motivates the exploration of encoding models for the digit related kinematic events. In addition, it is worth mentioning that we only require a *single* neuron to reliably perform the decoding, in some cases getting accuracies up to 96%. In contrast, other works that pose multiclass decoding problems (Shin et al., 2009b; Hamed et al., 2007) need at least twenty neurons to get reasonable accuracies.

In our approach we based the decoding on specific times when events were known to have happened. An extension of this work is to incorporate event timing detection. For example, one can use a Hidden Markov

Model to express prior knowledge about the relative timing of various events in a sequence.

5.1.7 Discussion

With modern intracortical arrays it is not significantly more expensive to implant an array of electrodes than it is to implant a single electrode. However, this does not undermine the idea of considering discrete interpretable kinematic events for the purposes of controlling a prosthetic device. For example, since events are more robustly decoded than continuous trajectories and require less data, they can be potentially more useful for other clinical settings such as non-invasive systems. Despite not giving a continuous trajectory one can imagine building a trajectory as a sequence of discrete events. And, in fact, one promising avenue of future work would be to model sequences of discrete kinematic events and decode these sequences using neural data.

5.2 Laplace-Gaussian Filter (LGF) for decoding continuous states

Some neural-motor prosthesis aim at reconstructing kinematics from spike train histories of multiple recorded neurons. Position, velocity and/or acceleration can be recorded continuously over time, and usually these variables contain significant temporal structure. Temporal structure of kinematic continuous variables can be described through a latent stochastical dynamical system (whose instantaneous value is the *state*) that evolves in time following a Markovian function. The set of noisy observations, in this case, the spike train histories can be assumed to be a parametric function of the state model. State-space models provide a framework to decode kinematic signals from neural spike trains (Brown et al., 1998; Brockwell et al., 2004; Truccolo et al., 2004; Srinivasan et al., 2007; Paninski et al., 2010).

The central problem in state-space models is *filtering*. Filtering refers to estimating the unobserved state at the present time, given the information provided by all of the observations received up to the present time. When linear and Gaussian assumptions are made, the *Kalman filter* provides the optimal solution. For nonlinear or non-Gaussian models, on the other hand, the main approach in literature is to obtain approximate solutions sometimes based on direct optimization (Paninski et al., 2010), but most frequently based on simulations such as particle filtering and its variants (Doucet et al., 2001). However, existing filtering methods, including sequential Monte Carlo, tend to be either inaccurate or slow. In this section, we propose a new nonlinear filter which uses Laplace's method, an asymptotic series expansion, to approximate the conditional mean and variance, and a Gaussian approximation to the conditional distribution of the state. This *Laplace-Gaussian filter* (LGF) gives fast, recursive, deterministic state estimates. In (Koyama et al., 2010a) we show that the corresponding error (which is set by the stochastic characteristics of the model) is stable over time. Here, we illustrate the decoding ability of the LGF by applying it to the problem of neural decoding and by comparing it to sequential Monte Carlo both in simulations and with real data. We find the LGF to be far more accurate, for equivalent computational cost, than particle filtering.

This section is organized as follows. In Section 5.2.1 we formalize the state-space model setting. In Section 5.2.2 we specify the construction of the Laplace-Gaussian filter and smoother in detail. In Section 5.2.3 we provide details of the implementation and the computational complexity of the LGF. In Section 5.2.4 we consider how to estimate the parameters of the model because, in general, these parameters are not known and need to be learnt from data. In Section 5.2.5 we mention the theoretical guarantees that Shinsuke Koyama proved. In Sections 5.2.6 we place the LGF within the context of the neural decoding problem, and show its use in simulations (Section 5.2.7) and in real data (Section 5.2.8). In Section 5.2.9 we discuss our conclusions and contributions.

5.2.1 Formalization and related work on filtering

Let $\{x_t\}_{t=1}^T$ be an stochastic state process and a related observation process $\{y_t\}_{t=1}^T$, which is a noisy function, generally nonlinear, of the state process. Filtering consists of estimating the the state x_t given a sequence observations $y_1, y_2, \ldots, y_t$ denoted by $y_{1:t}$, that is, finding the posterior distribution $p(x_t|y_{1:t})$ of the state, given the sequence. Assume that the state x_t is a first-order homogeneous Markov process, with initial density $p(x_1)$ and transition density $p(x_{t+1}|x_t)$, and that y_t is independent of states and observations at all other times given x_t, with observation density $p(y_t|x_t)$. Applying Bayes's rule gives a recursive

filtering formula,

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}, \tag{5.1}$$

where

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{5.2}$$

is the predictive distribution, which convolves the previous filtered distribution with the transition density.

In principle, Equations 5.1 and 5.2 give a complete, recursive solution to the filtering problem for state-space models: the mean-square optimal point estimate is simply the mean of Equation 5.1. However, when the dynamics are nonlinear, non-Gaussian, or even just high dimensional, computing these estimates sequentially can be a major challenge.

A way for estimating the posterior mean is to provide an approximation by sampling from the posterior distribution. Applying Monte Carlo to Equations 5.1–5.2 would let us draw from $p(x_t|y_{1:t})$, if we had $p(x_t|y_{1:t-1})$. Particle filtering (Doucet et al., 2001) consists in approximating the latter distribution by Monte Carlo sampling. This turns the recursive equations for the filtering distribution into a stochastic dynamical system of interacting particles (Del Moral and Miclo, 2000), each representing one draw from that posterior. While particle filtering has proven itself to be useful in practice (Doucet et al., 2001; Brockwell et al., 2004; Ergün et al., 2007), like any Monte Carlo scheme it is computationally very costly; moreover, the number of particles (and so the amount of computation) needed for a given accuracy grows rapidly with the dimensionality of the state-space. For real-time processing, such as neural decoding, the computational cost of effective particle filtering can become prohibitive.

The primary difficulty with the nonlinear filtering equations comes from their integrals. Particle filtering approximates the integrals stochastically. We propose here to approximate them deterministically, with the asymptotic expansion known as Laplace's method (Erdélyi, 1956). Specifically, we use Laplace's method to obtain estimates of the mean and variance of the posterior density (5.1), and then approximate that density by a Gaussian with that mean and variance. This distribution is then recursively updated in its turn when the next observation is taken. We call this method the *Laplace-Gaussian Filter* (LGF).

There are several versions of Laplace's method, all of which replace integrals with series expansion around the maxima of integrands. An expansion parameter, γ (sample size or the sharpness of a tuning curve) measures the concentration of the integrand about its peak. In the simplest version, the posterior distribution is replaced by a Gaussian centered at the posterior mode. Under mild regularity conditions, this gives a first-order approximation of posterior expectations, with error of order $O(\gamma^{-1})$. Several papers have applied some form of the first-order Laplace approximation sequentially (Brown et al., 1998; Eden et al., 2004). In the ordinary static context, Tierney et al. (1989) showed that a refined procedure, the "fully exponential" Laplace approximation, gives a second-order approximation for posterior expectations, having an error of order $O(\gamma^{-2})$. In (Koyama et al., 2010a) we provide theoretical results justifying these approximations in the sequential context. Because state estimation proceeds recursively over time, it is conceivable that the approximation error could accumulate, which would make the approach ineffective. Koyama's proofs show that, under reasonable regularity conditions, this does not happen: the posterior mean from the LGF approximates the true posterior mean with an error $O(\gamma^{-\alpha})$ uniformly across time, where $\alpha = 1$ or 2 depending on the order of the LGF.

In the following section we develop the LGF and in later sections we show its application to neural decoding of continuous variables in simulations and in real data.

5.2.2 Laplace-Gaussian Filter and Smoother

Let $x_{t|t}$ and $v_{t|t}$ denote the mode and variance of the true filtered distribution at time t given a sequence of observations $y_{1:t}$. Similarly, let $x_{t|t-1}$ and $v_{t|t-1}$ be the true mode and variance of the true predictive distribution at time t given $y_{1:t-1}$. Let $\hat{x}$ denote the approximated posterior mode, and $\tilde{x}$ the approximated posterior mean.

LGF algorithm

The procedure of the LGF consists of the following steps:

1. At time $t = 1$, initialize the distribution of the state $p(x_1)$.
2. Observe y_t.
3. *Filtering.* Obtain the approximate posterior mean $\tilde{x}_{t|t}$ and variance $\tilde{v}_{t|t}$ by Laplace's method (see below), and set $\hat{p}(x_t|y_{1:t})$ to be a Gaussian distribution with these approximated mean and variance.
4. *Prediction.* Calculate the predictive distribution,

$$\hat{p}(x_{t+1}|y_{1:t}) = \int p(x_{t+1}|x_t)\hat{p}(x_t|y_{1:t})dx_t. \tag{5.3}$$

5. Increment t and go to step 2.

First-order Laplace approximation. In the first-order Laplace approximation the posterior mean and variance are

$$\tilde{x}_{t|t} = \hat{x}_{t|t} \equiv \underset{x_t}{\operatorname{argmax}}\, \hat{l}(x_t)$$

and

$$\tilde{v}_{t|t} = [-l''(\hat{x}_{t|t})]^{-1} ,$$

where

$$l(x_t) = \log p(y_t|x_t)\hat{p}(x_t|y_{1:t-1}) .$$

Note that for this approximation to be valid $l(x_t)$ must be unimodal.

Second-order (fully exponential) Laplace approximation. The second-order Laplace approximation is calculated as follows (Tierney et al., 1989). For a given function g of the state, let $k(x_t) = \log g(x_t)p(y_t|x_t)\hat{p}(x_t|y_{1:t-1})$ and $\bar{x}_{t|t}$ maximize k. The posterior expectation of g for the second order approximation is then

$$\hat{E}[g(x_t)|y_{1:t}] \approx \frac{|-k''(\bar{x}_{t|t})|^{-\frac{1}{2}} \exp[k(\bar{x}_{t|t})]}{|-l''(\hat{x}_{t|t})|^{-\frac{1}{2}} \exp[l(\hat{x}_{t|t})]}. \tag{5.4}$$

When the g we care about is not necessarily positive, so we add a large constant c to g so that $g(x) + c > 0$, apply Equation 5.4, and then subtract c. The posterior mean is thus calculated as $\tilde{x}_{t|t} = \hat{E}[x_t + c] - c$. See

(Tierney et al., 1989) for the details of this method.[2] The posterior variance is set to be $\tilde{v}_{t|t} = [-l''(\hat{x}_{t|t})]^{-1}$, as this suffices for obtaining the second-order accuracy (Koyama et al., 2010a).

To use this method to estimate a d-dimensional space, one simply takes the function g to be each coordinate in turn, i.e., $g(\boldsymbol{x}_t) = x_{t,i} + c$ for each $i = 1, 2, \ldots, d$. Each g is a function of $\mathbb{R}^d \to \mathbb{R}$, and $|-l''(\hat{x}_{t|t})|^{-\frac{1}{2}}$ and $|-k''(\bar{x}_{t|t})|^{-\frac{1}{2}}$ in Equation 5.4 are replaced by the determinants of the Hessians of $l(\hat{\boldsymbol{x}}_{t|t})$ and $k(\bar{\boldsymbol{x}}_{t|t})$, respectively. Thus, estimating the state with the second-order LGF takes d times as long as using the first-order LGF, since posterior means of each component of $\boldsymbol{x}_t$ must be calculated separately.

In many applications the state process is taken to be a linear Gaussian process (such as an autoregression or random walk) so that the integral in Equation 5.3 is analytic. When this integral is not done analytically, either an asymptotic expansion or a numerical method may be employed. To apply our theoretical results, the numerical error in the integration must be $O(\gamma^{-\alpha})$, where γ is the expansion parameter discussed in the next paragraphs and $\alpha = 1$ or 2 depending on the order of the LGF.

Meaning of γ. Based on the regularity conditions stated in (Erdélyi, 1956; Kass et al., 1990), for a given state-space model γ is constructed by combining the model parameters so that the log posterior density is scaled by γ as $\gamma \to \infty$. In general, γ is interpreted in terms of the sample size the concentration of the observation density, and the inverse of the noise in the state dynamics. In the context of neural decoding, γ can be constructed in terms of the number of neurons, the noise in the state dynamics and the sharpness of the neuronal tuning curves (see Equation 5.12). From the construction of γ, the second derivative of the log-posterior density (which determines its concentration) is also scaled by γ. Therefore, the larger γ is, the more precisely variables can be estimated, and the more accurate Laplace's method becomes. When the concentration of the posterior density is not uniform across state dimensions in a multidimensional case, a multidimensional γ could be taken. Without a loss of approximation accuracy, however, a simple implementation of this case is taking the largest γ as a single expansion parameter.

Smoothing

Smoothing is a related operation to filtering, but computationally more challenging (Doucet et al., 2001). Smoothing corresponds to estimating the distribution of the state at a particular time given all of the observations up to some later time. That is, later observations are used to improve the estimates of states at earlier time points and, as a result, trajectory estimates tend to be smoother than those estimated by filtering (Doucet et al., 2001). Formally, the *smoothed* state distributions $p(x_t|y_{1:T})$ when $t \leq T$ are calculated from filtered and predictive distributions by recursing backwards (Anderson and Moore, 1979).

The LGF can be used for smoothing. Instead of the true filtered and predictive distributions, however, we now have the approximated filtered and predictive distributions computed by the LGF. By using these approximated distributions, the approximated smoothed distributions can be obained as

$$\hat{p}(x_t|y_{1:T}) = \hat{p}(x_t|y_{1:t}) \int \frac{\hat{p}(x_{t+1}|y_{1:T})p(x_{t+1}|x_t)}{\hat{p}(x_{t+1}|y_{1:t})} dx_{t+1}. \tag{5.5}$$

[2] In practice it suffices that the probability of the event $\{g(x_t) + c > 0\}$ be close to one under the true distribution of x_t. Allowing this to be merely very probable rather than almost sure introduces additional approximation error, which however can be made arbitrarily small simply by increasing the constant c.

Shinsuke Koyama provides a proof of the accuracy of LGF smoothing in (Koyama et al., 2010a).

5.2.3 Implementation and computational complexity

When fitting the LGF the log-likelihood of the model given the data must be maximized. In addition, the Hessian matrix of the loglikelihood must be obtained and evaluated. Maximizing the loglikelihood might not have an analytical solution and obtaining the second derivative might be difficult or cumbersome. Here we explain how we approached these challenges.

We maximize the log-likelihood through Newton's method (an iterative procedure). We initialize the algorithm with $\hat{x}_{t|t-1}$ for maximizing $l(x_t)$ and $\hat{x}_{t|t}$ for maximizing $k(x_t)$, since these initial values are often close to the solutions. As for the convergence criterion: the iteration should be stopped at the l^{th} iteration where $\left\|x^{(l+1)} - x^{(l)}\right\| < \gamma^{-\alpha}$ is satisfied, where $x^{(l)}$ is the value obtained at the l^{th} step. α is set to 1 for the first-order approximation, $\alpha = 2$ for the second-order approximation, and γ is a parameter that controls the accuracy of the Laplace's method. In our neural application, γ depends on the number of neurons, the noise of the state dynamics and the sharpness of the neural tuning curves (see Equation 5.12 below).

Laplace approximation requires the second derivative (or the Hessian matrix for multi-dimensional state-space) of the log-likelihood function evaluated at its maximum. However, obtaining the analytical derivative can be difficult or cumbersome. Therefore we use an accurate numerical approximation of the derivative: an iterated version of the *Richardson extrapolation* ((Kass, 1987) and Appendix Section 7.8.2).

Computational complexity

Let d be the dimensionality of the state, T the number of time steps, and N be the sample size (or the number of neurons in the neural decoding context). Assume maximization of the log-likelihood through Newton's method and the evaluation of the Hessian of the log-likelihood as explained above.

In the first order LGF $l(x_t)$ needs to be maximized at each time t. At each iteration of Newton's method, the evaluation of the Hessian matrix of $l(x_t)$ has time complexity $O(Nd^2)$, as d^2 is the time complexity for matrix manipulation. Maintaining accuracy as d and N grow does not require more iterations of Newton's method and the total number of iterations is rather small (around five iterations in our neural decoding application). Considering the T time steps, the total time complexity of the first-order LGF is $O(TNd^2)$.

The second order LGF requires maximizing both $l(x_t)$ and $k(x_t)$. Although the accuracy of approximation is order of γ^{-1} higher than that of the first-order LGF, it takes only one more iteration of Newton's method, because of its quadratic convergence. Thus the time complexity of calculating the posterior expectation of each $x_{t,i}$ is still $O(Nd^2)$, but calculating it for $i = 1, \ldots, d$ results in $O(Nd^3)$. Considering all T time steps, the complexity of the second-order LGF is $O(TNd^3)$.

For comparison, particle filter (PF) with M particles has complexity $O(TMNd)$: most of the computational cost of the PF comes from calculating M weights at each time step; finding each weight needs an invocation of the observation model, which requires time $O(Nd)$; and thus the computational cost across all T time steps is $O(TMNd)$. For the computational cost of the particle filter to be comparable with the LGF, the number of particles should be $M \sim d$ for the first order LGF and $M \sim d^2$ for the second order LGF.

5.2.4 Parameter estimation

The primary use of the LGF is to obtain estimates of the state x_t, but in the filtering problem one assumes that the model parameters θ are known. However, this is often not the case, which means that, in practice, the Markov model is actually $p(x_t|x_{t-1};\theta)$ and the observation model, $p(y_t|x_t;\theta)$. Hence, one must jointly estimate the states and the model parameters.

In principle, one could do this by maximizing the likelihood,

$$\log p(y_{1:T};\theta) = \log \int p(x_{1:T}, y_{1:T};\theta) dx_{1:T} = \log \int \prod_{t=1}^{T} p(x_t|x_{t_1};\theta) p(y_t|x_t;\theta) dx_{1:T}. \tag{5.6}$$

This is, however, hard without knowing $x_{1:T}$. A standard strategy is to use the expectation-maximization (EM) algorithm (Dempster et al., 1977; Baum et al., 1970) (see Appendix Section 7.8.3).

When the complete data distribution is of an exponential family, the E-step consists of finding the expected values of certain sufficient statistics of the complete data. Those expectation values can be approximated by the LGF. We illustrate the derivation of the EM algorithm with the LGF and smoothing within the context of our neural decoding application in Section 5.2.6.

5.2.5 Theoretical guarantees

Erdélyi (1956); Kass et al. (1990); Wojdylo (2006) stated and studied the regularity conditions that are sufficient for the validity of Laplace's method. Based on these conditions it can be shown that the LGF satisfies the following properties:

1. *Accuracy of predictive distributions / Non-amplification of error.* The error in approximated predictive distribution generated by the LGF approximation at each time-step does not accumulate along time, in fact, the error term is bounded uniformly across time. This theorem is proved by calculating the asymptotic expansions of both the true and approximated predictive distributions and matching terms.

2. *Accuracy in the posterior expectations.* The error in the approximated posterior expectation is also bounded uniformly over time.

3. *Stability of the algorithm.* Under suitable regularity conditions, the sequential Laplace approximation to the predictive distribution is *stable* in the sense that two approximately equal versions at time $t-1$ lead to approximately equal versions at time t. That is, minor differences in the initially-guessed distribution of the state tend to be reduced, rather than amplified, by conditioning on further observations, even under the Laplace approximation. This means that, estimates become progressively less dependent on the initial prior as more data arrives.

