To my family
ACKNOWLEDGMENTS

My deepest gratitude goes to my advisor Dr. Jose C. Principe because of his guidance, support, critiques and sense of humor. I feel blessed to be one of his students. Inspired by his suggestions, encouraged by his rigorous comments, relaxed by his jokes, touched by his caring, I was brought along into the fabulous world of research. It is Dr. Principe who taught me to how to think as a researcher.

My gratitude also goes to Dr. Justin C. Sanchez, Dr. John G. Harris, Dr. Dapeng Wu, (who are members of my committee), and Antonio Paiva, for their suggestions and help in my research.

I am also grateful to Aysegul Gunduz, John DiGiovanna, Shalom Darmanjian, Ruijiang Li, Weifeng Liu, Dr. Rui Yan, Dr. Jianwu Xu, and Dr. Dongming Xu for their incredible supports and caring. I also would like to express gratitude to Julie Veal, who helps me with my English writing.

Last but not least, I am indebted to my mother, my father and my fiancé, for their endless love, support and strong belief in me. This dissertation is dedicated to them.
# TABLE OF CONTENTS

**ACKNOWLEDGMENTS** ...............................................................................................................4

**LIST OF TABLES** ....................................................................................................................7

**ABSTRACT** .............................................................................................................................10

**CHAPTER**

1 **INTRODUCTION** ..................................................................................................................12
   Description of Brain Machine Interfaces................................................................................12
   Review of the Approaches in Spike Domain................................................................................13
   Spike Sorting—Preprocessing Neural Activities ............................................................ 14
   Spike-Based Association Analysis ..................................................................................15
   Spike-Based Modeling ....................................................................................................17
   Encoding analysis.....................................................................................................18
   Decoding algorithms .................................................................................................21
   Outline ............................................................................................................................25

2 **PROBABILISTIC APPROACH FOR POINT PROCESS** ....................................................30
   Sequential State Estimation Problem – Pros and Cons ..........................................................30
   Review of the Previous Probabilistic Approaches..................................................................31
   Adaptive Algorithms for Point Processes...............................................................................32
   Adaptive Filtering for Point Processes with Gaussian Assumption................................33
   Monte Carlo Sequential Estimation for Point Processes.................................................35
   Simulation of Monte Carlo Sequential Estimation on Neural Spike Train Decoding............40
   Interpretation...........................................................................................................................44