These three results together with an analogous result for the LG smoother were proved by Shinsuke Koyama in our paper (Koyama et al., 2010a).

5.2.6 LGF in the context of neural decoding

Neurons in the primary motor cortex fire preferentially in response to velocity $\boldsymbol{v}_t \in \mathbb{R}^3$ and position $\boldsymbol{z}_t \in \mathbb{R}^3$ of the hand Georgopoulos et al. (1986a); Ketter et al. (1988); Paninski et al. (2004a); Wang and Moran

(2007). These covariates can be linearly combined to model the logarithm of the mean firing rate of neurons in primary motor cortex in a similar fashion as in (Truccolo et al., 2004) where the conditional intensity function of a point process was used to define a neuron's spiking probability in terms of a linear combination of functions of the covariates.

Let $\boldsymbol{v}_t \in \mathbb{R}^3$ denote the velocity and $\boldsymbol{z}_t \in \mathbb{R}^3$ the position of the hand at time t . Consider a population of N neurons, such that the mean firing rate of neuron i is:

$$\lambda_i(\boldsymbol{z}_t, \boldsymbol{v}_t) = \exp(\alpha_i + \boldsymbol{\beta}_i \cdot \boldsymbol{v}_t + \boldsymbol{\eta}_i \cdot \boldsymbol{z}_t), \tag{5.7}$$

where $\alpha_i \in \mathbb{R}^1$ represents the baseline firing rate of neuron i, $\boldsymbol{\beta}_i \in \mathbb{R}^3$ denotes the *preferred direction* and the sharpness of its tuning curve, and $\boldsymbol{\eta}_i \in \mathbb{R}^3$ can be understood as the neuron's *position gradient*. We can write $\boldsymbol{x}_t = (\boldsymbol{z}_t, \boldsymbol{v}_t)$ as a state column vector in 6-D space, and $\boldsymbol{\theta}_i = (\boldsymbol{\eta}_i, \boldsymbol{\beta}_i)$ as a *preferred movement* column vector in 6-D space. In fact, in general, we can consider a potentially large dimension d on the state $\boldsymbol{x}_t \in \mathbb{R}^d$ that could represent, for instance, the fingers configuration, velocity and/or acceleration. Hence, we can model the instantaneous mean firing rate of neuron i as:

$$\lambda_i(\boldsymbol{x}_t) = \exp(\alpha_i + \boldsymbol{\theta}_i \cdot \boldsymbol{x}_t), \tag{5.8}$$

where $\boldsymbol{\theta}_i$ corresponds to the parameters that determine the tuning of the neuron based on kinematic variables. In this and the following sections we refer to Equation 5.7 as the simple model and to Equation 5.8 as the general model of firing rates.

In principle, the response of the neurons is a continuous-time point process, but we assume (in line with the experimental situation) that we make observations at discrete times recording the number of spikes that happen over time-intervals of length Δ. Let $y_{i,t}$ denote the spike count of neuron i at time t and suppose that $y_{i,t}$ is a *poisson process* with intensity $\lambda_i(\boldsymbol{x}_t)\,\Delta$, and that the firing of different neurons is independent from each other conditioned on $\boldsymbol{x}_t$. Then, the probability distribution of $\boldsymbol{y}_t$, the vector of all the $y_{i,t}$ corresponds to the product of the firing probabilities of each neuron.

The state model in the simple model can be assumed to be

$$\boldsymbol{x}_t = \begin{pmatrix} I & \Delta I \\ O & I \end{pmatrix} \boldsymbol{x}_{t-1} + \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\epsilon}_t \end{pmatrix}, \tag{5.9}$$

where $\boldsymbol{\epsilon}_t$ is a 3-D Gaussian random variable with mean zero and covariance matrix $\sigma^2 I$, I being the identity matrix. Or, in the general model, the state model can be assumed to be a d-dimensional autoregressive (AR) process:

$$\boldsymbol{x}_t = \boldsymbol{F}\,\boldsymbol{x}_{t-1} + \mathrm{E}_t \tag{5.10}$$

where $\boldsymbol{F} \in \mathbb{R}^{d \times d}$ and E_t is a d-dimensional Gaussian random variable with mean zero and covariance matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$.

Parameters $\boldsymbol{\theta}_i$ and σ^2 can be learned via expectation maximization. However, in the context of neural decoding (and to make the comparison fair to other neural decoders) we learn parameters $\boldsymbol{\theta}_i$ through Poisson regression of spike counts on kinematic variables (and use the same preferred movement vectors in all

decoders), and we learn σ^2 (or $\boldsymbol{W}$) via maximum likelihood (through EM).

The construction for the LGF for the simple model Equation (5.7) and (5.9) and the EM for learning the variance is as follows.

Derivation of LGF for simple model of neural decoding

Let $\tilde{\boldsymbol{x}}_{t|t-1} \in \mathbb{R}^6$ and $\tilde{V}_{t|t-1} \in \mathbb{R}^{6\times 6}$ be the predictive mean and the covariance matrix, and similarly let $\tilde{\boldsymbol{x}}_{t|t}$ and $\tilde{V}_{t|t}$ be the posterior mean and covariance at time t.

In the *filtering* step, we introduce

$$l(\boldsymbol{x}_t) = \log p(\boldsymbol{y}_t|\boldsymbol{x}_t)\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})$$

and

$$k_i(\boldsymbol{x}_t) = \log g_i(\boldsymbol{x}_t)p(\boldsymbol{y}_t|\boldsymbol{x}_t)\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) \ .$$

Here $g_i(\boldsymbol{x}_t) = x_{t,i} + c$ for computing the i-th element of the posterior mean of $\boldsymbol{x}_t$, and $c \gg 0$ is a large constant[3]. By inspecting the second derivative of l,

$$l''(\boldsymbol{x}_t) = -\sum_{i=1}^{N} \exp(\alpha_i + \boldsymbol{\theta}_i \cdot \boldsymbol{x}_t)\Delta\boldsymbol{\theta}_i\boldsymbol{\theta}_i^T - V_{t|t-1}^{-1}, \tag{5.11}$$

we can identify the expansion parameter:

$$\gamma = \frac{1}{\sigma^2} + \sum_{i=1}^{N} e^{\alpha_i}\|\boldsymbol{\theta}_i\|^2 \ , \tag{5.12}$$

so Laplace's method grows more accurate as the state noise shrinks, as the number of neurons grows, as the firing rates of neurons increase, and as the neuronal response becomes more sharply peaked around the preferred movement.

In the *prediction* step, since both the approximated filtered density, $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$, and the transition density, $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)$, are Gaussian, the predictive density (Equation 5.3) is also Gaussian, with mean

$$\tilde{\boldsymbol{x}}_{t+1|t} = \begin{pmatrix} I & \Delta I \\ O & I \end{pmatrix} \tilde{\boldsymbol{x}}_{t|t}, \tag{5.13}$$

and covariance

$$\tilde{V}_{t+1|t} = \begin{pmatrix} I & \Delta I \\ O & I \end{pmatrix} \tilde{V}_{t|t} \begin{pmatrix} I & O \\ \Delta I & I \end{pmatrix} + \begin{pmatrix} O & O \\ O & \sigma^2 I \end{pmatrix} \ . \tag{5.14}$$

We take the initial value for filtering to be the hand position and velocity at the time origin. This completes the LGF for this model.

[3]We took $c = 10^4$ in our simulation and data analysis below.

EM for learning the covariance of the state

We apply the EM algorithm[4] to determine the optimal value of σ^2. Given a current estimate $\sigma^2_{(l)}$, maximizing Equation 7.11 with respect to σ^2 leads to the updating formula as

$$\sigma^2_{(l+1)} = \frac{1}{3(T-1)} \sum_{t=2}^{T} \sum_{i=4}^{6} \left\{ \tilde{V}^{(i,i)}_{t|T} + \tilde{V}^{(i,i)}_{t-1|T} - 2\tilde{V}^{(i,i)}_{t,t-1|T} + (\tilde{x}_{t|T,i} - \tilde{x}_{t-1|T,i})^2 \right\}, \tag{5.15}$$

where the smoothed mean, $\tilde{\boldsymbol{x}}_{t|T}$, variance, $\tilde{V}_{t|T}$, and covariance of the state, $\tilde{V}_{t,t-1|T}$, are calculated with $\sigma^2_{(l)}$. Since the approximated predictive and filtered distributions are both Gaussian, the smoothed distributions (Equation 5.5) are also Gaussian, and the backward formula for calculating the mean and variance of the smoothed distributions are

$$\tilde{\boldsymbol{x}}_{t|T} = \tilde{\boldsymbol{x}}_{t|t} + K_t(\tilde{\boldsymbol{x}}_{t+1|T} - \tilde{\boldsymbol{x}}_{t+1|t}), \tag{5.16}$$

and

$$\tilde{V}_{t|T} = \tilde{V}_{t|t} + K_t(\tilde{V}_{t+1|T} - \tilde{V}_{t+1|t})K_t^T, \tag{5.17}$$

for $t = T-1, T-2, \ldots, 1$, where

$$K_t = \tilde{V}_{t|t} \begin{pmatrix} I & O \\ \Delta I & I \end{pmatrix} \tilde{V}^{-1}_{t+1|t}. \tag{5.18}$$

(See (Anderson and Moore, 1979).) The smoothed covariance can be computed by

$$\tilde{V}_{t+1,t|T} = K_t \tilde{V}_{t+1|T}, \tag{5.19}$$

for $t = 1, 2, \ldots, T-1$ (de Jong and Mackinnon, 1988).

Thus, the complete EM procedure for this model, with σ^2 unknown, goes as follows:

1. At the iteration step $l = 1$, set the initial value, $\sigma^2_{(1)}$.
2. *E-step.* Apply the LGF and smoothing algorithm with $\sigma^2_{(l)}$ to obtain $\tilde{\boldsymbol{x}}_{t|T}$, $\tilde{V}_{t|T}$ and $\tilde{V}_{t,t-1|T}$.
3. *M-step.* Update the value of σ^2 by Equation 5.15.
4. Repeat steps 2-3 until the value of σ^2 converges[5].

5.2.7 Simulation study

We performed two simulation studies. In the first one we compared the performance of the LGF to the actual posterior mean and to estimates of obtained trhough particle filtering. We also showed how the accuracy of LGF varied as the state dimensionality grew. In the second simulation we show experimentally how the accuracy of the LGF state estimates grow as the number of neurons increases.

[4] A similar algorithm for neural data analysis was derived in (Smith and Brown, 2003).

[5] We used the convergence criterion $|\sigma^2_{(l+1)} - \sigma^2_{(l)}|/\sigma^2_{(l)} < 10^{-3}$.

LGF accuracy and error scaling when varying the state dimensionality

In this first simulation, we aimed at comparing the performance of the LGF state estimates to (1) the actual posterior mean, and (2) to the estimates obtained through the particle filtering as in (Brockwell et al., 2004); and we observed how the LGF accuracy behaved when varying the state dimensionality. We obtained the first order and second order LGF estimates and labeled them as LGF-1 and LGF-2.

We performed these comparisions on the general model specified by Equations 5.8 and 5.10 at various values of the state dimensionality $d = 6, 10, 20, 30$. The model parameters were set as follows: $T = 30$ time steps of duration, $\Delta = 0.03$ seconds, a population of $N = 100$ neurons with $\alpha_i = 2.5 + \mathcal{N}(0, 1)$ and $\boldsymbol{\theta}_i$ sampled uniformly from the unit sphere in $\mathbb{R}^d$; where the spike counts were drawn from Poisson distributions with firing rates $\lambda_i(\boldsymbol{x}_t)$; and where the state dynamics were determined by $\boldsymbol{F} = 0.94I$, $\boldsymbol{W} = 0.019I$ and Equation (5.10).

In order to compute the *actual* posterior mean we used particle filtering with a large number of particles (10^6), averaged over 10 independent realizations and verified that the order of the mean integrated square error (MISE) was neglegible – it resulted in an error of order $O(10^{-7})$. We then compared the two versions of LGF with PF with 100 particles for the different values of d. In addition, to be fair and based on the computational cost analysis, we compared to what we called PF-scaled. For this analysis we calculated the number of particles needed to match the computational time of PF with that one of the second order LGF. Hence, we chose 100, 300, 500 and 1000 particles for $d = 6, 10, 20$ and 30.

The first four rows in Table 5.2 show the mean integrated square error in approximating the actual posterior mean. The best results were attained by the LGF-2, followed by LGF-1. The two versions of LGF yielded better results than the versions of particle filtering. Table 5.3 shows the time in seconds used to obtain the estimates for the four filters. The LGF-1 is the fastest method; LGF-2 and PF-scaled take roughly the same amount of time.

Figure 5.4 shows the mean integrated square error for particle filtering in approximating the actual posterior mean for $d = 6$. The figure shows that PF needs on the order of 10^4 particles to be as accurate as the first order LGF, and about 10^6 particles to match the second order LGF. Notice that if we allow the LGFs and the PF to have the same accuracy, LGF-1 is about 1000 times faster than the PF, and LGF-2 is expected to be about 10,000 faster than the PF.

The fifth row of Table 5.2 shows the MISE between the true state and the actual posterior mean. The error in using the actual posterior mean to estimate the true state is the statistical error, that is, the error inherent to the system's stochastic characteristics and not the error due to approximations. Note that the statistical error is an order of magnitude larger than the approximation error in the LGF, so that increasing the accuracy with which the posterior expectation is approximated does little to improve the estimation of the state. The approximation error in the PFs however, becomes on the same order as the statistical error when the state dimension is larger ($d = 20$ or 30). In such cases the inaccuracy of the PF will produce comparatively inaccurate estimates of the true state.

LGF accuracy when varying the number of neurons

In this simulation we aimed at isolating the effect of increasing the number of neurons ($N = 20, 50, 100, 200$) in the accuracy of the LGF, and at comparing the accuracy of LGF with Particle Filtering with 100 particles.

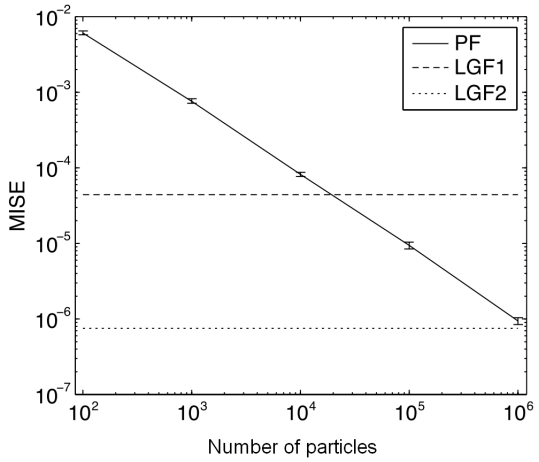


Figure 5.4: Scaling error: Laplace Gaussian Filter vs Particle Filter when the state dimensionality is $d = 6$. The solid line represents the scaling of mean integrated squared error (MISE, vertical axis) for the PF as a function of the number of particles (horizontal axis). Error here is with respect to the actual posterior expectation (optimal filter). The dashed and dotted lines represent the MISE for the first- and second-order LGF, respectively.

We considered the simple model 5.7 with state dynamics given by Equation 5.9. We generated the neural parameters as follows: $\alpha_i = 2.5 + \mathcal{N}(0, 1)$, β_i uniformly distributed on the unit sphere in $\mathbb{R}^3$, and η_i drawn from $\mathcal{N}(0, \frac{\pi}{2})$. We assumed the neural observations to be a Poisson process with intensity given by $(\lambda_i \cdot \Delta)$ where $\Delta = 0.05$ and $T = 100$ time steps. We set the velocity to be $\boldsymbol{v}_t = \left(sin\frac{2\pi}{T}t, sin\frac{2\pi}{T}t, cos\frac{2\pi}{T}t\right)$ and obtained the position at each time step by integration. We learned σ^2 as explained in Section 5.2.6 and assumed the other parameters to be known.

Figure 5.5 shows the filters' MISE in approximating the actual posterior mean as a function of the number of neurons. The first observation is that all filters increase their accuracy as the number of neurons increases. The behavior of the LGF matches what is predicted in theory by Equation 5.12.

The LGF-2 gives the best approximation, followed by the LGF-1, and then by the particle filtering with 100 particles. These results demonstrate that the LGFs give a fast approximation of the posterior that is provably accurate.

Note that the MISE between the true state and the actual posterior mean is 0.064 ± 0.0033 for 100 neurons. The error in using even the optimal filter, i.e., the actual posterior mean, to estimate the true state is orders of magnitude larger than the approximation errors; most of the filtering error is inherent statistical error of the posterior itself, and not due to the approximations. Thus, the first-order LGF was enough for decoding the actual state with the model we used here.

5.2.8 Real data analysis

We applied the LGF and the simple model developed in Section 5.2.6 to data from a 3-D center out experiment from the Schwartz lab. A monkey was presented with a virtual 3-D cube, containing eight possible targets corresponding to the edges of the cube and a cursor that the monkey could control through a multi-

MISE	State dimensionality d			
Method	6	10	20	30
LGF-2	0.0000008	0.000002	0.00001	0.00006
LGF-1	0.00003	0.00004	0.0001	0.0002
PF-100	0.006	0.01	0.03	0.04
PF-scaled	0.006	0.007	0.01	0.02
Posterior	0.03	0.04	0.06	0.07

Table 5.2: Mean-integrated squared errors (MISEs) for different filters as a function of the dimensionality of the state. The first four rows give the discrepancy between three approximate filters and the optimal filter (approximation error): LGF-2 and -1 (second and first order LGF); PF-100, a particle filter with 100 particles; PF-scaled, a particle filter scaled in time to LGF-2. The fifth row gives the MISE between the true state and the estimate of the optimal filter, i.e., the actual posterior mean (statistical error). All values are means of 10 independent replicates and the standard errors are not reported because they are all smaller than the leading digit in the table.

MISE	State dimensionality d			
Method	6	10	20	30
LGF-2	0.24	0.43	1.0	2.0
LGF-1	0.018	0.024	0.032	0.056
PF-100	0.18	0.18	0.18	0.19
PF-scaled	0.18	0.50	0.81	1.8

Table 5.3: Time in seconds needed to decode for different filters as a function of the dimensionality of the state. All values are means of 10 independent replicates and the standard errors are not reported because they are all smaller than the leading digit in the table.

electrode implanted in their motor cortex. The task was to move the cursor to reach a high-lighted target; the monkey received a reward upon successful completion.

In total 78 distinct neurons were recorded simultaneously in the session. The session consisted of 104 trials. Our data each trial consist of time series of spike-counts from these neurons, along with the recorded hand positions, and hand velocities found by taking differences in hand position at successive $\Delta = 0.03$ second intervals. Each trial contained 23 time steps on average.

For decoding, we used the simple state-space model defined by Equations 5.7 and 5.9. Sixteen trials consisting of two presentations of each of the eight targests were used for estimating the parameters of the model. The response function parameters of the neurons, α_i and $\boldsymbol{\theta}_i$, were estimated by Poisson regression of spike counts on cursor position and velocity [6]. These parameters were then used by an EM algorithm to estimate the variance of the state process, σ^2 as explained in Section 5.2.6. Having estimated all the parameters, cursor motions were reconstructed from spike trains for the other 88 trials. For comparison, we also reconstructed the cursor motion with a PF with 100 particles and PVA.

[6] The preferred movement vectors were estimated by Poisson regression instead of an EM method in order to have a fair comparison; the same preferred movements were used in LGF, PF, and PVA.

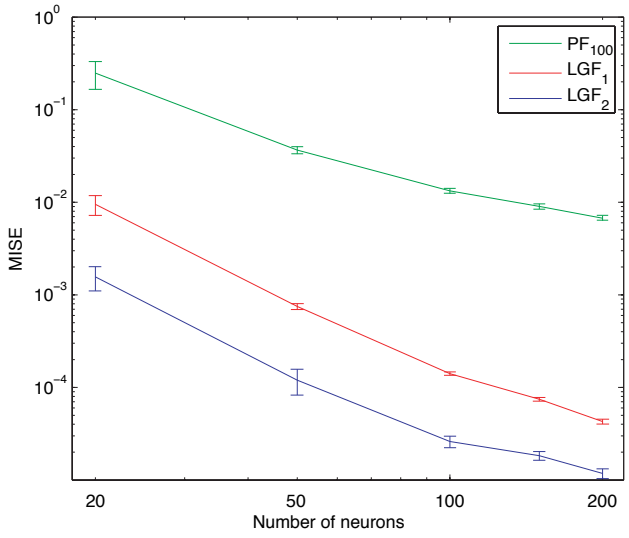


Figure 5.5: Sensitivity to number of neurons: LGF vs Particle Filtering. Horizontal axis: number of neurons; Vertical axis: MISE for the filters with respect to the actual posterior expectation (optimal filter obtained with 10 independent replications of 10^6 particles).

Time lag. Note that the time lag between the hand movement and each neural activity was estimated from the training data by fitting a model over different values of time lag ranging from 0 to 3Δ s. The estimated optimal time lag was the value at which the model had the highest R^2.

Figure 5.6 part (a) shows the comparison between the two versions of the LGF. Note that, despite the second-order LGF being more accurate than the first-order LGF, their approximations are very similar in practical terms since the approximation error is much smaller than the statistical error (as discussed in the simulation section). However, the comparison between LGF-1 and PF-100 (Figure 5.6 part (b)) shows that for most of the trials the LGF yields better results than the particle filter. Finally, Figure 5.6 part (c) illustrates that the numerical error in the PF-100 is of the same order as the error resulting from using PVA. Figure 5.7 displays the results for a single trial and illustrates the observations we made. Overall, the LGF reconstructs the cursor motion better than the PF or PVA.

5.2.9 Discussion and main contributions

In state-space methods, we should distinguish between modelling and computation. The Laplace Gaussian Filter is not about modeling, instead, it is a method for efficiently approximating the recursive computation. The LGF cannot thus be compared to the linear Gaussian model or to OLE; instead, it should be compared against Particle Filtering or with using a Gaussian model versus a Poisson model, that is, using a linear versus a non-linear tuning function.

Hence, the main contribution in this section and in (Koyama et al., 2010a, 2008) is the development of the LGF as a deterministic approximation to the posterior mean. This approximation is guaranteed, under certain regularity conditions, not to accumulate error along time. In the context of neural decoding of hand

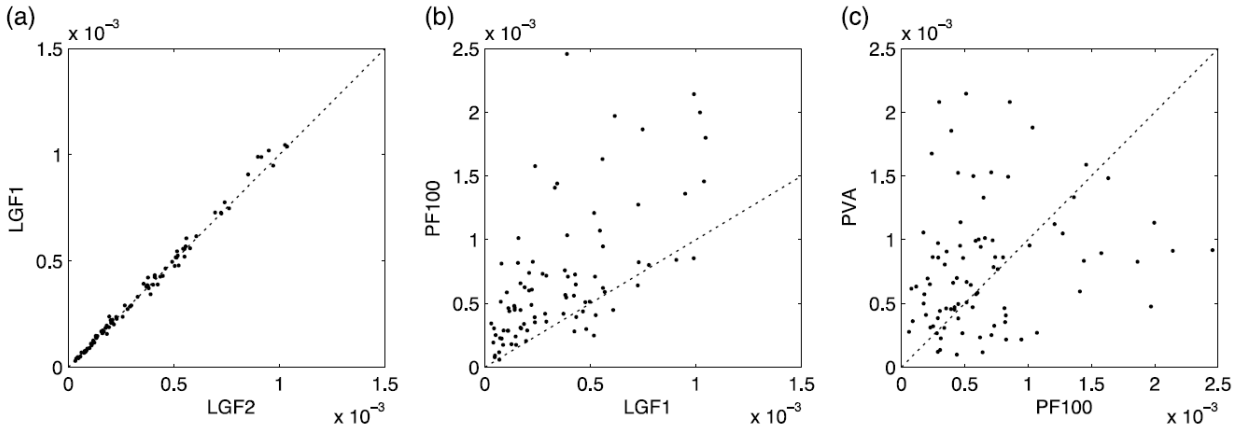


Figure 5.6: Algorithm comparisons: LGF-1, LGF-2, PF-100 and PVA. The x- and y-axis represent the MISE of different algorithms in estimating the true cursor position. Each point compares two different algorithms for a trial. (a) Comparison LGF-1 vs LGF-2; (b) LGF-1 vs PF-100; (c) PF-100 vs PVA

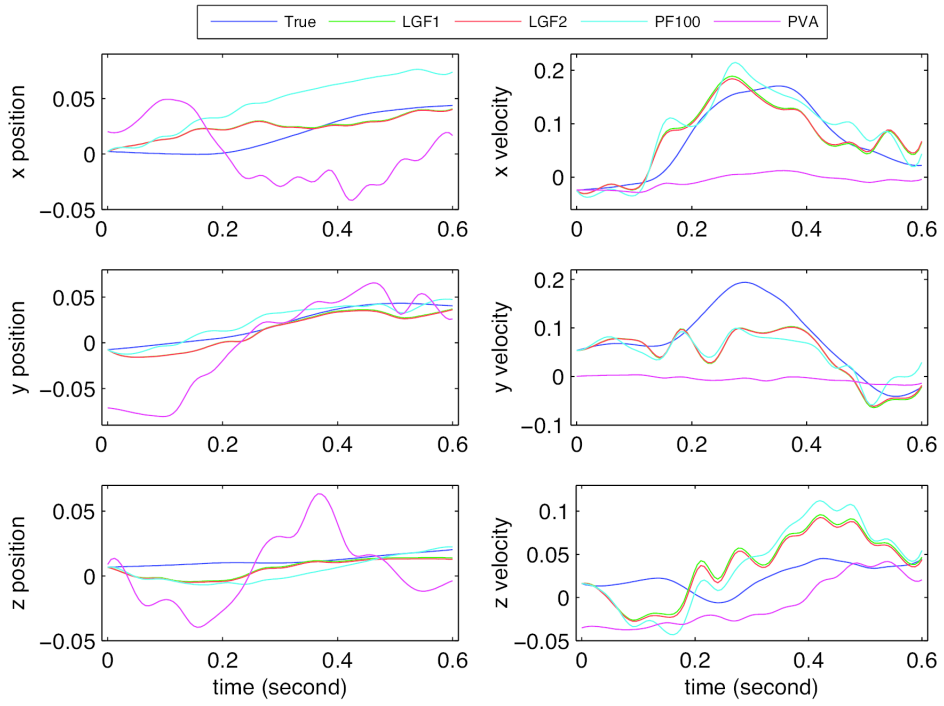


Figure 5.7: The cursor trajectory of a trial and its filtered versions. Left: the cursor position. Right: the cursor velocity. *True*: actual trajectory. LGF-1 and LGF-2: trajectory as estimated by first- and second- order LGF, respectively. PF-100: trajectory estimated by particle filters with 100 particles. PVA: trajectory estimated by the population vector algorithm.

kinematics we found in simulation results that the LGF is much more accurate than the particle filter with the same computational cost. And, in fact, for the 6-dimensional case approximately 10^4 particles are required in the PF to yield comparable results than the first order LGF; and approximately 10^6 particles are needed to match the second order LGF accuracy. The LGF was shown to be much more accurate than the PVA in our off-line real data analysis. In this case, the first order and second order LGF yielded very similar results in terms of accuracy, however, the second order LGF did imply a higher computational cost. Therefore, in the case of neural decoding, our results suggest that using the first order LGF is the best approach.

With regards simplicity of the implementation of the method potential caveats are the maximization of the log-likelihood, the calculation of the Hessian matrix of the log-likelihood and the evaluation of the latter in its maximum. However, the use of numerical methods like Newton's method for optimization and the iterated version of the *Richardson extrapolation* (Section 7.8.2, (Kass, 1987)) avoid the challenges of finding analytic solutions and provide accurate approximations.

One disadvantage of the LGF when contrasted with other methods like particle filtering is that LGF can only be applied when the posterior distribution is unimodal and it would not work in the multimodal case like particle filtering would.

For state space models, the best-known state estimator is the Kalman filter. For truly linear and Gaussian systems, this *is* the optimal filter. But if the system is not linear or non Gaussian (like neural systems are thought to be (Tuckwell, 1989)) a different approach needs to be taken. The simplest non-linear filter, the *extended Kalman filter* (EKF) (Ahmed, 1998), linearizes the state dynamics and the observation function around the current state estimate $\hat{x}$, assuming Gaussian distributions for both. The error thus depends on the strength of the quadratic nonlinearities and on the accuracy of preceding estimates, and so error can accumulate dramatically. The LGF makes no linear approximations — every filtering step is a (generally simple) nonlinear optimization — nor does it need to approximate either the state dynamics or the observation noise as Gaussians.

The *unscented Kalman filter* (UKF) (Julier and Uhlmann, 1997) is another extension of the Kalman filter, which uses a deterministic sampling technique known as the unscented transform to pick a minimal set of sample points (called "sigma points") around the mean. These sigma points are then propagated through the non-linear functions and the covariance of the estimates is recovered. The unscented transformation is more accurate than linearization of non-linear functions, but there is no guarantee that repeatedly applying the unscented transformation does not accumulate errors across time.

Particle filtering, as a stochastic approximation to the integrals Equation 5.1–5.2 that the LGF series expands, is a more natural point of comparison. Just as the error of Laplace's method shrinks as $\gamma \to \infty$, so too does the error of particle filtering vanish as the number of particles grows. In fact, while γ for the LGF is set by the system, the number of particles can be made as large as desired. The biggest drawback of particle filtering is that it needs many particles, which makes it slow: the accuracy grows only sub-linearly with the number of particles, but the time complexity is super-linear in the number of particles (Doucet et al., 2001). In our simulation study, the particle filter needed 10^4 particles to be as accurate as the LGF. The interaction among the particles, via the denominator in Equation 5.1 (Del Moral and Miclo, 2000), makes it hard to parallelize; and it seems any Monte Carlo filter will hit the same obstacles.

Finally, there have been many more or less systematic proposals for nonlinear filters where the poste-

rior distribution is approximated by a density from a well-behaved and low-dimensional parametric family (Azimi-Sadjadi and Krishnaprasad, 2005; Brigo et al., 1995), especially Gaussian distributions (Brown et al., 1998; Eden et al., 2004; Wu et al., 2006), often in some kind of combination with particle filtering. Few of these come with any kind of theoretical guarantee of the validity over time of the approximation, or the higher-order accuracy of the fully exponential Laplace approximation.

In this section we introduced the Laplace Gaussian filter for state estimation. At each step, applying the order-α Laplace approximation to the posterior expectation introduces an $O(\gamma^{-\alpha})$ error, but the recursion propagates the last step's error to higher orders of $1/\gamma$, so that if our initial approximation of the posterior density is accurate to order $O(\gamma^{-\alpha})$, we retain that order of accuracy even after infinitely many steps. (An analogous result appears to hold for the posterior mean, as opposed to the posterior density.) Thus, our theorems show that Laplace's method can be used to approximately evaluate the filtering integrals without compounding the error over time. Our application studies showed that the LGF is able, in a model system, to decode hand motion from neural spike trains, and provided a better estimate, at comparable computational cost, than the particle filter or the population vector algorithm. In fact, most of the error in the LGFs' estimates of the hand motion would be present even in the optimal filter. That is, statistical error dominates approximation error. In practice, other sources of error, e.g. model mis-estimation or even mis-specification, can be even more important than statistical error, and it is far from clear that taking pains to eliminating the approximation error will yield meaningful improvements, whether in neural decoding or other fields. By combining accuracy with tractability, the LGF adds a valuable tool to the kit of the state-space modeler.

5.3 Summary and main contributions

In this chapter we discussed neural-kinematic decoding, the problem of translating neural data into a representation of the kinematic variables.

Grasping is described by a large number of degrees of freedom, and as such, Bayesian decoding of continuous grasp configuration hinges on the availability of several tens of recorded neurons. In Section 5.1 we proposed an approach to (1) summarize grasping through a small set of interpretable variables – a type of dimensionality reduction; to (2) summarize grasping through a set of discrete relevant grasping events based on the interpretable variables, and to (3) decode those events with the activity of a single neuron. We showed that there is a large proportion of neurons in our dataset that reliably decode these discrete grasping events.

One of the difficulties when decoding continuous data in the Bayesian setting is that of calculating the posterior distribution, especially when dealing with high dimensions or with nonlinear or non-Gaussian assumptions. In Section 5.2 we addressed these issues by proposing a deterministic and efficient way of approximating the posterior distribution. We showed, in simulations and in real data, that our approximation yields better results in the off-line setting as compared to Particle Filtering and the traditional PVA.

Chapter 6

Discussion

Despite a century of research, the neural mechanisms that enable finger and grasping movements in primates are largely unknown. The primate hand forms a complex anatomical structure, containing over twenty kinematic degrees-of-freedom. And while it requires complex, coordinated control, endows its owner with an astounding range of dexterous finger movements. In this thesis, we provided models for understanding hand kinematics and algorithms capable of kinematic decoding.

Finger kinematic modelling. One working hypothesis in the neuroscience community is that the central nervous system controls finger kinematics by utilizing kinematic synergies, or correlated joint movements, thus reducing the effective number of degrees of freedom nnecessary for simultaneous control. Previous work (Santello et al., 1998; Todorov and Ghahramani, 2004; Mason et al., 2001, 2004; Soechting and Flanders, 1997; Pesyna et al., 2011; Thakur et al., 2008) has shown that low-dimensional representations of the arm and hand successfully capture most of the joint movement variability during grasping behaviors. These low-dimensional representations are usually estimated using standard matrix factorization approaches, such as principal components analysis (PCA). While useful, these techniques have some shortcomings:

- They are constrained to capturing only linear correlations,
- They confound temporal variability with both experimental and kinematic variability,
- They do not take advantage of the repeated trial structure of most experiments,
- They do not account for variation due to different speeds in the performance of the task.

Vinjamuri et al. (2007, 2010a) obtained temporal synergies by fitting a convoluted mixture model. This approach is similar to dictionary learning (from Computer Vision), but does not provide a generative model from grasping.

We specifically addressed modelling non linearities in the finger variables in Chapter 3 where we neutralized time variation by focusing on relevant landmarks that we defined based in the total kinetic energy of a trial. We found that modelling non linear kinematics resulted in more informative synergies throughout the evolution of the reach-to-grasp movements, as measured by predicting which object was grasped. We also found some evidence of better neural encoding of non linear synergies for specific neurons, albeit not at a population level.

However, while these non linear synergies capture the non linear relationships in the data that allow for better object classification, they lack interpretability and we lose the ability to reconstruct reach-to-grasp movements. Therefore, we proposed a generative model for grasping that is fully interpretable and that is capable of reconstructing reach-to-grasp movements accurately. We showed that through the interpretable learned parameters of the model we can characterize shared variation among replications and also understand how specific replications differ between each other. This essentially corresponds to decoupling sources of variation by leveraging the experimental design. We showed that critical points in the space of learned synergies associated with individual replications have a clear correspondence in the space of hand configurations, and we illustrated how our MGPFM is a tool that can be used to understand the evolution of reach-to-grasp movements. Another methodological contribution of our model, together with the decomposition and dimensionality reduction of the variation, is to use the total kinetic energy of a trial to statistically align the different multivariate replications of the reach-to-grasp movements. We showed that this strategy has direct impact on the reconstruction ability of the model and contributes to advantage over PCA. The reason for this gain is that the aligning procedure helps to emphasize kinematic correlations by explaining out differences in speeds of execution of the reach-to-grasp movements.

Decoding: a discrete framework for interpretable grasping features. To obtain a continuous decoding of kinematic information from neural data, we typically need simultaneous recordings of many neurons. However, it is illuminating to ask the question of how much information of grasping can one robustly extract from one single neuron. To study this question we designed a number of interpretable hand engineered variables that summarized grasp, such as, fist opening, finger curling and finger spread. We then used a Bayesian classifier in a discrete decoding framework to predict critical events of the grasp such as maximum hand aperture. Our analysis showed sound evidence that there exist neurons that preferentially decode digit related events (after having controlled for arm events). And in some cases we obtained up to 96% of accuracy for some neurons (where chance level is 50%).

Decoding: an algorithm for continuous decoding, the Laplace Gaussian Filter. In the continuous decoding setting of linear Gaussian systems, the Kalman Filter is known to be the optimal filter, but neural systems are thought to be non linear and non Gaussian. To perform inference with these more complex distributions the particle filter is a natural choice, but it requires significant computation which could potentially be limiting in real time systems. We presented the Laplace Gaussian Filter as an alternative to Monte Carlo methods to address the intractability of the integrals to obtain the posterior mean by providing a deterministic approximation. We showed that our method can be applied in the continuous decoding setting, and that our method outperforms particle filtering in accuracy and accurately decodes cursor trajectories in a 3D center-out-task. Combining accuracy with computational efficiency our LGF approach adds a valuable tool to the kit of the state-space modelers. There remain, however, a number of directions for future work– one disadvantage of the LGF is that it can only be applied to unimodal posterior distributions. Another consideration in the realm of BCI is that the subjects adaptively correct for algorithmic biases (Chase et al., 2009), raising the question whether it is so important to be highly accurate.

Chapter 7

Appendix

7.1 Related work: hand synergies, neural-motor decoding

Static hand synergies					
Reference	Task	Subject	Dataset, coordinate system	Method	Results
(Santello et al., 1998)	hold imaginary object	human	15 JA: flexion of all fingers and thumb. No abduction.	PCA	Number components for explaining 85% variance: 2; for 95%, 4.
(Todorov and Ghahramani, 2004)	specific manipulation	human	15 JA: as before. Consider three different angle standarizations. Report average of number of components across these standarizations.	PCA	Number components for explaining for 85% of variance: 6.6; for 95%, 9.6.
	specific manipulation	human	20 JA: position. Standarized in the three ways.	PCA	Number components for explaining 85% of variance: 7.3; for 95%, 11.
(Mason et al., 2001)	reach-to-grasp	human	3D marker positions	SVD	1st principal component explained 97.3% of variance. The 2nd component 1.9%.
(Mason et al., 2004)	reach-to-grasp	monkey	3D marker positions	SVD	1st principal component explained 93% of variance. The 2nd component 5%.
(Soechting and Flanders, 1997)	skilled activity	human	11 JA: no thumb included. MCP flexion, DIP flexion, and abduction between 4 fingers.	PCA	Number components for explaining 90% of variance: 4.
(Thakur et al., 2008)	unconstrained haptic exploration	human	3D marker positions	PCA	Number components for explaining 90% of variance: 7.
Dynamic hand synergies					
Reference	Task	Subject	Dataset, coordinate system	Approach	
(Vinjamuri et al., 2007, 2010a,b)	reach and grasp	human	JA velocities	Dictionary based approach. Convolutive-mixture model learned through an SVD and an ℓ_1 optimization step.	