3 **INFORMATION THEORETICAL ANALYSIS OF INSTANTANEOUS MOTOR**
   CORTICAL NEURON ENCODING.....................................................................................48
   Experimental Setups .................................................................................................................48
   Data Recording......................................................................................................................48
   Simulation vs. Vivo Recordings............................................................................................50
   Review of Tuning Analysis ....................................................................................................51
   Visual Inspection of a Tuning Neuron .....................................................................................55
   Metric for Tuning ...................................................................................................................55
   Tuning Depth......................................................................................................................56
   Information Theoretic Tuning Metric ..................................................................................57
   Simulated Neural Recordings ..............................................................................................59
   In Vivo Neural Recordings.................................................................................................63
   Information Theoretical Neural Encoding..........................................................................64
   Instantaneous Tuning Function in Motor Cortex ..............................................................64
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Comparison results of all algorithms with different $Q_k$</td>
<td>45</td>
</tr>
<tr>
<td>3-1</td>
<td>Assignment of the sorted neural activity to the electrodes</td>
<td>77</td>
</tr>
<tr>
<td>3-2</td>
<td>The statistical similarity results comparison</td>
<td>79</td>
</tr>
<tr>
<td>3-3</td>
<td>The comparison of percentage of Monte Carlo results in monotonically increasing</td>
<td>79</td>
</tr>
<tr>
<td>4-1</td>
<td>The kinematics reconstructions by Monte Carlo SE for segment of test data</td>
<td>121</td>
</tr>
<tr>
<td>4-2</td>
<td>Averaged performance by Monte Carlo SE of the kinematics reconstructions for segment of test data</td>
<td>123</td>
</tr>
<tr>
<td>4-3</td>
<td>Statistical performance of the kinematics reconstructions using 2 criteria</td>
<td>123</td>
</tr>
<tr>
<td>4-4</td>
<td>Results comparing the kinematics reconstructions averaged among Monte Carlo trials and synthetic averaging</td>
<td>126</td>
</tr>
<tr>
<td>4-5</td>
<td>Statistical performance of the kinematics reconstructions by Monte Carlo SE and synthetic averaging</td>
<td>127</td>
</tr>
<tr>
<td>4-6</td>
<td>Results comparing the kinematics reconstruction by Kalman PP and Monte Carlo SE for a segment of data</td>
<td>127</td>
</tr>
<tr>
<td>4-7</td>
<td>Statistical performance of the kinematics reconstructions by Kalman PP and Monte Carlo SE (synthetic averaging)</td>
<td>130</td>
</tr>
<tr>
<td>4-8</td>
<td>Statistical performance of the kinematics reconstructions by different encoding models</td>
<td>130</td>
</tr>
<tr>
<td>4-9</td>
<td>Statistical performance of the kinematics reconstructions Kalman filter and Kalman PP</td>
<td>133</td>
</tr>
<tr>
<td>4-10</td>
<td>Statistical performance of the kinematics reconstructions by spike pates and by point process</td>
<td>133</td>
</tr>
<tr>
<td>4-11</td>
<td>Statistical performance of the kinematics reconstructions by neuron subset and full ensemble</td>
<td>135</td>
</tr>
<tr>
<td>5-1</td>
<td>Results of the kinematics reconstructions by Kalman and dual Kalman for segment of test data</td>
<td>159</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Brain machine interface paradigm</td>
<td>29</td>
</tr>
<tr>
<td>2-1</td>
<td>The desired velocity generated by triangle wave with Gaussian noise</td>
<td>45</td>
</tr>
<tr>
<td>2-2</td>
<td>The simulated neuron spike train generated by an exponential tuning function</td>
<td>45</td>
</tr>
<tr>
<td>2-3</td>
<td>The velocity reconstruction by different algorithms</td>
<td>46</td>
</tr>
<tr>
<td>2-4</td>
<td>$p(v_k</td>
<td>\Delta N_k)$ at different time</td>
</tr>
<tr>
<td>3-1</td>
<td>The BMI experiments of 2D target reaching task. The monkey moves a cursor (yellow circle) to a randomly placed target (green circle), and is rewarded if a cursor intersects the target</td>
<td>77</td>
</tr>
<tr>
<td>3-2</td>
<td>Tuning plot for neuron 72</td>
<td>77</td>
</tr>
<tr>
<td>3-3</td>
<td>A counterexample of neuron tuning evaluated by tuning depth. The left plot is a tuning plot of neuron 72 with tuning depth 1. The right plot is for neuron 80 with tuning depth 0.93</td>
<td>78</td>
</tr>
<tr>
<td>3-4</td>
<td>The conditional probability density estimation</td>
<td>78</td>
</tr>
<tr>
<td>3-5</td>
<td>The average tuning information across trials by different evaluation</td>
<td>79</td>
</tr>
<tr>
<td>3-6</td>
<td>Traditional tuning depth for all the neurons computed from three kinematics</td>
<td>80</td>
</tr>
<tr>
<td>3-7</td>
<td>Information theoretic tuning depth for all the neurons computed from 3 kinematics plotted individually</td>
<td>81</td>
</tr>
<tr>
<td>3-8</td>
<td>Block diagram of Linear-Nonlinear-Poisson model</td>
<td>82</td>
</tr>
<tr>
<td>3-9</td>
<td>Sketch map of the time delay between neuron spike train (bottom plot) and the kinematics response (upper plot)</td>
<td>82</td>
</tr>
<tr>
<td>3-10</td>
<td>The conditional probability density estimation</td>
<td>83</td>
</tr>
<tr>
<td>3-11</td>
<td>Mutual information as function of time delay for 5 neurons</td>
<td>83</td>
</tr>
<tr>
<td>3-12</td>
<td>Nonlinearity estimation for neurons</td>
<td>84</td>
</tr>
<tr>
<td>3-13</td>
<td>Correlation coefficient between the nonlinearity calculated from windowed kinematics and the instantaneous kinematics with optimum delay</td>
<td>86</td>
</tr>
<tr>
<td>3-14</td>
<td>Comparison of encoding results by instantaneous modeling and windowed modeling</td>
<td>87</td>
</tr>
</tbody>
</table>
3-15 Comparison of encoding similarity by instantaneous modeling and windowed modeling across kernel size .................................................................88

4-1 Schematic of relationship between encoding and decoding processes for Monte Carlo sequential estimation of point processes .................................................................121

4-2 The posterior density of the reconstructed kinematics by Monte Carlo SE ................. 122

4-3 The reconstructed kinematics for 2-D reaching task .................................................. 123

4-4 Linear model error using different $\alpha$ ...................................................................... 124

4-5 cdf of noise distribution using different density ............................................................ 125

4-6 Nonlinearity of neuron 72 using different $\sigma$ ............................................................ 125

4-7 Decoding performances by different $x_n$ .................................................................... 126

4-8 The reconstructed kinematics for a 2-D reaching task ................................................. 128

4-9 The decoding performance by algorithms in PP for different data sets ...................... 131

4-10 Threshold setting for sorted information theoretic tuning depths for 185 neurons .... 133

4-11 Selected neuron subset (30 neurons) distribution ...................................................... 134

4-12 Statistical performance of reconstructed kinematics by different neuron subsets ...... 136

5-1 The reconstructed kinematics for 2-D reaching task by Kalman and dual Kalman filter .................................................................................................................. 159