Table 7.1: Previous work on dimensionality reduction for extracting hand synergies. It is important to mention Principal Component Analysis and Singular Value Decomposition are essentially the same. JA stands for joint angles, and 3D for the three dimensional position of markers.

Tables 7.2, 7.3, 7.4, 7.5 show a representative survey on related work on on-line continuous and discrete decoding and in off-line continuous and discrete decoding. In these tables *intracortical* refers to single or multiple spikes unless otherwise specified.

On-line continuous decoding					
Organism	Neural-motor area	Neural variables	Kinem. variables	Method	Reference
Rat	M1 and ventrolateral thalamus	Intracortical	lever movement timing and magnitude– 1D control	PCA neural activity, use 1 pc into an ANN	(Chapin et al., 1999)
Rhesus monkey	M1, PMd, PC	Intracortical	1D/3D manipulandum	Linear model / Non linear ANN	(Wessberg et al., 2000)
Rhesus monkey	M1	Intracortical	2D cursor position	Linear estimator	(Serruya et al., 2002)
Rhesus monkey	M1	Intracortical	3D cursor reaching. visual feedback, open and closed loop	PVA	(Taylor et al., 2002)
Rhesus monkey	M1, PMd, SMA, S1, PPC	Intracortical	2D cursor grasping force, hand position, velocity.	Linear filters	(Carmena et al., 2003)
Rhesus monkey	PPC	Intracortical	trajectory of a cursor	Bayesian decoder KF including target information, also tried linear decoders (least-squares and ridge regression)	(Mulliken et al., 2008)
Rhesus monkey	M1	Intracortical	robotic arm trajectory and gripper	PVA	(Velliste et al., 2008)
Human being		Intracortical (neurotrophic electrode) – 1 or 2 units!	horizontal position of cursor on screen to reach and select letters on screen keyboard (vertical direction with EMG on toe muscle)	Linear filter	(Kennedy et al., 2000; Kennedy and Bakay, 1998)
Human being		EEG	2D cursor control	*Adaptive algorithm*: horizontal and vertical location predicted as a linear combination of weighted values of relevant frequencies, these change over time.	(Wolpaw and McFarland, 2004)
Human being	M1	Intracortical	2D cursor control, movement intent, various tasks; grasp with a robotic limb	Linear filter	(Hochberg et al., 2006)
Human being	M1	Intracortical	cursor velocity and position (the later not as successfully)	Bayesian decoder KF and linear filter (the later with less accuracy)	(Kim et al., 2008)
Human being	Parietal lobe, temporal lobe and posterior portion of the frontal lobe of the left hemisphere	ECoG	virtual hand with 10 degrees of freedom	Convolutional model	(Vinjamuri et al., 2011)
Human being	M1	Intracortical	reach and grasp robotic arm/hand	Bayesian decoder KF	(Hochberg et al., 2012)
Human being	M1, S1	ECoG	3D cursor movement	OLE	(Wang et al., 2013)

Table 7.2: On-line continuous decoding – related work.

On-line discrete decoding					
Organism	Neural-motor area	Neural variables	Kinem. variables	Method	Reference
Rhesus monkey	Medial intraparietal area, PPC, PMd	Intracortical	intended target selection	Bayesian decoder	(Musallam et al., 2004)
Rhesus monkey	PMd	Intracortical	discrete target selection on screen	Maximum likelihood (generative) – Gaussian and Poisson spike models	(Santhanam et al., 2006)
Rhesus monkey	AIP (intra parietal cortex), PMv	Intracortical	two grasp types power and precision grip and 5 wrist orientations (discrete)	Bayesian decoder – Poisson spike model	(Townsend et al., 2011)
Human being	Over left frontal-parietal-temporal cortex	ECoG	1D cursor - binary movement up or down the screen (discrete) (and off-line 2D direction)	Selection of electrodes and frequency bands highly correlated with imaginary training tasks- use those to predict cursor location	(Leuthardt et al., 2004)
Human being	M1	Intracortical (and stability after 100 days)	2D cursor velocities and point-and-click (continuous and discrete)	Bayesian decoder KF (for velocities); linear discriminant classifier (for click intentions)– Gaussian spike model	(Kim et al., 2011),(Simeral et al., 2011)

Table 7.3: On-line discrete decoding – related work.

Off-line continuous decoding					
Organism	Neural-motor area	Neural variables	Kinem. variables	Method	Reference
Rhesus monkey	M1	Intracortical	direction	PVA	(Georgopoulos et al., 1986b)
Rhesus monkey	M1	Intracortical	2D velocity, position	Bayesian decoder KF	(Wu et al., 2002, 2003, 2006)
Rhesus monkey	M1	Intracortical	2D velocity, position	Bayesian decoder switching KF	(Wu et al., 2004)
Rhesus monkey	M1	Intracortical	arm trajectories	Bayesian decoder PF	(Brockwell et al., 2004)
Rhesus monkey	M1 and PMd	Intracortical	selection and planning of discrete movement classes and/or postures followed by the execution of continuous limb trajectories (discrete and continuous)	Continuous: linear filter and ANN. Discrete: Maximum likelihood classifiers - Poisson (better performance) and Gaussian spike models; and ANN.	(Hatsopoulos et al., 2004)
Rhesus monkey	PMd and M1	Intracortical	full trajectory	Bayesian decoder MTM - Gaussian spike model	(Yu et al., 2007)
Rhesus monkey	M1	Intracortical	hand aperture	Bayes decoder with hidden variables – Gaussian spike model	(Artemiadis et al., 2007)
Rhesus monkey	M1	Intracortical	position of fingers and wrist while flexing and extending – movement constrained by pistol-grip manipulandum. Asynchronous decoding ie automatic identification of time of event	Linear filter, ANN, KF	(Aggarwal et al., 2009)
Rhesus monkey	M1	Intracortical (LFP high frequency band and multi unit activity)	3D arm position and velocity, and grip aperture in a reach and grasp task	Bayesian decoder KF	(Zhuang et al., 2010)
Rhesus monkey	M1	Intracortical (spike trains, LFP)	arm, wrist and hand postures during movement (25 PCA-ed joint angles)	Bayesian decoder (a single joint angle at a time)	(Vargas-Irwin et al., 2010)
Rhesus monkey	M1 and PMv	Intracortical (spiking activity and LFP several bands)	3D arm position and velocity, and grip aperture in a reach and grasp task	Bayesian decoder KF	(Bansal et al., 2012, 2011)
Human being		ECoG	grasp apperture in a reach-grasp-hold task of objects varying in shape and size	Generalized linear models	(Fifer et al., 2011)
Human being	M1	Intracortical (spiking activity, LFP and multi unit activity)	intended movement of specific joints: shoulder (elevation, rotation) , elbow (flex-ext), wrist (flex-ext), forearm (pron-supination), hand (open-close); seen on a screen being performed by a virtual character – subject had to imagine performing the movement	Linear system identification. Goodness of fit: accounted variance	(Ajiboye et al., 2012)

Table 7.4: Off-line continuous decoding – related work.

Off-line discrete decoding					
Organism	Neural-motor area	Neural variables	Kinem. variables	Method	Reference
Rhesus monkey	M1	Intracortical	movement direction (discrete)	Bayesian classifier – Gaussian spike model including correlations. Also k-NN not as good.	(Maynard et al., 1999)
Rhesus monkey	PPC (parietal reach region)	Intracortical	movement direction (discrete)	Maximum likelihood decoder (generative) Poisson spike model	(Shenoy et al., 2003)
Rhesus monkey	M1	Intracortical (multi unit activity – without explicit spike detection)	prehension- grasp type and direction of movement (discrete) or tracing movements – 2D velocities (continuous)		(Stark and Abeles, 2007)
Rhesus monkey	M1	Intracortical	flexion of single (or pairs of) fingers (discrete) – movement constrained by pistol-grip manipulandum	a variation of PVA, logistic regression (or two layered ANN), softmax estimator (and variation with information theory)	(Hamed et al., 2007)
Rhesus monkey	M1	Intracortical	flexion and extension of fingers and wrist – movement constrained by pistol-grip manipulandum (discrete). Asynchronous decoding ie automatic identification of time of event	ANN	(Aggarwal et al., 2008; Acharya et al., 2008)
Rhesus monkey	M1	Intracortical	flexions and extensions of fingers (discrete) - movement constrained by pistol-grip manipulandum	Bayes classifier (MAP)– Poisson model for neural activity	(Baker et al., 2009)
Rhesus monkey	M1	Intracortical	single and multiple finger flexions, wrist flexions (discrete) - movement constained by pistol-grip manipulandum	Maximum likelihood - Skellam model for neural activity	(Shin et al., 2010, 2009a,b)
Rhesus monkey	PMv (area F5)	Intracortical	types of grips (discrete)	SVM, k-NN, ANN (discriminative)	(Carpaneto et al., 2011, 2012)
Rhesus monkey	M1	Intracortical	Single or multiple fingers flexions or extensions (discrete) – movement constrained by a pistol-grip manipulandum	Ad-hoc method: identify neurons tuned for movemnt, voting scheme	(Egan et al., 2011, 2012)
Rhesus monkey	M1 and PMd	Intracortical	four grasp types (objects) and a resting state (discrete on the core, but concatenated in time to produce trajectory)	fuzzy k-NN + finite state machine	(Hao et al., 2012)
Rhesus monkey	M1	Intracortical	objects that were grasped (discrete) in a reach-and-grasp task	SVM, k-NN (discriminative)	(Xu et al., 2013) - this is a method for selecting neurons

Table 7.5: Off-line discrete decoding – related work.

7.2 Hand kinematic model - joint angles derivation

As input we have 3D marker location as indicated with red segments in Figure 7.1. As output we seek:

- three flexion joint angles per finger (index, middle, ring and pinky): $\theta_1, \theta_2, \theta_3$ from proximal to distal;
- θ_2^{thumb} and $(\theta_1^{thumb}, \varphi)$ to describe thumb joint angles, where the last two angles are represented in spherical coordinates to describe the most proximal joint for the thumb. The representation in spherical coordinates helps representing rotation ability of thumb.
- four angles measuring finger spread or separation $\alpha_{n,m}$ between fingers n and m.

We start by finding the flexion angles for all fingers (refer to Figure 7.1 right panel). We denote T the thumb, I the index, L middle finger (as in largest), R ring finger, and P pinky.

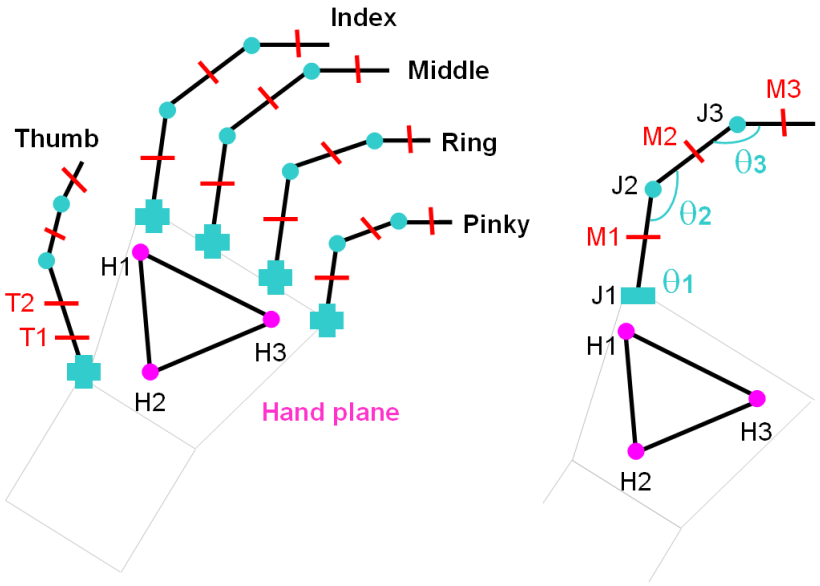


Figure 7.1: *(Left panel)* Diagram showing the markers positions indicated by red line segments. *(Right panel)* Sketch for obtaining flexion angles of index finger. M denotes marker position; J denotes joint position; H denotes hand marker position. We know the coordinates for all M, and H. Problem: determine values of θ, which represent the joint angle flexion/extension. The finger moves on the *finger plane*. Points H determine the *hand plane*.

Find θ_2 and θ_3 for I, L, R, P; and $\theta 2$ for T We use the *law of cosines*, which states: $c^2 = a^2 + b^2 - 2\,a\,b\,cos(\theta)$.

In our case: a denotes the distance between Mi and $J\,(i+1)$; b denotes the distance between $J\,(i+1)$ and $M\,(i+1)$; and c denotes the distance between Mi and $M\,(i+1)$ for $i = 1, 2$

Find $\theta 1$ for I, L, R, P To do this we first need to obtain $J2$ for each digit:

1. Obtain the finger plane. This is fully determined by the vector normal to the plane: $N^{finger} = \frac{(M1-M2)\times(M3-M2)}{\|(M1-M2)\times(M3-M2)\|}$. Note that N^{finger} points to the right of the fingers (palms down, no matter what hand).

2. Consider the two dimensional finger plane, and let $M1$ to be the origin. Goal: find (x, y) the intersection between two circles:

   (a) the circle centered in the origin with diameter $d_{M1,J2}$ where $d_{i,j}$ denotes the Euclidean distance between i and j:
   $$x^2 + y^2 = d^2_{M1,J2}$$

   (b) the circle centered in $(\|M2 - M1\|, 0)$ is described by the equation:
   $(x - \|M2 - M1\|)^2 + y^2 = d^2_{M2,J2}$.

   The intersection between these two circles is given by:

   - $x = \frac{(d^2_{M2,J2} - d^2_{M1,J2}) - \|M2-M1\|^2}{-2\|M2-M1\|}$ when $\|M2 - M1\| \neq 0$
   - $y = \pm\sqrt{d^2_{M1,J2} - x^2}$ selecting the physiologically correct, that is, the one with the positive sign.

3. Then $J2 = \vec{a} + \vec{b}$ where $\vec{a} = x\left(\frac{M2-M1}{\|M2-M1\|}\right)$ and $\vec{b} = y\left(N^{finger} \times \frac{M2-M1}{\|M2-M1\|}\right)$. Refer to Figure 7.2.

We now have $J2$ the 3D position of the second joint of the finger. To calculate θ_1 we consider the *hand plane* and the *finger plane*, and find their intersection.

The joint angle θ_1 is the angle between the intersecting line and the line on the finger plane that passes through $M1$ and $J2$.

The *hand plane* is defined by the normal vector: $N^{hand} = \frac{(H3-H2)\times(H1-H2)}{\|(H3-H2)\times(H1-H2)\|}$. Note that the normal vector to the hand points to the opposite direction of hand closure.

Next, we need to find the intersection between the two planes defined as: $\begin{cases} N^{hand} \cdot (v - H1) = 0 \\ N^{finger} \cdot (v - M1) = 0 \end{cases}$

This intersection is $v^* = k + t(N^{hand} \times N^{finger})$, where k is a point on the intersection line. Note that the vector $N^{hand} \times N^{finger}$ points towards the tip of the finger. We can then conclude that:

$$\theta_1 = arccos\left(\frac{(J2 - M1) \cdot v^*}{\|J2 - M1\| \ \|v^*\|}\right)$$

Find separation for I, L, R, P Instead of defining an angle of abduction-adduction per finger, we chose to define a measure of stretch of the hand, or finger aperture. This aperture is defined as the angle between the two planes defined by the markers of two contiguous fingers. Consider two contiguous fingers n and m, and their respective defining normal vector N^{finger_n} and N^{finger_m}. Then, the angle between the finger planes $\alpha_{n,m}$ is obtained by:

$$\alpha_{n,m} = arccos\left(N^{finger_n} \cdot N^{finger_m}\right)$$

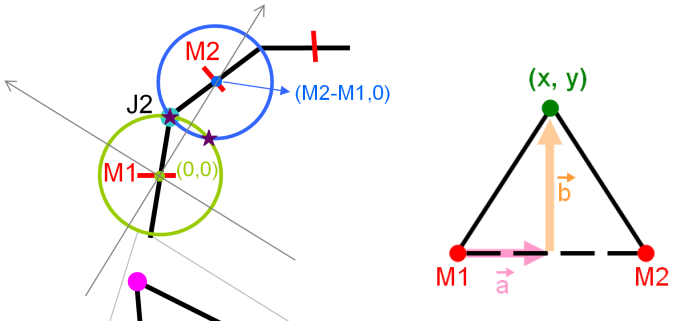


Figure 7.2: Diagram to calculate J2 position. (*Left panel*) $M1$ and $M2$ are known, $J2$ needs to be determined. An auxiliary coordinate system is defined with origin centered in $M1$. This system allows to describe with simple equations two circumferences: (a) the circumference with center in $M1$ that intersects $J2$, and (b) the circumference with center in $M2$ that intersects $J2$. These circumferences intersect, by definition, in $J2$ and in a point that is physiologycally infeasible. (*Right panel*) This virtual triangle lies on the finger plane.

Find a description of the proximal joint for T. The angle θ_1 as defined for I, L, R, P is not suitable to define the thumb mechanics, due to the rotation that the thumb presents. We use spherical coordinates to describe the kinematics of the proximal joint of the thumb.

We consider the system of coordinates shown in Figure 7.3 and define $\vec{T}$, the vector on the thumb plane as $\vec{T} = \frac{T2-T1}{\|T2-T1\|}$.

We need to convert the markers coordinates into a suitable reference coordinate system, namely:

$$(x, y, z) = (left/right, hand, up)$$

This translates into the following expression for each of the hands:

$$(x,y,z)_{right} = \left(\vec{T}\cdot N^{left}, \vec{T}\cdot N^{hand}, \vec{T}\cdot N^{up}\right), \quad (x,y,z)_{left} = \left(\vec{T}\cdot N^{right}, \vec{T}\cdot N^{hand}, \vec{T}\cdot N^{up}\right)$$

for right and left hand respectively, where

$$N^{up} = \frac{\frac{1}{2}(H1+H3)-H2}{\left\|\frac{1}{2}(H1+H3)-H2\right\|}; \quad N^{right} = \frac{N^{up}\times N^{hand}}{\|N^{up}\times N^{hand}\|}; \quad N^{left} = \frac{N^{hand}\times N^{up}}{\|N^{hand}\times N^{up}\|}.$$

Finally, we obtain θ_1^{thumb} and φ in spheric coordinates (see Figure 7.4). The angles θ_1^{thumb} and φ describe the proximal joint for T. And are obtained as follows:

$$\theta = arctan\left(\frac{y}{x}\right); \quad \varphi = arcos\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right).$$

7.3 Datasets for static synergies appendix

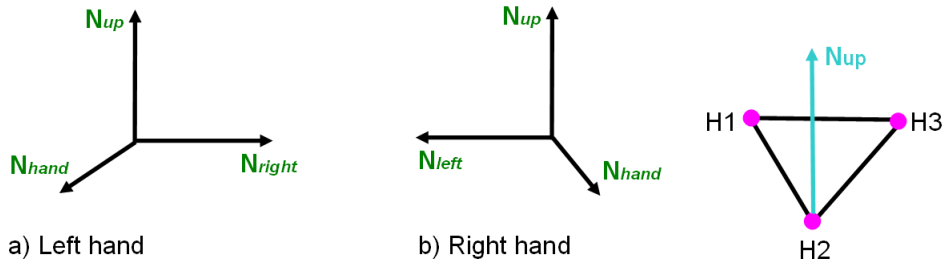


Figure 7.3: Coordinate system for describing the mechanics of the proximal joint of the thumb. Scheme a) corresponds to the left hand, and N_{right} refers to the direction towards which the thumb points. The opposite holds for the right hand (Scheme b). On the right most panel, we show the definition of N^{up}. We define $H1$ as the marker placed on the left-most part of the glove when the palm is facing downwards.

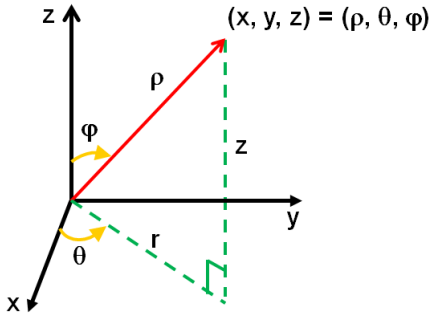


Figure 7.4: Spherical coordinates.

Baxter

Total count of (Objects, Orientation)

↓ Obj / Pos →	1	2	3	4	5	6	7	Sum
1	19	13	18	20	19	0	0	89
2	4	4	2	0	7	7	8	32
3	0	0	0	0	0	0	0	0
4	5	8	1	7	6	5	2	34
5	17	14	15	15	16	16	14	107
6	18	19	20	17	19	19	15	127
7	10	11	9	1	12	13	10	66
8	11	13	8	9	15	14	9	79
9	15	0	0	17	18	14	11	75
10	15	0	0	13	15	13	7	63

Mean count of (Objects, Orientation) per session

↓ Obj / Pos →	1	2	3	4	5	6	7
1	4.75	3.25	4.50	5.00	4.75	0.00	0.00
2	1.00	1.00	0.50	0.00	1.75	1.75	2.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	1.25	2.00	0.25	1.75	1.50	1.25	0.50
5	4.25	3.50	3.75	3.75	4.00	4.00	3.50
6	4.50	4.75	5.00	4.25	4.75	4.75	3.75
7	2.50	2.75	2.25	0.25	3.00	3.25	2.50
8	2.75	3.25	2.00	2.25	3.75	3.50	2.25
9	3.75	0.00	0.00	4.25	4.50	3.50	2.75
10	3.75	0.00	0.00	3.25	3.75	3.25	1.75

Table 7.6: Baxter: count and mean of number of trials per condition (object, orientation).

Vinny

Total count of (Objects, Orientation)

↓ Obj / Pos →	1	2	3	4	5	6	7	Sum
1	60	55	51	55	58	0	0	279
2	31	28	41	0	33	23	17	173
3	54	28	58	13	50	61	24	288
4	63	39	69	60	60	67	41	399
5	59	14	52	52	47	57	54	335
6	72	67	69	65	70	65	69	477
7	62	53	62	60	64	60	52	413
8	65	56	64	56	65	64	62	432
9	12	26	5	7	20	0	0	70
10	67	64	43	56	70	0	0	300

Mean count of (Objects, Orientation) per session

↓ Obj / Pos →	1	2	3	4	5	6	7
1	3.75	3.44	3.19	3.44	3.63	0.00	0.00
2	1.94	1.75	2.56	0.00	2.06	1.44	1.06
3	3.38	1.75	3.63	0.81	3.13	3.81	1.50
4	3.94	2.44	4.31	3.75	3.75	4.19	2.56
5	3.69	0.88	3.25	3.25	2.94	3.56	3.38
6	4.50	4.19	4.31	4.06	4.38	4.06	4.31
7	3.88	3.31	3.88	3.75	4.00	3.75	3.25
8	4.06	3.50	4.00	3.50	4.06	4.00	3.88
9	0.75	1.63	0.31	0.44	1.25	0.00	0.00
10	4.19	4.00	2.69	3.50	4.38	0.00	0.00

Table 7.7: Vinny: count and mean of number of trials per condition (object, orientation).

7.4 Appendix Esteban datasets

In Figure 7.5 we show a visual analysis of which sessions are suitable for analysis of simultaneous neurons based on the number of valid trials and simultaneous neural recordings, and in Figure 7.6 we display a glimpse on the heterogeneity of the population of recorded neurons based on their modulation during the reach-to-grasp movement. Figure 7.5

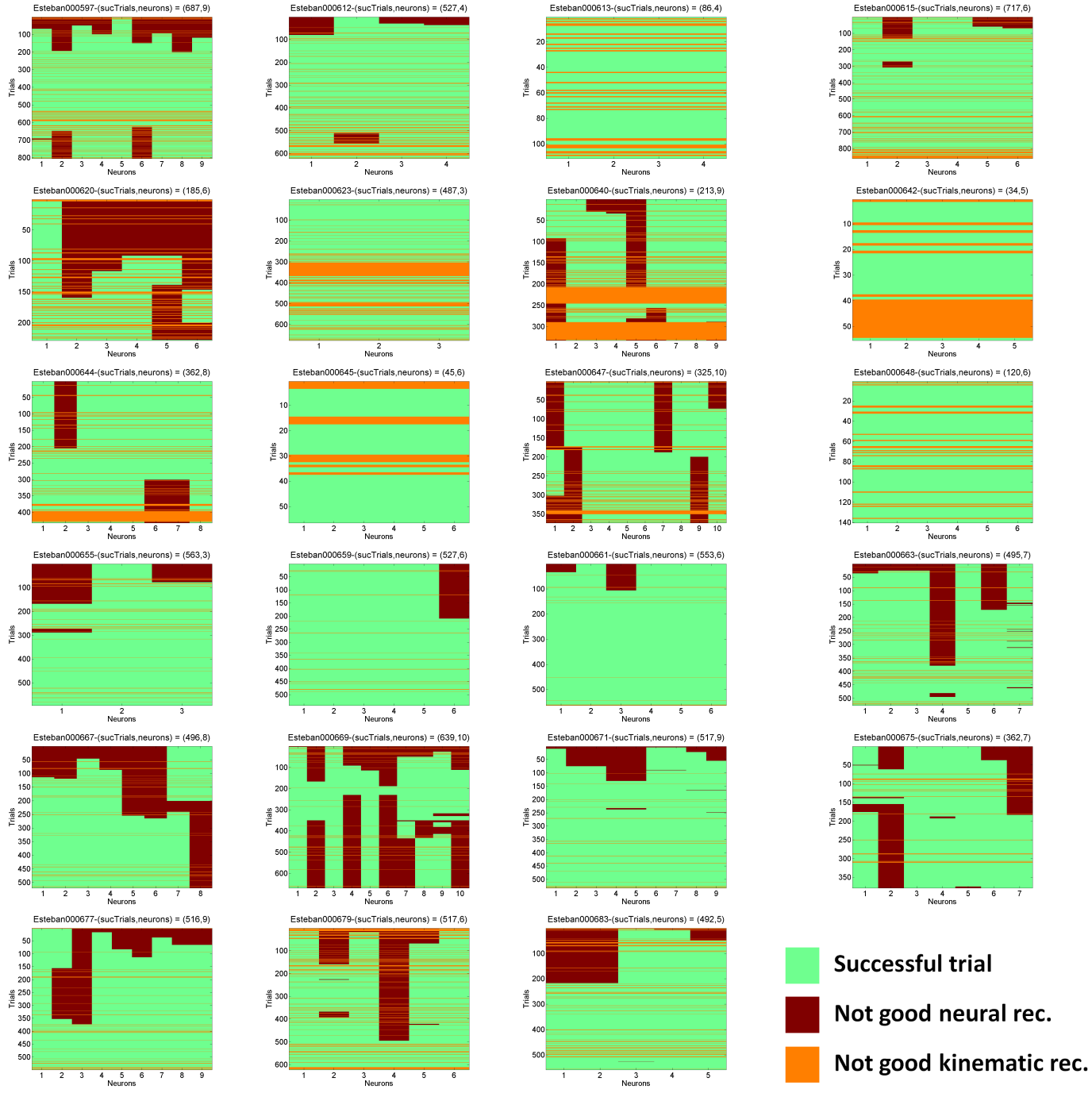


Figure 7.5: Successful trials and neural recordings from Esteban. Each plot corresponds to a session: on the y-axis there are trials, on the x-axis neurons. Trials with good kinematics are on green, with bad kinematics on orange; and trials where a specific neural unit was not recording are red. With this matrix visualization one can determine which sessions have many valid kinematic trials that have associated several neural units simultaneously.

7.5 Methods for dimensionality reduction and classification

In this section, that corresponds to Chapter 3 we mention and briefly explain the methods used to (1) obtain the static synergies through the application of dimensionality reduction methods; and (2) evaluate the synergies through classification of objects.

7.5.1 Dimensionality reduction methods

Dimensionality reduction techniques deal with the problem of finding or building a *small* number of components that describe most of the information that the original data set has. In other words, dimensionality reduction consists of transforming a dataset X with dimensionality K into a new dataset Y with dimensionality k where $k < K$, while retaining as much as possible some important properties of the data, like variance structure or geometry.

According to the geometric assumptions made on the data, there are linear and non-linear techniques for dimensionality reduction. Linear techniques assume that the data lie on or near a linear subspace of a high-dimensional space. Non-linear techniques do not rely on the linearity assumption, and thus a more complex embedding of the data in the high-dimensional space can be identified.

Principal Component Analysis (PCA). (Algorithm 1) PCA finds a low dimensional representation of the data points that best preserves their variance as measured in the high dimensional input space, this is equivalent to assuming that the original data set lies on or near a linear hyperplane. The principal components are the eigenvector basis of the covariance matrix, that is, a spectral decomposition of the covariance matrix. The computational complexity of this operation is $O(m^2\,K + K^3)$ where n is the number of observations and K is the number of considered variables or dimensionality of observations.

If the structure of the dataset is not linear, PCA will not account for all of the structure. A *kernelized* version of PCA proposed by Schölkopf et al. (1998) can extract more information by using suitable non linear features.

Kernelized PCA (kPCA). (Algorithm 2) This method is PCA generalized through the *kernel trick* (Schölkopf et al., 1998), which consists of substituting Euclidean dot products in the space of input patterns by generalized dot products in a large dimensional *feature space*. The procedure takes as input a specific non linear kernel function, applies it to the input observations (mapping the data into a possibly high-dimensional space), and finds principal components which are not linearly related to the input space. The idea is that the low dimensional hidden structure might be easier to discover in the feature space. This procedure also needs a spectral decomposition, but in this case, of the kernel matrix. Therefore the complexity of kernel PCA is $O(m^3)$ where m is the number of training examples. That is, the complexity of kPCA scales cubically with the number of training examples and usually the number of observations is very large with respect to the number of variables.

The selection of a kernel is rather arbitrary. Three classical examples of kernels are the linear kernel $\mathcal{K}(x_i, x_j) = \langle x_i \cdot x_j \rangle$ (which is equivalent to linear PCA), the polynomial kernels $\mathcal{K}(x_i, x_j) = (1 + \langle x_i \cdot x_j \rangle)^p$ for $p \in \mathbb{Z}^+$ and the Gaussian kernels[1] $\mathcal{K}(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{\sigma^2}\right)$. The linear kernel identifies the fea-

[1] $\|x\| = \sqrt{\langle x \cdot x \rangle}$

Figure 7.6: Heterogeneity of neural population from Esteban. Each bar represents the firing rate of the neuron in an epoch: baseline, pre movement, reach and grasp. In these plots there are neurons whose activity is modulated to different extent to reaches, to grasps, to both reaches and grasps, or to none of them.

ture space with the input space, the polynomial kernel maps the inputs into a feature space of dimensionality $O(K^p)$, and the Gaussian kernel maps the inputs onto the surface of an infinite-dimensional sphere.

Linear discriminant analysis (LDA). PCA (and kPCA) are unsupervised methods of dimensionality reduction. If X is a data set whose points belong to a specific class, then it is possible to apply dimensionality reduction on X taking into account the classes of the data points (supervised dimensionality reduction). A classic method to reduce dimensionality in a supervised way is Fisher Linear Discriminant Analysis (LDA). In LDA the objective is to maximize the ratio of between class variability over within class variability.

Fisher Linear Discriminant analysis maximizes the function $J(w) = \frac{w' S_B w}{w' S_W w}$, where S_B is the between class scatter matrix and S_W is the within class scatter matrix. The scatter matrices are defined as $S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T$ and $S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T$ where c indexes over the classes, $\bar{x}$ is the overall mean of the data, and μ_c the mean of the data that belongs to class c. The scatter matrices are proportional to the covariance matrices, and thus solutions using the scatter matrices or the covariance matrices are the same. The maximization problem is solved using the Lagrangian and KKT conditions, and it is reduced to a spectral decomposition as in PCA.

7.5.2 Classification of objects – goodness of synergies

A way of evaluating the goodness of the construction of the synergies is to examine the ability to predict what object is going to be grasped. In the next paragraphs we briefly explain the classification methods we considered. The kinematic data set consists of a set of vectors describing the configuration of the hand at the selected time point. Each of these vectors is associated with an object the monkey is grasping. This setting corresponds naturally to a supervised classification task, where the aim is to learn a function that assigns each vector to one of a finite number of discrete categories, in this case, the objects. Therefore we can take advantage of the framework we have, and use classification accuracy to *indirectly* quantify the goodness of the dimensionality reduction in the sense that we get an idea of how much *information* we preserve regarding the object being grasped. However, it is important to mention that good dimensionality reduction does not imply good classification accuracy, and good classification accuracy does not imply good dimensionality redution. Two classic examples are shown in Figure 7.7. Furthermore, the results might be classifier dependent, that is limited by the classifier assumption.Therefore, the results of classification should only be taken as an indication of information being preserved, but not proof.

In order to perform a classification task, a classifier must be trained. We chose two classifiers to be trained: a discriminative and a generative one, and we briefly explain them below.

Naive Bayes classifier. The Naive Bayes classifier is a supervised generative classifier based on Bayes rule. It makes the assumption that the considered variables are conditionally independent given the class to which the input data belongs. This assumption is made to reduce the complexity of general Bayesian classifiers from $O(2^n)$ to $O(n)$ (Mitchell, 1997).

Consider $\mathcal{O}$ a random variable indicating the object that is being grasped in a specific trial, and $X \in \mathbb{R}^{N \times K}$ the matrix of joint angles. The Naive Bayes classifier models the joint probability of observed variables given the object class as:

$$P(X_1, \ldots, X_K | \mathcal{O}) = \prod_{i=1}^{K} P(X_i | \mathcal{O}).$$

input : $X \in \mathbb{R}^{N \times K}$ where $N =$ number of observations, and $K =$ number variables,
[Optional: $x \in \mathbb{R}^{1 \times K}$ *test example*]
output: $Y \in \mathbb{R}^{N \times k}$,
[Optional: $y \in \mathbb{R}^{1 \times k}$ *k-dimensional encoding of test example,*
$\hat{x}$ *reconstruction of test example*]

1. Prepare data X for PCA

1.1 Center data X
$\tilde{X} \leftarrow X - \texttt{mean}(X)$

1.2 Obtain covariance matrix C of centered data
$C \leftarrow \tilde{X}^T \tilde{X}$ where $C \in \mathbb{R}^{K \times K}$

2. Compute the spectral decomposition of C
$\Lambda \leftarrow \texttt{Eigenvalues}(C)$ where Λ is a diagonal matrix in $\mathbb{R}^{K \times K}$
$V \leftarrow \texttt{Eigenvectors}(C)$

2.1 Obtain the intrinsic linear dimensionality of the data k
$k \leftarrow \texttt{numberTopEigenvalues}(\Lambda)$

2.2 Build the matrix with the k eigenvectors corresponding to the top eigenvalues, and the diagonal matrix of top k eigenvalues
$\tilde{V} \leftarrow V^{K \times k}$
$\tilde{\Lambda} \leftarrow \Lambda^{k \times k}$

2.3 Obtain the square root element by element of the diagonal matrix of top k eigenvalues $\tilde{\Lambda}$
$S_{\tilde{\Lambda}} \leftarrow \texttt{sqrt}(\tilde{\Lambda})$ where $S_{\Lambda} \in \mathbb{R}^{k \times k}$ and is diagonal

3. Compute k-dimensional embedding of training data
$Y \leftarrow \frac{1}{\sqrt{K}} X \tilde{V} S_{\tilde{\Lambda}}$

[Optional 4.] Encode test example
4.1 Obtain the matrix $J_{S_{\tilde{\Lambda}}}$ that contains in the diagonal the inverse of the elements of the diagonal of $S_{\tilde{\Lambda}}$
$J_{S_{\tilde{\Lambda}}} \leftarrow \texttt{diagInv}(S_{\tilde{\Lambda}})$

4.2 Obtain encoding of the test example
$y \leftarrow x \tilde{V} J_{S_{\tilde{\Lambda}}}$

Algorithm 1: Principal Component Analysis (PCA)

input : $X \in \mathbb{R}^{N \times K}$ where $N =$ number of observations, and $K =$ number variables; k the low dimension, $\mathcal{K}$ *a kernel selected a priori*, [Optional: $x \in \mathbb{R}^{1 \times K}$ a *test example*]
output: $Y \in \mathbb{R}^{N \times k}$,
[Optional: $y \in \mathbb{R}^{1 \times k}$ *k-dimensional encoding of test example*]

0. Compute kernel on data
$\mathcal{K} \leftarrow \mathcal{K}(X, X)$ where $\mathcal{K} \in \mathbb{R}^{N \times N}$

1. Compute normalized kernel $\tilde{\mathcal{K}}$ in the feature space
$\tilde{\mathcal{K}} \leftarrow (I - \frac{1}{N}\,\bar{1}\,\bar{1}^T)\,\mathcal{K}\,(I - \frac{1}{N}\,\bar{1}\,\bar{1}^T)$

2. Compute the spectral decomposition of $\tilde{\mathcal{K}}$
$\Lambda \leftarrow \texttt{Eigenvalues}(\tilde{\mathcal{K}})$ where Λ is a diagonal matrix in $\mathbb{R}^{N \times N}$
$V \leftarrow \texttt{Eigenvectors}(\tilde{\mathcal{K}})$