5-2 The tracking of the tuning parameters for the 10 most important neurons in dual Kalman filter ................................................................................................. 160
Brain Machine Interface (BMI) design uses linear and nonlinear models to discover the functional relationship between neural activity and a primate’s behavior. The loss of time resolution contained in spike timing cannot be captured in traditional adaptive filtering algorithms and might exclude useful information for the generation of movement. More recently, a Bayesian approach based on the observed spike times modeled as a discrete point process has been proposed. However, it includes the simplifying assumption of Gaussian distributed state posterior density, which in general may be too restrictive. We proposed in this dissertation a Monte Carlo sequential estimation framework as a probabilistic approach to reconstruct the kinematics directly from the multi-channel neural spike trains. Sample states are generated at each time step to recursively evaluate the posterior density more accurately. The state estimation is obtained easily by reconstructing the posterior density with Parzen kernels to obtain its mean (called collapse). This algorithm is systematically tested in a simulated neural spike train decoding experiment and then in BMI data. Implementing this algorithm in BMI requires knowledge of both neuronal representation (encoding) and movement decoding from spike train activity. Due to the on-line nature of BMIs, an instantaneous encoding estimation is necessary which is different from the current models using time windows. We investigated an information
theoretic technique to evaluate neuron’s tuning functional relationship between the instantaneous kinematic vector and neural firing in the motor cortex by a parametric linear-nonlinear-Poisson model. Moreover, mutual information is utilized as a tuning criterion to provide a way to estimate the optimum time delay between motor cortical activity and the observed kinematics. More than half (58.38%) of the neurons instantaneous tuning curves display a 0.9 correlation coefficient with those estimated with the temporal kinematic vector.

With the knowledge gained from tuning analysis encapsulated in an observation model, our proposed Brain Machine Interface becomes a problem of state sequential estimation. The kinematics is directly reconstructed from the state of the neural spike trains through the observation model. The posterior density estimated by Monte Carlo sampling modifies the amplitude of the observed discrete neural spiking events by the probabilistic measurement. To deal with the intrinsic spike randomness in online modeling, synthetic spike trains are generated from the intensity function estimated from the neurons and utilized as extra model inputs in an attempt to decrease the variance in the kinematic predictions. The performance of the Monte Carlo Sequential Estimation methodology augmented with this synthetic spike input provides improved reconstruction further. The current methodology assumes a stationary tuning function of neurons, which might not be true. The effect of the tuning function non-stationary was also studied by testing the decoding performance in different segment of data. The preliminary results on tracking the non-stationary tuning function by a dual Kalman structure indicate a promising avenue for future work.
CHAPTER 1
INTRODUCTION

Description of Brain Machine Interfaces

Brain-Machine Interfaces (BMIs) exploit the spatial and temporal structure of neural activity to directly control a prosthetic device. The early work in the 1980s by Schmidt [1980], and Georgopoulos, Schwartz and colleagues [1986], first described the concepts, application and design of BMI as an engineering interface to modulate the motor system by neural firing patterns. Two decades later, several research groups have designed experimental paradigms to implement the ideas for Brain Machine Interfaces [Wessberg et al., 2000; Serruya et al., 2002]. These are illustrated in Figure 1-1.

In this framework [Wessberg et al., 2000; Serruya et al., 2002], neuronal activity (local field potentials and single unit activity) has been synchronously collected from microelectrode arrays implanted into multiple cortical areas while animals and humans have performed 3-D or 2-D target-tracking tasks. Several signal-processing approaches have been applied to extract the functional relationship between the neural recordings and the animal’s kinematic trajectories [Wessberg et al. 2000; Sanchez, et al., 2002b; Kim, et al., 2003; Wu, et al., 2006; Brockwell, et al., 2004]. The models predict movements and control a prosthetic robot arm or computer to implement them. Many decoding methodologies use binned spike trains to predict movement based on linear or nonlinear optimal filters [Wessberg et al. 2000; Sanchez et al. 2002b; Kim et al., 2003]. These methods avoid the need for explicit knowledge of the neurological dynamic encoding properties, and standard linear or nonlinear regression is used to fit the relationship directly into the decoding operation. Yet another methodology can be derived probabilistically using a state model within a Bayesian formulation [Schwartz, et al. 2001; Wu et al. 2006; Brockwell et al. 2004]. From a sequence of noisy observations of the neural activity, the
probabilistic approach analyzes and infers the kinematics as a state variable of the neural
dynamical system. The neural tuning property relates the measurement of the neural activity to
the animal’s behaviors, and builds up the observation measurement model. Consequently, a
recursive algorithm based on all available statistical information can be used to construct the
posterior probability density function of each kinematic state given the neuron activity at each
time step from the prior density of the state. The prior density in turn becomes the posterior
density of previous time step updated with the discrepancy between an observation model and
the neuron firings. Movements can be recovered probabilistically from the multi-channel neural
recordings by estimating the expectation of the posterior density or by maximum a posterior.

**Review of the Approaches in Spike Domain**

The mathematical model in Brain Machine Interfaces requires the application of signal
processing techniques to functionally approximate the relationship between neural activity and
kinematics, such as spike sorting and association analysis between neurons and neuron
encoding/decoding algorithms. Adaptive signal processing is a well-established engineering
domain to analyze the temporal evolution of system characteristics [Haykin, 2002]. Traditional
adaptive processing requires continuous measurement of signals using tools such as the Wiener
filter, least square algorithm, and Kalman filter. Early BMI research frequently employed a
binning process to analyze and develop algorithms to obtain the neural firing rate as a continuous
signal. This binning process conceals the randomness of neural firing behaviors, and the binning
window size is always a concern. In Brain Machine Interfaces, neural activity and plasticity are
characterized by spike trains. The loss of time resolution for “true” neuron activities might
exclude information useful for movement generation. Thus an analysis of the spike domain is
necessary for this specific application of BMI.
Spike Sorting—Preprocessing Neural Activities