2.1 Build the matrix with the k eigenvectors corresponding to the top eigenvalues, and the diagonal matrix of top k eigenvalues
$\tilde{V} \leftarrow V^{N \times k}$
$\tilde{\Lambda} \leftarrow \Lambda^{k \times k}$

3. Compute k-dimensional embedding of training data
3.1 Obtain the square root element by element of $\tilde{\Lambda}$
$S_{\tilde{\Lambda}} \leftarrow \texttt{sqrt}(\tilde{\Lambda})$ where $S_{\tilde{\Lambda}} \in \mathbb{R}^{k \times k}$ and is diagonal

3.2 Obtain the low dimensional representation of the training data
$Y \leftarrow \tilde{V} S_{\tilde{\Lambda}} X$

[Optional 4.] Encode test example
4.1 Obtain the matrix that contains in the diagonal the inverse of the elements of the diagonal of $S_{\tilde{\Lambda}}$
$J_{S_{\tilde{\Lambda}}} \leftarrow \texttt{diagInv}(S_{\tilde{\Lambda}})$

4.2 Obtain normalized kernel evaluated in the test point
4.2.1 Compute kernel vector: apply input kernel function on training data and test point
$\mathcal{K}_x \leftarrow \mathcal{K}(X, x)$ with $\mathcal{K}_x \in \mathbb{R}^{N \times 1}$

4.2.2 Normalize kernel vector
$\tilde{\mathcal{K}}_x \leftarrow \mathcal{K}_x - \bar{1} \cdot \sum_{i=1,\ldots,N} \mathcal{K}_{xi} - B + \bar{1}\left(\frac{1}{N^2} \sum_{i=1,\ldots,N} \sum_{j=1,\ldots,N} \mathcal{K}_{i,j}\right)$ where $\bar{1} \in \mathbb{R}^{N \times 1}$,
$B \in \mathbb{R}^{N \times 1}$ with its i-th coordinate: $B_i = \sum_{\alpha=1,\ldots,N} \mathcal{K}_{\alpha,i}$

4.3 Obtain encoding of the test example
$y \leftarrow (J_{S_{\tilde{\Lambda}}} \cdot \tilde{V}^T \cdot \tilde{\mathcal{K}}_x)^T$

Algorithm 2: kernel Principal Component Analysis (kPCA)

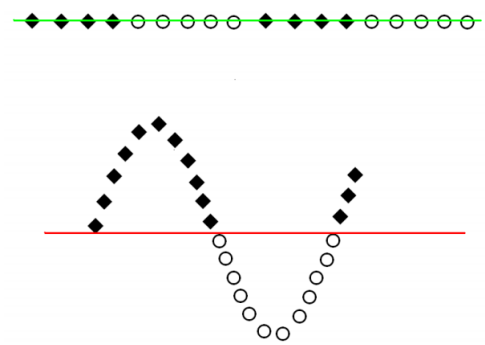


Figure 7.7: Good dimensionality reduction does not imply good classification, nor vice versa. In the example at the top the green line yields the best dimensionality reduction, but the classification is bad. In the panel at the bottom the classification accuracy of the red line is excellent, but clearly the dimensionality reduction should yield a sine function, as oppose to the straight line which determines the best classification.

By Bayes rule, the Naive Bayes classifiation procedure indicates that the object to be assigned to the new reach and grasp trial is given by:

$$O \leftarrow argmax_{o_j} P(O = o_j) \prod_{i \in Objects} P(X_i | \mathcal{O} = o_j).$$

We make the assumption that each X_i is normally distributed, and is defined by the mean and the variance specific to the variable X_i and the class o_j. The way of obtaining the mean and the variances of each Gaussian is through the expressions:

$$\mu_{ij} = \mathbb{E}(X_i | \mathcal{O} = o_j), \;\; \sigma_{ij}^2 = \mathbb{E}((X_i - \mu_{ij})^2 | \mathcal{O} = o_j).$$

And the priors on $\mathcal{O}$ can be estimated as: $\pi_j = P(\mathcal{O} = o_j)$

We selected this classifier because of its simplicity, low complexity and because it has been applied successfully in applications in spite of the conditionally independent assumption being invalid.

Multi-Class Support Vector Machine. Suport Vector Machines (Boser et al., 1992) were developed from statistical learning theory and do not assume any probabilistic model for the data. They were theoretically motivated, and posed as an optimization problem where the idea is to maximize the margin between the class boundary and the training patterns. The function to minimize is an additive combination of training error and a complexity term. A convenient property is that the optimization problem to be solved is convex (thus there is no local minima), and is solved using classical optimization theory. The resulting classification function only depends on the (few) training examples that are closest to the decision boundary, the so-called *support vectors*, and thus the solution is sparse.

The classical SVM theory was developed for binary problems. We have a multi-class problem. A traditional extension from a binary classifiaction framework to a multi-class framework is to decompose the problem into multiple independent binary class tasks (or the one-versus-all approach). In our case we chose

to use the implementation of Tsochantaridis et al. (2005) where the notion of separation margin is generalized (Crammer and Singer, 2001), the multiclass classification is framed as a constrained optimization problem with a quadrative objective function with a potentially prohibitive number of constraints, but where the problem is solved through a cutting plane algorithm in polynomial time.

7.6 Percentage of variability explained by PCA

Number components needed to explain specific percentage of variance

Criter. outliers: 1 – Averaging fixing (object, orientation)

Time of Interest	Baxter 85%	Vinny 85%	Baxter 95%	Vinny 95%
firstBef	4	3	7	6
lastBef	3	3	5	6
max	2	4	4	6
firstAft	2	3	5	5
lastAft	2	3	5	6

Criter. outliers: 1 – Sampling

Time of Interest	Baxter 85%	Vinny 85%	Baxter 95%	Vinny 95%
firstBef	4	5	8	8
lastBef	5	5	8	9
max	4	5	7	9
firstAft	4	4	7	7
lastAft	4	5	8	8

Criter. outliers: 2 – Averaging fixing (object, orientation)

Time of Interest	Baxter 85%	Vinny 85%	Baxter 95%	Vinny 95%
firstBef	4	3	6	6
lastBef	3	4	4	6
max	2	4	4	6
firstAft	2	3	5	5
lastAft	2	4	5	6

Criter. outliers: 2 – Sampling

Time of Interest	Baxter 85%	Vinny 85%	Baxter 95%	Vinny 95%
firstBef	5	5	8	8
lastBef	4	5	7	9
max	4	5	7	9
firstAft	4	4	7	8
lastAft	4	5	7	8

Table 7.8: Number components needed to explain specific percentage of variance for each of the experimental conditions (see Table 3.3).

7.7 Dynamic synergies appendix

7.7.1 Supplement for simulation studies

We show results of two exemplary simluations. We generated $R = 80$ samples for training and 500 samples for testing. We used the following learning settings: we initialized the parameters with the MLE of the matrix-normal distribution, we assumed $\boldsymbol{\mu}$ to be modelled with B-splines and stopped the algorithm after 50 iterations. We learned three models: the first one modelling the observed data only with the mean (as a baseline), the last two corresponding to the MGPFM assuming Σ free and Σ constrained.

In Figure 7.8 (top panel) we show the true latent process X together with two dimensions of the observed Y and estimated $\hat{Y}$ for the two MGPFM models. Note that we obtain smoother estimates when constraining Σ. In the middle and bottom panels of Figure 7.8 we show error profiles for the three models. The baseline model (that disregards the MGP term) results in significantly worse estimates as compared to either setting for the MGPFM. In addition, by constraining Σ we are able to remove all unaccounted structure left in the residuals when modelling Σ free.

7.7.2 Supplement for reach-to-grasp data analysis

Figure 7.9 shows more visualizations of the columns of **B** under different experimental conditions. Unlike PCA, in which one obtains *canonical* directions of movement, these visualizations only exemplify the space of possible configurations of change of movement in the dataset.

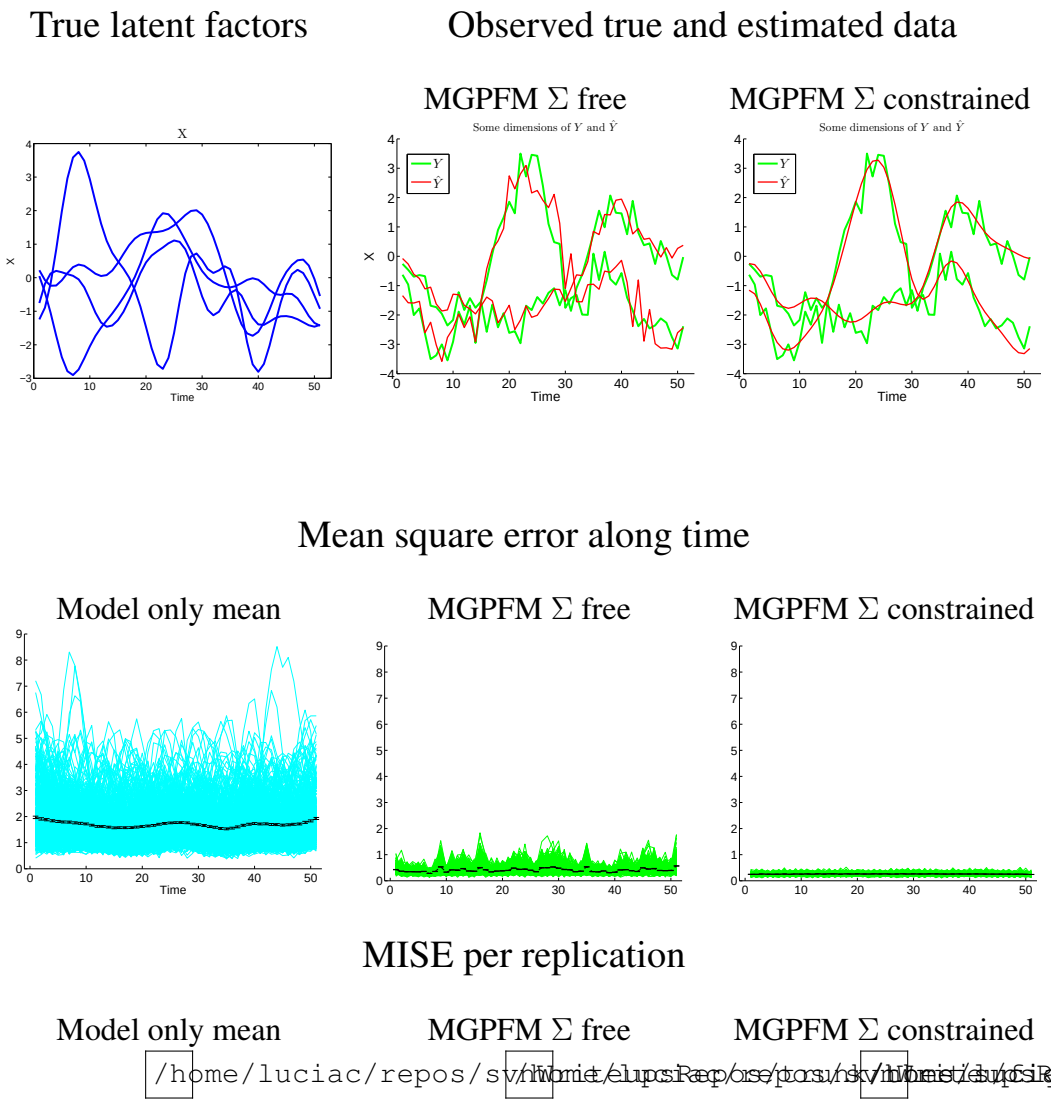


Figure 7.8: Results of two exemplary simulations. (Top panel) True latent process X, and two dimensions of the 50-dimensional Y and $\hat{Y}$. Estimates of $\hat{Y}$ are smoother when Σ is constrained. (Middle and bottom panel) Error profiles for three models: baseline when modelling only the mean (left), MGPFM with Σ free (middle) and MGPFM with Σ constrained (left). Best results are achieved with the MGPFM when Σ is constrained.

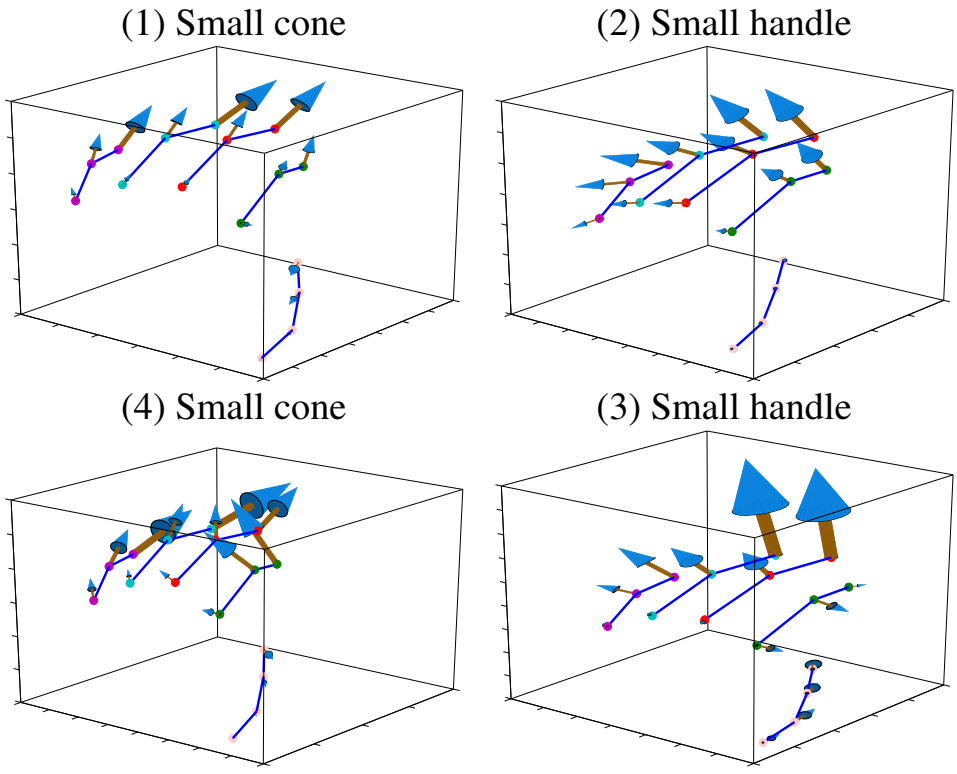


Figure 7.9: Visualization of the columns of the factor loading matrix $\hat{\mathbf{B}}$ in selected conditions. These visualizations exemplify some of the ways that a replication can differentiate itself from others in the same data set. (1) and (2) two types of grasp opening, the former through extension of fingers corresponding to interphalangeal joint angle extension and the latter through metacarpophalangeal joint angle extension; (3) Markers of two fingers move significantly faster than the others in a non-synchronized grasping movement; (4) Complex movement corresponding to curling fingers around a cone.

7.8 Laplace Gaussian Filter appendix

7.8.1 Laplace's method

We consider the following integral,

$$I(\gamma) = \int g(x)e^{-\gamma h(x)}dx, \tag{7.1}$$

where $x \in \mathbb{R}$; γ, the expansion parameter, is a large positive real number; $h(x)$ and $g(x)$ are independent of γ (or very weakly dependent on γ); and the interval of integration can be finite or infinite. There is a computationally efficient method to compute the coefficients in the infinite asymptotic expansion of the integral Wojdylo (2006). Suppose that $h(x)$ has an interior minimum at x_0, and $h(x)$ and $g(x)$ are assumed

to be expandable in a neighborhood of x_0 in series of ascending powers of x. Thus, as $x \to x_0$,

$$h(x) \sim h(x_0) + \sum_{s=0}^{\infty} a_s (x - x_0)^{s+2}, \tag{7.2}$$

and

$$g(x) \sim \sum_{s=0}^{\infty} b_s (x - x_0)^s, \tag{7.3}$$

in which $a_0, b_0 \neq 0$.

Consider the two dimensionless sets of quantities, $A_i \equiv a_i/a_0$ and $B_i \equiv b_i/b_0$, as well as the constants $\alpha_1 = 1/a_0^{1/2}$ and $c_0 = b_0/a_0^{1/2}$. Then the integral in Eq. 7.1 can be asymptotically expanded as

$$I(\gamma) \sim c_0 e^{-\gamma h(x_0)} \sum_{s=0}^{\infty} \Gamma(s + \frac{1}{2}) \alpha_1^{2s} c_{2s}^* \gamma^{-s-\frac{1}{2}}, \tag{7.4}$$

where

$$c_s^* = \sum_{i=0}^{s} B_{s-i} \sum_{j=0}^{i} \left(\begin{array}{c} -\frac{s+1}{2} \\ j \end{array} \right) \mathcal{C}_{i,j}(A_1, \ldots) \,, \tag{7.5}$$

where $\mathcal{C}_{i,j}(A_1, \ldots)$ is a partial ordinary Bell polynomial, the coefficient of x^i in the formal expansion of $(A_1 x + A_2 x^2 + \cdots)^j$. $\mathcal{C}_{i,j}(A_1, \ldots)$ can be computed by the following recursive formula,

$$\mathcal{C}_{i,j}(A_1, \ldots) = \sum_{m=j-1}^{i-1} A_{i-m} \mathcal{C}_{m,j-1}(A_1, \ldots) \,, \tag{7.6}$$

for $1 \geq j \geq i$. Note that $\mathcal{C}_{0,0}(A_1, \ldots) = 1$, and $\mathcal{C}_{i,0}(A_1, \ldots) = \mathcal{C}_{0,j}(A_1, \ldots) = 0$ for all $i, j > 0$.

7.8.2 Numerically obtaining second derivatives

Consider calculating the second derivative of $l(x)$ at x_0 for the one-dimensional case. For $n = 0, 1, 2, \ldots$ and $c > 1$, define the second central difference quotient,

$$A_{n,0} = [l(x_0 + c^{-n} h_0) + l(x_0 - c^{-n} h_0) - 2l(x_0)]/(c^{-n} h_0)^2, \tag{7.7}$$

and then for $k = 1, 2, \ldots, n$ compute

$$A_{n,k} = A_{n,k-1} + \frac{A_{n,k-1} - A_{n-1,k-1}}{c^{2(k+1)} - 1}. \tag{7.8}$$

When the value of $|A_{n,k} - A_{n-1,k}|$ is sufficiently small, the approximation $A_{n,k+1}$ is used.

This algorithm is an iterated version of the second central difference formula, often called *Richardson extrapolation*, producing an approximation with an error of order $O(h^{2(k+1)})$ (G. and A., 1974).

In the d-dimensional case of a second-derivarive approximation at a maximum, Kass (1987) proposed an efficient numerical routine which reduces the computation of the Hessian matrix to a series of one-dimensional second-derivative calculations. The trick is to apply the second-difference quotient to suitably-defined functions f of a single variable s, as follows.