In BMI neurophysiologic recordings, extracellular neural action potentials are recorded with multiple electrodes representing the simultaneous electrical activity of a neuronal population. To identify the action potentials of each neuron, the multi-channel data processing of the spike train data analysis or decoding in BMIs starts with a spike sorting step. Most commonly, an action potential is detected by imposing a threshold on the amplitude of the amplified signal, thereby generating a pulse every time an action potential occurs. However, this method is subject to failure due to noise contamination and spike overlapping, and the results may not contain a single threshold for all conditions of interest. Previous research introduced many algorithms to analyze spike shape features and to perform spike sorting by classifying multiple spike shapes at the same time [Lewicki, 1998]. Clustering provides a simple way to organize spikes by their shape, but also has an unfortunate trade-off between false positives and missed spikes. Clustering in Principal Component Analysis space avoids the noise problem and separates the different spike shapes according to the primary, or more robust, result. Template-based Bayesian clustering quantifies the certainty of the spike classification by computing the likelihood of data given a particular class. Fee et al. [1996] developed an approach to choose the number of classes for Bayesian clustering by guiding the histogram of the interspike intervals. An optimal filter-based method based on the assumption of accurate estimation of the spike shapes and noise spectrum [Gozani & Miller, 1994] was also proposed to discriminate the spikes from each other and the background noise. These methods remain unable to cope with overlapping spikes. Neural networks, however, showed improved performance by providing more general decision boundaries [Chandra & Optican, 1997]. Multi-recording of neuron activity resulted in the ability to discriminate overlapping spikes. Independent Component Analysis (ICA) was successfully used for multi-channel spike sorting [Makeig et al., 1997; McKeown et
ICA has a strong assumption that each channel should be regarded as one signal source and that all sources are mixed linearly. Although a significant body of work has addressed spike detection/sorting algorithms, the problem is far from solved. The major shortcomings are (1) assumption of stationary spike shapes across the experiment, which disregards electrode drift; (2) assumption of stationary background noise; (3) the necessity of proper spike alignment techniques for overlapping action potentials. The accuracy of the spike detection/sorting techniques directly affects the prediction results of BMIs, but to what level this occurs is unknown. Sanchez [2005] showed that the results of linear models using unsorted spike data differs little from the sorted spikes in simple movement prediction, but it may affect more complex movement prediction.

**Spike-Based Association Analysis**

The most common methods for spike train analysis are based on histograms, which require the assumption of stationary parameters. The association among multi-neural spike trains can be analyzed with and without neural stimulation. The functional relationship between neural spikes and local field potentials can also be analyzed based on pre-stimulus patterns.

Brody [1999] proposed the unnormalized cross-correlogram (cross-covariance) to measure the pair-wise association between two binned spike rates over different time lags, but this method lacks time resolution. Cross-intensity function [Brillinger, 1992], a similar concept, measures the spike rate of one neuron when another neuron fires a spike, and it preserves the temporal resolution.

To quantify the association among more than two neurons in an ensemble (i.e., the presence of spatiotemporal patterns); two statistical approaches to parameterize these interactions have been introduced (1) coefficients of log-linear models, (2) a Bayesian approach for inferring the existence or absence of interactions, and an estimation of the strength of those interactions.
[Martignon et al., 2000]. A data-mining algorithm, originally developed to analyze the
generation of interictal activity in EEG recordings [Bourien et al., 2005] was also applied to
automatically extract co-activated neurons. This method provided the statistical evidence for the
existence of neuron subsets based on the stationary characteristics of neural activities. The
automatic extraction of neuron subsets needs long data segments in order to be useful. An online
realization has yet to be developed.

Another technique for the association analysis between neurons, appropriate when a
stimulus is present, is the Joint-Peri-Stimulus-Time-Histogram (JPSTH) [Gerstein & Perkel,
1969], which is the extension concept of PSTH for a single neuron [Abeles, 1982]. JPSTH is the
joint histogram between two spike trains, and describes the joint pdf of the synchrony when a
stimulus occurs. The computation is based on the null hypothesis that the spike trains are the
realization of independent Poisson-point processes, and as such are independent. The neuron
response to the stimulus is assumed statistically stationary.

The association analysis between spike firings and local field potentials (LFP) also has
been investigated in terms of stimulus. Researchers have described the temporal structure in
LFPs and spikes where negative deflections in LFPs were proposed to reflect excitatory, spike-
causing inputs to neurons near the electrode [Arieli et al., 1995]. The most appropriate feature
detection method explores the correlation between the amplitude modulated (AM) components
of the movement-evoked local field potentials and single-unit activities recorded as stimulus at
the same electrode across all movement trials [Wang et al., 2006a]. The correlation between
pairs of peri-event time histograms (PETH) and movement evoked local field potentials (mEP) at
the same electrode showed high correlation coefficients for some neurons, suggesting that the
extracellular dendritic potentials indicate the level of neuronal output. A critical demonstration of
this relationship was the process of averaging the LFP and single unit activity across the lever press trials, thus reducing the noise contamination caused by the random realization of unmodeled brain spontaneous activities. More work is needed toward reducing noise contamination.