1. Initialize the increment $h = (h_1, \ldots, h_d)$.
2. Find the maximum of $l(\boldsymbol{x})$, and call it $\hat{\boldsymbol{x}}$.
3. Get all unmixed second derivatives for each $i = 1$ to d, using the function

$$\begin{aligned} x_i &= \hat{x}_i + s \\ x_j &= \hat{x}_j \quad \text{for } j \text{ not equal to } i \\ f(s) &= l(\boldsymbol{x}(s)). \end{aligned} \tag{7.9}$$

   Compute the second difference quotient; then repeat and extrapolate until the difference in successive approximations meets a relative error criterion, as in Eq. 7.8; store as diagonal elements of the Hessian matrix array, $l''_{i,i} = f''(0)$.
4. Similarly, get all the mixed second derivatives. For each $i = 1$ to d, for each $j = i + 1$ to d, using the function

$$\begin{aligned} x_i &= \hat{x}_i + s/\sqrt{l''_{i,i}} \\ x_j &= \hat{x}_j + s/\sqrt{l''_{j,j}} \\ x_k &= \hat{x}_k \quad \text{for } k \text{ not equal to } i \text{ or } j \\ f(s) &= l(\boldsymbol{x}(s)). \end{aligned} \tag{7.10}$$

   Compute the second difference quotient; then repeat and extrapolate until difference in successive approximations is less than relative error criterion as in Eq. 7.8; store as off-diagonal elements of the Hessian matrix array, $l''_{i,i} = (f''(0)/2 - 1)\sqrt{l''_{i,i} l''_{j,j}}$.

7.8.3 Expectation Maximization Algorithm

The Expectation Maximization algorithm (Dempster et al., 1977; Baum et al., 1970) used to estimate the states and the model parameters through maximization of the likelihood

$$\log p(y_{1:T}; \theta) = \log \int p(x_{1:T}, y_{1:T}; \theta) dx_{1:T} = \log \int \prod_{t=1}^{T} p(x_t | x_{t_1}; \theta) p(y_t | x_t; \theta) dx_{1:T}.$$

1. At the beginning (iteration step $i = 1$), set the initial value of parameter, $\theta_{(1)}$.
2. (E-step) Given the current estimate $\theta_{(i)}$ of θ, compute the posterior expecation of the log joint probability density,

$$Q(\theta | \theta_{(i)}) = E[\log p(x_{1:T}, y_{1:T} | \theta) | y_{1:T}; \theta_{(i)}]. \tag{7.11}$$

3. (M-step) Updated the estimate $\theta_{(i+1)}$ by

$$\theta^{(i+1)} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta | \theta_{(i)}). \tag{7.12}$$

4. Repeat steps 2 and 3 until the estimate converges.

7.8.4 The Population Vector Algorithm

The *population vector algorithm* (PVA) is a standard method for neural decoding, especially for directionally-sensitive neurons like the motor-cortical cells recorded from in the experiments we analyze (Dayan and Abbott, 2001, pp. 97–101). Briefly, the idea is that each neuron i, $1 \leq i \leq N$, has a preferred motion vector $\boldsymbol{\theta}_i$, and the expected spiking rate λ_i varies with the inner product between this vector and the actual motion vector $\boldsymbol{x}(t)$,

$$\frac{\lambda_i(t) - r_i}{\Lambda_i} = \boldsymbol{x}(t) \cdot \boldsymbol{\theta}_i \ , \tag{7.13}$$

where r_i is a baseline firing rate for neuron i, and Λ_i a maximum firing rate. (Eq. 7.13 corresponds to a cosine tuning curve.) If one observes $y_i(t)$, the actual neuronal counts over some time-window Δ, then averaging over neurons and inverting gives the *population vector*

$$\boldsymbol{x}_{\mathrm{pop}}(t) = \sum_{i=1}^{N} \frac{y(t) - r_i\Delta}{\Lambda_i \Delta} \boldsymbol{\theta}_i \ , \tag{7.14}$$

which the PVA uses as an estimate of $\boldsymbol{x}(t)$. If preferred vectors $\boldsymbol{\theta}_i$ are uniformly distributed, then $\boldsymbol{x}_{\mathrm{pop}}$ converges on a vector parallel to $\boldsymbol{x}$ as $N \to \infty$, and is in that sense unbiased (Dayan and Abbott, 2001, p. 101). If preferred vectors are not uniform, however, then in general the population vector gives a biased estimate.

Bibliography

S. Acharya, F. Tenore, V. Aggarwal, R. Etienne-Cummings, M.H. Schieber, and N.V. Thakor. Decoding individuated finger movements using volume-constrained neuronal ensembles in the m1 hand area. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 16(1):15 –23, feb. 2008.

V. Aggarwal, S. Acharya, F. Tenore, Hyun-Chool Shin, R. Etienne-Cummings, M.H. Schieber, and N.V. Thakor. Asynchronous decoding of dexterous finger movements using m1 neurons. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 16(1):3 –14, feb. 2008.

V. Aggarwal, F. Tenore, S. Acharya, M.H. Schieber, and N.V. Thakor. Cortical decoding of individual finger and wrist kinematics for an upper-limb neuroprosthesis. In Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE, pages 4535 –4538, sept 2009.

N. U. Ahmed. Linear and Nonlinear Filtering for Scientists and Engineers. World Scientific, Singapore, 1998.

A Ajiboye, J Simeral, J Donoghue, L Hochberg, and R Kirsch. Prediction of imagined single-joint movements in a person with high level tetraplegia. In IEEE Trans Biomed Eng, pages 2755–65, 2012.

B. D. Anderson and J. B. Moore. Optimal filtering. Prentice-Hall, New Jersey, 1979.

P. K. Artemiadis, G. Shakhnarovich, C. Vargas-Irwin, J. P. Donoghue, and M. J. Black. Decoding grasp aperture from motor-cortical population activity. In Neural Engineering, 2007. CNE '07. 3rd International IEEE/EMBS Conference on, pages 518–521, 2007.

James Ashe and Apostolos P Georgopoulos. Movement parameters and neural activity in motor cortex and area 5. Cerebral Cortex, 4(6):590–600, 1994.

Babak Azimi-Sadjadi and P. S. Krishnaprasad. Approximate nonlinear filtering and its applications in navigation. Automatica, 41:945–956, 2005.

Justin Baker, William Bishop, Spencer Kellis, Todd Levy, Paul House, and Bradley Greger. Multi-scale recordings for neuroprosthetic control of finger movements. In Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE, pages 4573–4577. IEEE, 2009.

Arjun K Bansal, Wilson Truccolo, Carlos E Vargas-Irwin, and John P Donoghue. Decoding 3d reach and grasp from hybrid signals in motor and premotor cortices: spikes, multiunit activity, and local field potentials. Journal of neurophysiology, 107(5):1337–1355, 2012.

Arjun K Bansal, Carlos E Vargas-Irwin, Wilson Truccolo, and John P Donoghue. Relationships among low-frequency local field potentials, spiking activity, and three-dimensional reach and grasp kinematics in primary motor and ventral premotor cortices. Journal of neurophysiology, 105(4):1603–1619, 2011.

G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach, 2000.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. Annals of Mathematical Statistics, 41: 164–171, 1970.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society. Series B (Methodological), 57(1): 289–300, 1995.

Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. The Annals of Statistics, 29(4):1165–1188, 2001.

Otto Bock and Thomas Feix. Emergence of cognitive grasping through emulation, introspection, and surprise (grasp). taxonomy., 2008.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pages 144–152. ACM Press, 1992.

D. Brigo, B. Hanzon, and F. LeGland. A differential geometric approach to nonlinear filtering: The projection filter. In Proceedings of the 34th IEEE conference on decision and control, pages 4006–4011, 1995.

A. E. Brockwell, A. L. Rojas, and R. E. Kass. Recursive bayesian decoding of motor cortical signals by particle filtering. Journal of Neurophysiology, 91(4):1899–1907, 2004.

Emery N Brown, Loren M Frank, Dengda Tang, Michael C Quirk, and Matthew A Wilson. A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. The Journal of Neuroscience, 18(18):7411–7425, 1998.

D William Cabel, Paul Cisek, and Stephen H Scott. Neural activity in primary motor cortex related to mechanical loads applied to the shoulder and elbow during a postural task. Journal of neurophysiology, 86(4):2102–2108, 2001.

Jose M Carmena, Mikhail A Lebedev, Roy E Crist, Joseph E O'Doherty, David M Santucci, Dragan F Dimitrov, Parag G Patil, Craig S Henriquez, and Miguel A. L Nicolelis. Learning to control a brain machine interface for reaching and grasping by primates. PLoS Biol, 1(2):1–16, 10 2003.

J Carpaneto, MA Umiltà, L Fogassi, A Murata, V Gallese, S Micera, and V Raos. Decoding the activity of grasping neurons recorded from the ventral premotor area f5 of the macaque monkey. Neuroscience, 188: 80–94, 2011.

Jacopo Carpaneto, Vassilis Raos, Maria A Umiltà, Leonardo Fogassi, Akira Murata, Vittorio Gallese, and Silvestro Micera. Continuous decoding of grasping tasks for a prospective implantable cortical neuroprosthesis. Journal of neuroengineering and rehabilitation, 9(1):84, 2012.

Lucia Castellanos. Grasping in primates: Mechanics and neural basis. Master's thesis, Carnegie Mellon University. Machine Learning Department, 2010.

Lucia Castellanos, Jonathan Huang, Chance Spalding, Sagi Perel, Andrew B. Schwartz, and Robert E. Kass. Decoding of grasping features from primary motor cortex. Women in Machine Learning Workshop. Neural Information Processing Systems, 2010.

Lucia Castellanos, Chance Spalding, Samuel Clanton, Robert E. Kass, and Andrew B. Schwartz. Preliminary steps toward decoding cortical signals for multidimensional hand movement. Society for Neuroscience Meeting, 2008.

Lucia Castellanos, Vince Vu, Sagi Perel, Andrew B. Schwartz, and Robert E. Kass. A multivariate gaussian process factor model for hand shape during reach-to-grasp movements. Submitted, 2013.

Lillian Y. Chang and Yoky Matsuoka. A kinematic thumb model for the act hand. In ICRA, pages 1000–1005. IEEE, 2006.

John K. Chapin, Karen A. Moxon, Ronald S. Markowitz, and Miguel A. L. Nicolelis. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. Nature Neuroscience, 2:664–670, 1999.

Steven M. Chase, Andrew B. Schwartz, and Robert E. Kass. Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of spiking-based brain-computer interface algorithms. Neural Networks, 22(9):1203–1213, 2009.

Jessie Chen, Shari D. Reitzen, Jane B. Kohlenstein, and Esther P. Gardner. Neural Representation of Hand Kinematics During Prehension in Posterior Parietal Cortex of the Macaque Monkey. J Neurophysiol, 102 (6):3310–3328, 2009.

Mark M Churchland, John P Cunningham, Matthew T Kaufman, Stephen I Ryu, and Krishna V Shenoy. Cortical preparatory activity: representation of movement or first cog in a dynamical machine? Neuron, 68(3):387–400, 2010.

M.M. Churchland, J.P. Cunningham, M.T. Kaufman, J.D. Foster, P. Nuyujukian, S.I. Ryu, and K.V. Shenoy. Neural population dynamics during reaching. Nature, 487(7405):51–6, 2012.

Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pages 3270–3275, 2007.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research, 2:265–292, 2001.

John P Cunningham, Vikash Gilja, Stephen I Ryu, and Krishna V Shenoy. Methods for estimating neural firing rates, and their application to brain–machine interfaces. Neural Networks, 22(9):1235–1246, 2009.

M. R. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. Robotics and Automation, IEEE Transactions on, 5(3):269–279, 1989.

Andrea d'Avella and Emilio Bizzi. Shared and specific muscle synergies in natural motor behaviors. Proceedings of the National Academy of Sciences of the United States of America, 102(8):3076–3081, 2005.

A Philip Dawid. Some matrix-variate distribution theory: notational considerations and a bayesian application. Biometrika, 68(1):265–274, 1981.

Peter Dayan and Laurence F. Abbott. Theoretical Neuroscience. MIT Press, Cambridge, Massachusetts, 2001.

P. de Jong and M. J. Mackinnon. Covariances for smoothed estimates in state space models. Biometrika, 75:601–602, 1988.

P. Del Moral and L. Miclo. Branching and interacting particle systems approximations of Feynman-Kac formulae with applications to non-linear filtering. In J. Azema, M. Emery, M. Ledoux, and M. Yor, editors, Semainaire de Probabilites XXXIV, pages 1–145, Berlin, 2000. Springer-Verlag.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: B, 79:1–38, 1977.

Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. Sequential Monte Carlo Methods in Practice. Springer-Verlag, Berlin, 2001.

Pierre Dutilleul. The mle algorithm for the matrix normal distribution. Journal of Statistical Computation and Simulation, 64(2):105–123, 1999.

Uri T. Eden, L. M. Frank, R. Barbieri, Victor Solo, and Emery N. Brown. Dynamic analyses of neural encoding by point process adaptive filtering. Neural Computation, 16:971–998, 2004.

Joshua Egan, Justin Baker, P House, and Bradley Greger. Decoding dexterous finger movements in a neural prosthesis model approaching real-world conditions. In IEEE Trans on Neural Systems and Rehabilitation Engineering, pages 836–844, 2012.

Joshua Egan, Justin Baker, Paul House, and Bradley Greger. Detection and classification of multiple finger movements using a chronically implanted utah electrode array. In Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE, pages 7320–7323. IEEE, 2011.

A. Erdélyi. Asymptotic Expansions. Dover, New York, 1956.

Ayla Ergün, Riccardo Barbieri, Uri Eden, Matthew A. Wilson, and Emery N. Brown. Construction of point process adaptive filter algorithms for neural systems using sequential monte carlo methods. IEEE Transactions on Biomedical Engineering, 54:419–428, 2007.

M.S. Fifer, M. Mollazadeh, S. Acharya, N.V. Thakor, and N.E. Crone. Asynchronous decoding of grasp aperture from human ecog during a reach-to-grasp task. Conf Proc IEEE Eng Med Biol Soc, 2011, 2011.

A. Fyshe, E.B. Fox, D.B. Dunson, and T.M. Mitchell. Hierarchical Latent Dictionaries for Models of Brain Activation. In Proc. International Conference on Artificial Intelligence and Statistics, April 2012.

Dahlquist G. and Bjorck A. Numerial Methods. Englewood, New Jersey, Prentice Hall, 1974.

Apostolos B. Georgopoulos, JF Kalaska, R Caminiti, and JT Massey. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. The Journal of Neuroscience, 2(11):1527–1537, 1982.

Apostolos B. Georgopoulos, Andrew B. Schwartz, and R. E. Kettner. Neural population coding of movement direction. Science, 233:1416–1419, 1986a.

Apostolos P. Georgopoulos, Ronald E. Kettner, and Andrew B. Schwartz. Primate motor cortex and free arm movements to visual targets in three-dimensional space. ii. coding of the direction of movement by a neuronal population. The Journal of Neuroscience, 8(8):2928–2937, 1988.

Apostolos P. Georgopoulos, Andrew B. Schwartz, and Ronald E. Kettner. Neuronal population coding of movement direction. Science, 233(4771):1416–1419, 1986b.

V. Gilja, P. Nuyujukian, C.A. Chestek, J.P. Cunningham, B.M. Yu, J.M. Fan, M.M. Churchland, M.T. Kaufman, J.C. Kao, S.I. Ryu, and K.V. Shenoy. A high-performance neural prosthesis enabled by control algorithm design. Nat Neurosci, 2012.

Vikash Gilja, Cindy A Chestek, Ilka Diester, Jaimie M Henderson, Karl Deisseroth, and Krishna V Shenoy. Challenges and opportunities for next-generation intracortically based neural prostheses. Biomedical Engineering, IEEE Transactions on, 58(7):1891–1899, 2011.

Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K. Allen. The columbia grasp database. 2009.

S. Ben Hamed, M. H. Schieber, and A. Pouget. Decoding m1 neurons during multiple finger movements. Journal of Neurophysiology, 98(1):327–333, 2007.

Yaoyao Hao, Weidong Chen, Shaomin Zhang, Qiaosheng Zhang, Bo Jiang, Ting Zhao, Xiaoxiang Zheng, et al. Continuous neural decoding of grasp types for asynchronous brain machine interfaces. In Conference proceedings:... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, volume 2012, pages 6422–6425, 2012.

Nicholas Hatsopoulos, Jignesh Joshi, and John G. O'Leary. Decoding continuous and discrete motor behaviors using motor and premotor cortical ensembles. Journal of Neurophysiology, 92(2):1165–1174, 2004.

Nicholas G. Hatsopoulos, Qingqing Xu, and Yali Amit. Encoding of movement fragments in the motor cortex. The Journal of Neuroscience, 27(19):5105–5114, 2007.

Claudia M. Hendrix, Carolyn R. Mason, and Timothy J. Ebner. Signaling of grasp dimension and grasp force in dorsal premotor cortex and primary motor cortex neurons during reach to grasp in the monkey. J Neurophysiol, April 2009.

Leigh R. Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick van der Smagt, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. Nature, 485(7398):372–375, 2012.

Leigh R. Hochberg, Mijail D. Serruya, Gerhard M. Friehs, Jon A. Mukand, Maryam Saleh, Abraham H. Caplan, Almut Branner, David Chen, Richard D. Penn, and John P. Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. Nature, 442:164–171, 2006.

Thea Iberall. The nature of human prehension: Three dextrous hands in one. In Robotics and Automation. Proceedings. 1987 IEEE International Conference on, volume 4, pages 396–401, 1987.

Thea Iberall. Human prehension and dexterous robot hands. The International Journal of Robotics Research, 16(3):285–299, June 1997.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. Machine learning, 37(2):183–233, 1999.

Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II, 1997.

E. R. Kandel, J. H. Schwartz, and T. M. Jessel. Principles of neural science. McGraw-Hill, 3rd edition, 1991.

Robert E. Kass. Computing observed information by finite differences. Communication in Statistics: Simulation and Computation, 2:587–599, 1987.

Robert E. Kass, Luke Tierney, and Joseph B. Kadane. The validity of posterior expectations based on laplace's method. In S. Geisser, J. S. Hodges, S. J. Press, and A. Zellner, editors, Essays in Honor of George Bernard. Elsevier Science Publishers, North-Holland, 1990.

Kazutomo Kawamura. The structure of multivariate poisson distribution. Kodai Mathematical Journal, 2: 337–345, 1979.

P. R. Kennedy and R AE Bakay. Restoration of neural output from a paralyzed patient by a direct brain connection. Neuroreport, 9(8):1707–1711, 1998.

P. R. Kennedy, R.A.E. Bakay, M.M. Moore, K. Adams, and J. Goldwaithe. Direct control of a computer from the human central nervous system. Rehabilitation Engineering, IEEE Transactions on, 8(2):198 –202, jun 2000.

Ronald E. Ketter, Andrew B. Schwartz, and Apostolos P. Georgopoulos. Primate motor cortex and free arm movements to visual targets in three-dimensional space. iii. positional gradients and population coding of movement direction from various movement origins. Journal of Neuroscience, 8:2938–2947, 1988.

Ronald E. Kettner, Andrew B. Schwartz, and Apostolos P. Georgopoulos. Primate motor cortex and free arm movements to visual targets in three- dimensional space. iii. positional gradients and population coding of movement direction from various movement origins. The Journal of Neuroscience, 8(8):2938–2947, 1988.

S.-P. Kim, J. Simeral, J. P. Hochberg, L. Donoghue, and M. J. Black. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. J. Neural Engineering, 5: 455–476, 2008.