All the above histogram-based methods can be considered empirically as approximations to the probabilistic density, and information theoretic measures can be introduced into each method. The information theoretic calculation for the spike train uses milliseconds, which, as the minimum time scale determined to contain information [Borst & Theunissen, 1999], is the “limiting spiking timing precision.” Entropy was proposed to qualify the information carried by the spike arrival times [Strong et al., 1998]. Mutual information can be used to measure the pairwise neural train association, the statistical significance conveyed by the neuron responding to the stimulus [Nirenberg et al., 2001], and the evaluation of the independence and redundancy from the nearby cortical neuron recordings [Reich et al., 2001]. The information theoretic calculation can be performed directly on the neural activity, but the operation needs enough data to ensure that the histogram-based analysis performs well. The mutual information summarizes the relationship between multi spike trains and the neural response to a biological stimulus, but in only a scalar quantity, which does not describe the complicated relationship as well as modeling does.

**Spike-Based Modeling**

In addition to determining stimulus response association through a statistical analysis of the neural spike train, researchers also investigated parametric probability modeling using the likelihood method to estimate point process properties. A good model is an optimal way to theoretically predict and analyze the underlying dynamics of neural spike generation. A simple inhomogeneous Poisson process has been used most frequently to model both the simulation and
quantification of the neural activity analysis of a single spike train [Tuckwell 1988; Rieke *et al.*, 1997; Gabbiani & Koch, 1998; & Reich *et al.*, 1998]. This model is particularly appealing because it can explicitly describe neuron spiking as a simple analytical Poisson process [Brown *et al.*, 1998; Zhang *et al.*, 1998]. The inhomogeneous Poisson model cannot, however, completely describe the neuron behavior with a multimodal inner spike interval distribution [Gabbiani & Koch, 1998; Shadlen & Newsome, 1998]. Non-Poisson spike train probabilistic models have been studied under the assumption that a neuron fires probabilistically, but the model depends on the experimental time and the elapsed time since the previous spike [Kass & Ventura, 2001]. Additionally, dependencies between multi-spike trains were analyzed through the pair-wise interactions among the ensemble of neurons, where the firing rate in the inhomogeneous Poisson was modeled as a function of the inhibitory and excitatory interaction history of nearby neurons [Okatan *et al.*, 2005]. Truccolo *et al.* [2005] proposed a similar analysis as a statistical framework, based on the point process likelihood function, to relate the neuron spike probability to the spiking history, concurrent ensemble activity, and extrinsic covariates such as stimuli or behavior. All of these parametric modeling methods provided a coherent framework to understand neural behavior and the base to statistically apply mathematical models to study the relationship between spike patterns of ensembles of neurons and an external stimulus or biological response (the encoding), which characterizes the neural spike activity as a function of the stimulus, and decoding, which infers the biological response from the neural spikes.

**Encoding analysis**

The ‘neural code’ refers how a neuron represents behavioral responses or how it responds to a stimulus. The parameterization of a tuning function requires an understanding of three interconnected aspects 1) What is the behavior/stimulus? 2) How does the neuron encode it? 3)
What is the criterion for quantifying the quality of the response? The tuning curve was measured initially as a cosine curve between the stimulus and the response [Georgopoulos et al., 1989] using mainly static stimuli to discriminate between the stimuli based on neural responses.

For neurons located in the motor cortex, researchers first developed the static descriptions of movement-related activity by applying electrical stimuli to motor areas to elicit muscle contraction [Fritsch & Hitzig, 1870; Leyton & Sherrington, 1917; Schafer, 1900]. Later, movement direction was correlated with cortical firing in a center-out task where the tuning function was initially modelled as a cosine curve [Georgopoulos et al. 1982]. The peak discharge rate of a cell is called preferred direction. To quantify the degree of tuning, the tuning depth has been proposed as a metric and it is defined as the difference between the maximum and minimum values in the firing rates, normalized by the standard deviation of the firing rate [Carmena et al., 2003, Sanchez et al., 2003]. As a scalar, the tuning depth summarizes the statistical information contained in the tuning curve to evaluate the neural representation, which indicates how modulated the cell’s firing rate is to the kinematic parameter of interest. However, this metric has some shortcomings since it can exaggerate the value of tuning depth when the neuron firing rate standard deviation is close to 0. Additionally, it depends on the binning window size to calculate the firing rate of the neuron. The tuning depth also relates to the scale of the behavior/stimulus and makes the analysis not comparable among neurons as we will see. A more principled metric, allowing comparisons among neurons and among kinematic variables, is necessary to mathematically evaluate the information encoded by neurons about the kinematics variables. If this is achieved, the new tuning depth metric can be utilized to distinguish the neuron’s tuning ability in BMI.
Besides the scalar description of tuning properties, different models are used to describe the tuning properties of the neurons parameterized by a few parameters. However, there is no systematic method to completely characterize how a specific stimulus parameter governs the subsequent response of a given neuron. Linear decoding, proposed by researchers to model the stimulus–response function, has been widely used [Moran & Schwartz, 1999]. The linear filter takes into account the sensitivity of preferred direction, the position and speed of the movement to represent the firing rate in cortical activity [Roitman et al., 2005].

However, linear encoding captures only a fraction of the overall information transmitted because the neuron exhibits nonlinear behavior with respect to the input signal. Brown et al. [2001] used a Gaussian tuning function for the hippocampal pyramidal neurons. Brockwell et al. [2003] assumed an exponential tuning function for their motor cortical data. These nonlinear mathematical models are not optimal for dealing with real data because the tuned cells could have very different tuning properties. Based on the white noise analysis to characterize the neural light response [Chichilnisky, 2001], Simoncelli and Paninski et al. [2004] proposed a cascading linear-nonlinear-Poisson model to characterize the neural response with stochastic stimuli. The spike-triggered average (STA) and the spike-triggered covariance (STC) provided the first linear filter stage in a polynomial series expansion of the tuning function [Paninski, 2003]. This linear filter geometrically directs the high dimensional stimulus to where the statistical moments of spike-triggered ensemble differ most from the raw signals. The nonlinear transformation of the second stage is estimated by an intuitive nonparametric binning technique [Chichilnisky, 2001] as the fraction of the two smoothed histograms. This gives a conditional instantaneous firing rate to the Poisson spike-generating model. The nonlinear stage is then followed by a Poisson generator. This modeling method assumes that the raw stimulus distribution is spherically
symmetric for STA and Gaussian distributed for STC, and that the generation of spikes depends only on the recent stimulus and is historically independent of previous spike times. Both STA and STC fail when the mean or the variance of the spike-triggered ensemble does not differ from the raw ensemble at the direction of the linear filter. For the information-theoretic metric, mutual information was proposed to quantify the predictability of the spike [Paninski & Shoham et al., 2004; Sharpee & Rust et al., 2002]. The multi-linear filters representing the trial directions were found to carry the most information between spikes and stimuli.

The encoding analysis provided a deeper understanding of how neuron spikes respond to a stimulus. This important mathematical modeling holds promise toward providing analytical solutions to the underlying mechanism of neuron receptive fields.

**Decoding algorithms**

In decoding, the biological response is estimated from the neural spike trains. The initial method, a population vector algorithm, was proposed by Georgopoulos et al. [1986], who studied the preferred direction of each cell as its tuning property. Using this method, the movement direction is predicted by a weighted contribution of all cell preferred direction vectors. The weights are represented as a function of a cell’s binned firing rate. The population vector algorithm demonstrated that effective decoding requires a pre-knowledge of the encoding models. A co-adaptive movement prediction algorithm based on the population vector method was developed to track changes in cell tuning properties during brain-controlled movement [Taylor et al., 2002]. Initially random, the estimate of cell tuning properties is iteratively refined as a subject attempts to make a series of brain-controlled movements.

Another decoding methodology uses binned spike trains to predict movement based on linear or nonlinear optimal filters. This method avoids the neurological dynamic encoding model of the neural receptive field, and standard linear or nonlinear regression is used to fit the
relationship directly into the decoding operation. The Wiener filter or time delay neural network (TDNN) was designed to predict the 3D hand using neuronal binned spike rates embedded by a 10-tap delay line [Wessberg et al., 2000]. In addition to this forward model, a recursive multilayer perceptrons (RMLP) model was proposed by Sanchez et al. [2002b] and improved with better performance using only relevant neuronal activities [Sanchez et al., 2002a]. Subsequently, Kim et al. [2003] proposed the development of switching multiple linear models combined with a nonlinear network to increase prediction performance in food reaching. Their regression model performed very well in decoding movement prediction. It is difficult to derive the neurological dynamics properties directly from models; however, this model is yet another viable method to use weight coefficients to analyze the active properties of neurons.

A bridge is needed to link the performance of the adaptive signal processing methods with the knowledge from the receptive field neuron dynamics. These symbioses will greatly improve the present understanding of decoding algorithms.

The probabilistic method based on the Bayesian formulation estimates the biological response from the ensemble spike trains. From a sequence of noisy observations of the neural activity, the probabilistic approach analyzes and infers the response as a state variable of the neural dynamical system. The neural tuning property relates the measurement of the noisy neural activity to the stimuli, and builds up the observation measurement model. Probabilistic state space formulation and information updating depend on the Bayesian approach of incorporating information from measurements. A recursive algorithm based on all available statistical information is used to construct the posterior probability density function of the biological response for each time, and in principle yields the solution to the decoding problem. Movements
can be recovered probabilistically from the multi-channel neural recordings by estimating the expectation of the posterior density or by maximum a posterior.

As a special case, the Kalman filter was applied to BMI that embodied the concepts of neural receptive field properties [Wu et al., 2006]. The Kalman filter assumes strongly that time-series neural activities are generated by kinematic stimulus through a linear system, so the tuning function is a linear filter only. Another strong assumption is that, given the neural spiking activities at every time step, the Gaussianity of the posterior density of the kinematic stimulus, which reduces all the richness of the interactions to second order information (mean and the covariance). These two assumptions may be too restrictive for BMI applications. The particle filter algorithm was also investigated to recover movement velocities from continuous spike binned data [Brockwell et al., 2004]. The particle filter can provide state estimation for a nonlinear system where the tuning function is assumed to be an exponential operation on linear filtered velocities [Schwartz, 1992].