Sung-Phil Kim, John D. Simeral, Leigh R. Hochberg, John P. Donoghue, Gerhard M. Friehs, and Michael J. Black. Point-and-click cursor control with an intracortical neural interface system by humans with tetraplegia. IEEE Trans Neural Syst Rehabil Eng., 19(2):193–203, 2011.

Shinsuke Koyama, Lucia Castellanos, Cosma R. Shalizi, and Robert E. Kass. Approximate methods for state-space models. Journal of the American Statistical Association, 105(489):170–180, March 2010a.

Shinsuke Koyama, Lucia Castellanos, Cosma Rohilla Shalizi, and Robert E. Kass. Laplaces method in neural decoding. Computational and Sytems Neuroscience, 2008.

Shinsuke Koyama, Steven M. Chase, Andrew S. Whitford, Meel Velliste, Andrew B. Schwartz, and Robert E. Kass. Comparison of brain-computer interface decoding algorithms in open-loop and closed-loop control. Journal of Computational Neuroscience, 29(1-2):73–87, 2010b.

Vernon Lawhern, Wei Wu, Nicholas Hatsopoulos, and Liam Paninski. Population decoding of motor cortical activity using a generalized linear model with hidden states. Journal of Neuroscience Methods, 189(2): 267 – 280, 2010.

Eric C Leuthardt, Gerwin Schalk, Jonathan R Wolpaw, Jeffrey G Ojemann, and Daniel W Moran. A brain-computer interface using electrocorticographic signals in humans. Journal of Neural Engineering, 1(2): 63, 2004.

C. R. Mason, J. E. Gomez, and T. J. Ebner. Hand synergies during reach-to-grasp. The Journal of Neurophysiology, 86(6):2896–2910, 2001.

Carolyn R Mason, Claudia M Hendrix, and Timothy J Ebner. Purkinje cells signal hand shape and grasp force during reach-to-grasp in the monkey. J Neurophysiol, 95(1):144–58, 2006.

Carolyn R Mason, Michael TV Johnson, Qing-Gong Fu, Jose E Gomez, and Timothy J Ebner. Temporal profile of the directional tuning of the discharge of dorsal premotor cortical cells. Neuroreport, 9(6): 989–995, 1998.

C.R. Mason, L.S. Theverapperuma, C.M. Hendrix, and T.J. Ebner. Monkey hand postural synergies during reach-to-grasp in the absence of vision of the hand and object. J Neurophysiol, 91(6):2826–37, 2004.

E. M. Maynard, N. G. Hatsopoulos, C. L. Ojakangas, B. D. Acuna, J. N. Sanes, R. A. Normann, and J. P. Donoghue. Neuronal interactions improve cortical population coding of movement direction. J. Neurosci., 19(18):8083–8093, September 1999.

Sebastian Mika, Gunnar Rtsch, Jason Weston, Bernhard Schlkopf, and Klaus-Robert Mller. Fisher discriminant analysis with kernels, 1999.

Tom Mitchell. Machine Learning. McGraw Hill, 1997.

Daniel W. Moran and Andrew B. Schwartz. Motor Cortical Representation of Speed and Direction During Reaching. J Neurophysiol, 82(5):2676–2692, 1999.

Grant H Mulliken, Sam Musallam, and Richard A Andersen. Decoding trajectories from posterior parietal cortex ensembles. Journal of Neuroscience, 28(48):12913–12926, 2008.

S. Musallam, B. D. Corneil, B. Greger, H. Scherberger, and R. A. Andersen. Cognitive control signals for neural prosthetics. Science, 305(5681):258–262, 2004.

J. R. Napier. The Prehensile Movements of the Human Hand. J Bone Joint Surg Br, 38-B(4):902–913, 1956.

Markus Ojala and Gemma C. Garriga. Permutation tests for studying classifier performance. Journal of Machine Learning Research, pages 1833–1863, June 2010.

A.L. Orsborn, S. Dangi, H.G. Moorman, and J.M. Carmena. Closed-loop decoder adaptation on intermediate time-scales facilitates rapid bmi performance improvements independent of decoder initialization conditions. IEEE Trans Neural Syst Rehabil Eng, 20(4):468–77, 2012.

Liam Paninski. Maximum likelihood estimation of cascade point-process neural encoding models. Network: Computation in Neural Systems, 15(4):243–262, 2004.

Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua T. Vogelstein, and Wei Wu. A new look at state-space models for neural data. Journal of Computational Neuroscience, 29(1-2):107–126, 2010.

Liam Paninski, M. R. Fellows, N. G. Hatsopoulos, and J. P. Donoghue. Spatiotemporal tuning of motor cortical neurons for hand position and velocity. Journal of Neurophysiology, 91:515–532, 2004a.

Liam Paninski, Jonathan Pillow, and Jeremy Lewi. Statistical models for neural encoding, decoding, and optimal stimulus design. In Computational Neuroscience: Progress in Brain Research. Elsevier, 2006.

Liam Paninski, Shy Shoham, Matthew R Fellows, Nicholas G Hatsopoulos, and John P Donoghue. Superlinear population encoding of dynamic hand trajectory in primary motor cortex. The Journal of neuroscience, 24(39):8551–8561, 2004b.

Colin Pesyna, Krishna Pundi, and Martha Flanders. Coordination of hand shape. The Journal of Neuroscience, 31(10):3757–3765, 2011.

J. Ramsay, Giles Hooker, and Spencer Graves. Functional Data Analysis with R and MATLAB. Springer Publishing Company, Incorporated, 1st edition, 2009.

J. Ramsay and B. W. Silverman. Functional Data Analysis. Springer Series in Statistics. Springer, 2005.

C.E. Rasmussen and C.K.I. Williams. Gaussian Processes for Machine Learning. Adaptive Computation And Machine Learning. Mit Press, 2006.

Jacob Reimer and Nicholas G Hatsopoulos. The problem of parametric neural coding in the motor system. Progress in Motor Control, pages 243–259, 2009.

C. Häger Ross and M. H. Schieber. Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies. The Journal of neuroscience : the official journal of the Society for Neuroscience, 20(22):8542–8550, November 2000.

Maryam Saleh, Kazutaka Takahashi, Yali Amit, and Nicholas G. Hatsopoulos. Encoding of coordinated grasp trajectories in primary motor cortex. The Journal of Neuroscience, 30(50):17079–17090, 2010.

Maryam Saleh, Kazutaka Takahashi, and Nicholas G Hatsopoulos. Encoding of coordinated reach and grasp trajectories in primary motor cortex. The Journal of Neuroscience, 32(4):1220–1232, 2012.

Emilio Salinas and L. F. Abbott. Vector reconstruction from firing rates. Journal of Computational Neuroscience, 1:89–107, 1994.

M. Santello, M. Flanders, and J.F. Soechting. Postural hand synergies for tool use. J Neurosci, 18(23): 10105–15, 1998.

Gopal Santhanam, Stephen I. Ryu, Byron M. Yu, Afsheen Afshar, and Krishna V. Shenoy. A high-performance brain-computer interface. Nature, 442(7099):195–198, July 2006.

Gopal Santhanam, Byron M. Yu, Vikash Gilja, Stephen I. Ryu, Afsheen Afshar, Maneesh Sahani, and Krishna V. Shenoy. Factor-analysis methods for higher-performance neural prostheses. Journal of Neurophysiology, 102(2):1315–1330, 2009.

B. Schölkopf, A. Smola, and K. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation, 10:1299–1319, 1998.

Andrew B. Schwartz. Motor cortical activity during drawing movements: single-unit activity during sinusoid tracing. Journal of neurophysiology, 68(2):528–541, 1992.

Andrew B. Schwartz. Motor cortical activity during drawing movements: population representation during sinusoid tracing. Journal of Neurophysiology, 70(1):28–36, 1993.

Andrew B. Schwartz, Ronald E. Kettner, and Apostolos P. Georgopoulos. Primate motor cortex and free arm movements to visual targets in three-dimensional space. i. relations between single cell discharge and direction of movement. The Journal of Neuroscience, 8(8):2913–2927, 1988.

Lauren E Sergio, Catherine Hamel-Pâquet, and John F Kalaska. Motor cortex neural correlates of output kinematics and kinetics during isometric-force and arm-reaching tasks. Journal of neurophysiology, 94 (4):2353–2378, 2005.

Lauren E Sergio and John F Kalaska. Changes in the temporal pattern of primary motor cortex activity in a directional isometric force versus limb movement task. Journal of Neurophysiology, 80(3):1577–1583, 1998.

Mijail Serruya, Nicholas G. Hatsopoulos, Liam Paninski, Matthew R. Fellows, and John P. Donoghue. Brain-machine interface: instant neural control of a movement signal. Nature, 416:141–142, 2002.

Krishna V Shenoy, Daniella Meeker, Shiyan Cao, Sohaib A Kureshi, Bijan Pesaran, Christopher A Buneo, Aaron P Batista, Partha P Mitra, Joel W Burdick, and Richard A Andersen. Neural prosthetic control signals from plan activity. Neuroreport, 14(4):591–596, 2003.

K.V. Shenoy, M.T. Kaufman, M. Sahani, and M.M. Churchland. A dynamical systems view of motor preparation implications for neural prosthetic system design. Prog Brain Res, 192, 2011.

H.-C. Shin, M. Schieber, and N. Thakor. Neural decoding of single and multi-finger movements based on ml. In Ratko Magjarevic, Chwee Teck Lim, and James C. H. Goh, editors, 13th International Conference on Biomedical Engineering, volume 23 of IFMBE Proceedings, pages 448–451. Springer Berlin Heidelberg, 2009a. ISBN 978-3-540-92841-6.

Hyun-Chool Shin, V. Aggarwal, S. Acharya, M.H. Schieber, and N.V. Thakor. Neural decoding of finger movements using skellam-based maximum-likelihood decoding. Biomedical Engineering, IEEE Transactions on, 57(3):754 –760, march 2010.

Hyun-Chool Shin, Marc H. Schieber, and Nitish V. Thakor. Neural subset decoding of finger movements, 2009b.

J. D. Simeral, S.-P. Kim, M. J. Black, J. P. Donoghue, and L. R. Hochberg. Neural control of cursor trajectory and click by a human with tetraplegia 1000 days after implant of an intracortical microelectrode array. J. of Neural Engineering, 8(2):025027, 2011.

Anne C. Smith and Emery N. Brown. Estimating a state-space model from point process observations. Neural Computation, 15:965–991, 2003.

J. F. Soechting and M. Flanders. Flexibility and repeatability of finger movements during typing: analysis of multiple degrees of freedom. Journal of computational neuroscience, 4(1):29–46, January 1997.

Lakshminarayan Srinivasan, Uri T Eden, Sanjoy K Mitter, and Emery N Brown. General-purpose filter design for neural prosthetic devices. Journal of neurophysiology, 98(4):2456–2475, 2007.

Eran Stark and Moshe Abeles. Predicting movement from multiunit activity. The Journal of neuroscience, 27(31):8387–8394, 2007.

Dawn M. Taylor, Stephen I. Helms Tillery, and Andrew B. Schwartz. Direct cortical control of 3D neuroprosthetic devices. Science, 296:1829–1832, 2002.

Pramodsingh H. Thakur, Amy J. Bastian, and Steven S. Hsiao. Multidigit Movement Synergies of the Human Hand in an Unconstrained Haptic Exploration Task. J. Neurosci., 28(6):1271–1281, 2008.

Luke Tierney, Robert E. Kass, and Joseph B. Kadane. Fully exponential Laplace approximations to expectations and variances of nonpositive functions. Journal of the American Statistical Association, 84: 710–716, 1989.

E. Todorov and Z. Ghahramani. Analysis of the synergies underlying complex hand manipulation. In 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 4637–4640, 2004.

Benjamin R Townsend, Erk Subasi, and Hansjörg Scherberger. Grasp movement decoding from premotor and parietal cortex. The Journal of Neuroscience, 31(40):14386–14398, 2011.

Wilson Truccolo, Uri T. Eden, Matthew R. Fellows, John P. Donoghue, and Emery N. Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. Journal of Neurophysiology, 93(2):1074–1089, 2004.

Wilson Truccolo, G.M. Friehs, J.P. Donoghue, and L.R. Hochberg. Primary motor cortex tuning to intended movement kinematics in humans with tetraplegia. J Neurosci, 28(5):1163–78, 2008.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research, 6:1453–1484, 2005.

Henry C. Tuckwell. Stochastic Processes in the Neurosciences. SIAM, Philadelphia, 1989.

Carlos E. Vargas-Irwin, Gregory Shakhnarovich, Payman Yadollahpour, John M. K. Mislow, Michael J. Black, and John P. Donoghue. Decoding complete reach and grasp actions from local primary motor cortex populations. The Journal of Neuroscience, 30(29):9659–9669, 2010.

M. Veber and T. Bajd. Assessment of human hand kinematics. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 2966–2971, 2006.

M. Veber, T. Bajd, and M. Munih. Assessing joint angles in human hand via optical tracking device and calibrating instrumented glove. Meccanica, 42:451–463, 2007.

Meel Velliste, Sagi Perel, Chance M. Spalding, Andrew S. Whitford, and Andrew B. Schwartz. Cortical control of a prosthetic arm for self-feeding. Nature, pages 1098–1101, May 2008.

Ramana Vinjamuri, Zhi Hong Mao, Robert Sclabassi, and Mingui Sun. Time-varying synergies in velocity profiles of finger joints of the hand during reach and grasp. In Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pages 4846–4849. IEEE, 2007.

Ramana Vinjamuri, Mingui Sun, Cheng-Chun Chang, Heung-No Lee, R.J. Sclabassi, and Zhi-Hong Mao. Temporal postural synergies of the hand in rapid grasping tasks. Information Technology in Biomedicine, IEEE Transactions on, 14(4):986 –994, july 2010a.

Ramana Vinjamuri, Mingui Sun, Cheng-Chun Chang, Heung-No Lee, Robert J. Sclabassi, and Zhi-Hong Mao. Dimensionality reduction in control and coordination of the human hand. IEEE Transactions on Biomedical Engineering, 57:284–295, 2010b.

Ramana Vinjamuri, Douglas J Weber, Zhi-Hong Mao, Jennifer L Collinger, Alan D Degenhart, John W Kelly, Michael L Boninger, Elizabeth C Tyler-Kabara, and Wei Wang. Toward synergy-based brain-machine interfaces. Information Technology in Biomedicine, IEEE Transactions on, 15(5):726–736, 2011.

D. A. Heldman Wang, S. S. Chan and D. W. Moran. Motor cortical representation of position and velocity during reaching. Journal of Neurophysiology, 97:4258–4270, 2007.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. IEEE Trans. Pattern Anal. Mach. Intell., 30(2):283–298, 2008.

Wei Wang, Sherwin S. Chan, Dustin A. Heldman, and Daniel W. Moran. Motor cortical representation of hand translation and rotation during reaching. The Journal of Neuroscience, 30(3):958–962, 2010.

Wei Wang, Jennifer L Collinger, Alan D Degenhart, Elizabeth C Tyler-Kabara, Andrew B Schwartz, Daniel W Moran, Douglas J Weber, Brian Wodlinger, Ramana K Vinjamuri, Robin C Ashmore, et al. An electrocorticographic brain interface in an individual with tetraplegia. PloS one, 8(2), 2013.

David K Warland, Pamela Reinagel, and Markus Meister. Decoding visual information from a population of retinal ganglion cells. Journal of Neurophysiology, 78(5):2336–2350, 1997.

Johan Wessberg, Christopher R. Stambaugh, Jerald D. Kralik, Pamela D. Beck, Mark Laubach, John K. Chapin, Jung Kim, S. James Biggs, Mandayam A. Srinivasan, and Miguel A. L. Nicolelis. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature, 408(6810):361–365, Nov. 2000.

John Wojdylo. On the coefficients that arise from laplace's method. Journal of Computational and Applied Mathematics, 196, 2006.

Jonathan R. Wolpaw and Dennis J. McFarland. Control of a two-dimensional movement signal by a non-invasive brain-computer interface in humans. Proceedings of the National Academy of Sciences of the United States of America, 101(51):17849–17854, 2004.

Wei Wu, M. J. Black, Y. Gao, E. Bienenstock, M. Serruya, and J. P. Donoghue. Inferring hand motion from multi-cell recordings in motor cortex using a kalman filter. In SAB02Workshop on Motor Control in Humans and Robots: On the Interplay of Real Brains and Artificial Devices, pages 66–73, 2002.

Wei Wu, M. J. Black, Y. Gao, E. Bienenstock, M. Serruya, A. Shaikhouni, and J. P. Donoghue. Neural decoding of cursor motion using a kalman filter. In in Advances in Neural Information Processing Systems 15, pages 133–140. MIT Press, 2003.

Wei Wu, Michael J. Black, David Mumford, Yun Gao, Elie Bienenstock, and John P. Donoghue. Modeling and decoding motor cortical activity using a switching kalman filter. In IEEE Trans Biomed Eng., pages 933–942, 2004.

Wei Wu, Yun Gao, Elie Bienenstock, John P. Donoghue, and Michael J. Black. Bayesian population decoding of motor cortical activity using a kalman filter. Neural Computation, 18:80–118, January 2006. ISSN 0899-7667.

Wei Wu and Nicholas G Hatsopoulos. Real-time decoding of nonstationary neural activity in motor cortex. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 16(3):213–222, 2008.

Wei Wu, J.E. Kulkarni, N.G. Hatsopoulos, and L. Paninski. Neural decoding of hand motion using a linear state-space model with hidden states. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 17(4):370 –378, aug. 2009.

Kai Xu, Yiwen Wang, Yueming Wang, Fang Wang, Yaoyao Hao, Shaomin Zhang, Qiaosheng Zhang, Weidong Chen, and Xiaoxiang Zheng. Local-learning-based neuron selection for grasping gesture prediction in motor brain machine interfaces. Journal of neural engineering, 10(2), 2013.

Byron M. Yu, John P. Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. Journal of Neurophysiology, 102(1):614–635, 2009.

Byron M. Yu, Caleb Kemere, Gopal Santhanam, Afsheen Afshar, Stephen I. Ryu, Teresa H. Meng, Maneesh Sahani, and Krishna V. Shenoy. Mixture of Trajectory Models for Neural Decoding of Goal-Directed Movements. J Neurophysiol, 97(5):3763–3780, may 2007.

Byron M. Yu, Gopal Santhanam, Maneesh Sahani, and Krishna V. Shenoy. Statistical Signal Processing for Neuroscience and Neurotechnology. Chapter 7: Neural Decoding for motor and communication prostheses. Academic Press, 2010.

Jun Zhuang, Wilson Truccolo, Carlos Vargas-Irwin, and John P Donoghue. Decoding 3-d reach and grasp kinematics from high-frequency local field potentials in primate primary motor cortex. Biomedical Engineering, IEEE Transactions on, 57(7):1774–1784, 2010.

K. Zilles, G. Schlaug, M. Matelli, G. Luppino, A. Schleicher, M. Qü, A. Dabringhaus, R. Seitz, and P. E. Roland. Mapping of human and macaque sensorimotor areas by integrating architectonic, transmitter receptor, mri and pet data. Journal of Anatomy, 187(3):515–537, December 1995.