All of the above algorithms, when applied to spike rates, are coarse approaches that lose spike timing resolution and may exclude rich neural dynamics. The primary reason for this limitation is that the sequential state estimation is applied normally to continuous value observations, and cannot be applied directly to discrete point processes. Indeed, when the observation becomes the spike train point process, only the time instance of the spike event matters without amplitude. Initially, Diggle, Liang and colleagues [1995] mentioned the estimation from the point process observations without a specific algorithm. Chan and Ledolter [1995] proposed a Monte Carlo Expectation-maximization (EM) algorithm using the Markov Chain sampling technique to calculate the expectation in the E-step of the EM algorithm. This method later became the theoretical base to derive an EM algorithm for a point process recursive
nonlinear filter [Smith & Brown, 2003]. The algorithm combined the inhomogeneous Poisson model on point process with the fixed interval smoothing algorithm to maximize the expectation of the complete data log likelihood. In this particular case, the observation process is a point process from an exponential family and the natural parameter is modeled as a linear function of the latent process.

A general point process adaptive filtering paradigm was recently proposed [Brown et al. 2001] to probabilistically reconstruct a freely running rat’s position from the discrete observation of the neural firing. This algorithm modeled the neural spike train as an inhomogeneous Poisson process feeding a kinematic model through a nonlinear tuning function. This approach also embodies the conceptual Bayesian filtering algorithm to predict the posterior density by a linear state update equation and revise it with the next observation measurement. More properties of this algorithm were discussed in Frank et al. [2002], Frank and Stanley et al. [2004], and Suzuki and Brown [2005]. The point process filter analogue of the Kalman filter, recursive least squares and the steepest descent algorithms were derived and compared to decode the tuning parameters and state from the ensemble neural spiking activity [Eden et al., 2004]. In this case, the point process analogue of the Kalman filter performs the best because it provides more adjustable step size to update the state, which is estimated from the covariance information. However, the method assumes incorrectly that the posterior density of the state vector, given the discrete observation, is always Gaussian distributed. A Monte Carlo sequential estimation algorithm on point process was addressed as a probabilistic approach to infer the kinematic information directly from the neural spike train [Wang et al., 2006b]. The posterior density of the kinematic stimulus, given the neural spike train was estimated at each time step without the Gaussian
assumptions. The preliminary simulations showed a better velocity reconstruction from the exponentially tuned neural spike train without imposing a Gaussian assumption.

Using all the probabilistic approaches to derive the kinematic information from the neural activity for the BMI requires pre-knowledge of the neuron receptive properties. In other words, the estimation of the tuning function between a kinematic stimulus and neural receptive responses and the good initialization of all the parameters in the algorithm can directly affect the results of the prediction of the primate’s movements in BMI. This is because all the probabilistic approaches are based on the Bayesian formulation to construct the posterior density at each time step from the prior density of the kinematic state, which is the posterior density of previous time step. The population vector algorithm hints that an accurate decoding prediction needs the encoding of the neuron tuning property. For the Bayesian approach, the knowledge of the prior density, including the good initialization of all the parameters and the format of the tuning functions, is also a key step if we want to probabilistically infer an accurate kinematic estimation from the posterior densities.

Outline

We are interested in building an adaptive signal processing framework for Brain Machine Interfaces working directly in the spike domain. The model will include the stochastic time information of neuron activities, which is different from conventional methods working on binned spike rates. The Bayesian approach will convert the decoding of neural activity required in BMIs into a state-estimation problem. The kinematics are described by a dynamic state model and inferred as a state from multi-neuron spike train observation, which is connected with the state through neuron tuning function. The good estimation of the state (decoding) depends on the well educated guess of the tuning property of the neuron (encoding). The schematic is shown in Figure 1-2.
Previous tuning analysis is done on windowed based estimation, which maps kinematics information from a segment to one spike, which is not appropriate when the decoding process tries to infer kinematics online from the spike train. Here we develop an instantaneous model for the tuning properties, which builds a one-to-one mapping from the kinematics state to the neuron spike trains. It would be interesting to also compare the instantaneous estimator with the traditional windowed estimator in term of encoding performance.

We will then implement the Bayesian algorithm to decode the kinematics from spike trains. The non-parametric estimation provides a nonlinear neuron tuning function with no constrains, which goes beyond the Gaussian assumption on the posterior density that is usually made in the previous Bayesian approaches. We are interested in lifting this assumption by designing an algorithm based on Monte Carlo sequential estimation on point process. In this algorithm, the full information of posterior density is estimated without Gaussian constrains in order to gain better performance on state estimation, which, will unfortunately be paid with higher computational complexity. The trade off between the performance and computational cost will be quantified.

In addition to the interest in non-Gaussian assumption, we would also like to investigate the stochasticity and the non-stationary of the neuron behavior in terms of the decoding performance. Due to experimental constraints, only a few neurons are recorded from the motor cortex. To study the effect of stochasticity intrinsic in single neuron representation of a neural assembly in online modeling, several synthetic spike trains are generated from the intensity function estimated from the neurons and utilized as extra model inputs. The decoding performance is averaged across the realizations in the kinematics domain to reduce the variance of original spike recordings as single realization. Lastly, the non-stationary of the neuron
behaviors is studied in the decoding performance of different test data segments with the fixed tuning function. Preliminary results show that a dual Kalman filter approach is able to track the tuning function change in the test data set, which indicates that the non-stationary of the neuron tuning could be promisingly overcome by dual decoding structure.

The outline of the dissertation is the following. In Chapter 2, we review the traditional probabilistic approach for adaptive signal processing as a state estimation problem, followed by our new proposed Monte-Carlo sequential estimation for the point process optimum filtering algorithm. This methodology estimates directly the posterior density of the state given the observations. Sample observations are generated at each time to recursively evaluate the posterior density more accurately. The state estimation is obtained easily by collapse, for example, by smoothing the posterior density with Gaussian kernels to estimate its mean. When tested in a one-channel simulated neuron spike train decoding experiment, our algorithm better reconstructs the velocity as compared with the point process adaptive filtering algorithm with the Gaussian assumption. In Chapter 3, we describe the experimental setups for Brain Machine Interfaces and state the differences between the simulation data and real BMI data. The neuron tuning properties are modeled to instantaneously encode the movement information of the experimental primate as the pre-knowledge for Monte-Carlo sequential estimation for BMI. It is also analyzed and compare in details with the traditional windowed encoding methods. In Chapter 4, the decoding framework for the Brain Machine Interfaces is presented directly in the spike domain and is followed with kinematics reconstruction results and performance analysis comparing to the adaptive filtering algorithm in spike domain with different encoding models. The results by synthetic averaging to reduce and variance of the kinematics prediction and the efforts to reduce the computational complexity by selecting the neuron subset in decoding
process are also presented in Chapter 4. Conclusions and future work, including the preliminary results on tracking the non-stationary neuron tuning property by Dual Kalman filter, are described in Chapter 5.
Figure 1-1. Brain machine interface paradigm

Figure 1-2. Schematic of relationship between encoding and decoding processes for BMIs
CHAPTER 2
PROBABILISTIC APPROACH FOR POINT PROCESS

Sequential State Estimation Problem – Pros and Cons

In sequential state estimation, the system state changes over time with a sequence of noisy measurements observed continuously on the system. The state vector that contains all the relevant information describes the system through a time-series model. Two models are required to analyze and infer the state of a dynamical system, the system model, which describes the evolution of the state with time, and the continuous observation measurement model, which relates the noisy measurements to the state. The probabilistic state space formulation and the updating of information are rooted in the Bayesian approach of incorporating information from measurements. A recursive algorithm based on all available information, including all available statistical information and, in principle, the solution to the estimation problem, is used to construct the posterior probability density function of the state for each observation. Adapting the filter is a two-stage process. The first stage, prediction, uses the system model to predict the posterior probability density of the state given the observation from one measurement to the next; the second stage, updating, revises the predicted posterior probability density based on the latest measurement of the observation. The Kalman filter exemplifies an analytical solution that embodies this conceptual filtering under the assumption that the time-series created by a linear system and the posterior density of the state, given the observation at every step, is Gaussian, hence only parameterized by mean and covariance.

Sequential state estimation can describe the decoding problem in Brain Machine Interfaces. Information on the primate’s movements can be regarded as the state, which changes over time through a kinematic dynamic system model. The neuron spike trains functionally encode the kinematic states, and this can be designed as a tuning function. This tuning function acts as the
observation model in the state sequential estimation problem. It probabilistically models the randomness of the neuron behaviors and characterizes the nonlinear neuron firing properties with the preferred kinematic directions, thereby describing the neuron receptive fields from the neurophysiologic point of view. The parameters of the tuning function can also represent the state changing slowly over the time, suggesting a possible investigation of the nonstationary aspects of neuron tuning properties. The Brain Machine Interface then converts the observations of multi-channel neuron spike trains to infer the kinematics as the state. This approach is problematic in BMI because channels of neuron spike trains are multi-dimensional observations driven by one state vector. A possible assumption is that all the neuron spike trains are motivated independently with the cooperation of the kinematic information, but this may not be true.

Another problem with this method is that the probabilistic approach is based on the Bayesian formulation, which constructs the posterior density from the prior recursively. To develop a good estimation of the states, the information describing how the system works must correspond with the pre-knowledge of the kinematic dynamics system and the neuron tuning function.

**Review of the Previous Probabilistic Approaches**

In Chapter 1, we reviewed several probabilistic approaches to decode the neuron activities that take place during a primate’s movement. The probabilistic methods investigated and applied to BMI by different research groups include the Kalman filter [Wu & Gao *et al.*, 2006], and the particle filter algorithm [Brockwell & Rojas *et al.*, 2004]. Both of these algorithms employ concepts of sequential state estimation. The usefulness of the Kalman filter is limited in that it reduces all the richness of the interactions to second order information (mean and the covariance) because it assumes the linear tuning property and the Gaussianity of the posterior density of the movements given the neural spiking activities at every time step. Although the particle filter provides state estimation for a nonlinear system, the tuning function was directly assumed to be
an exponential operation on linear filtered velocities [Schwartz, 1992]. Both of the algorithms were applied to continuous spike binned data and cannot be directly adapted to discrete point processes. A point process adaptive filtering algorithm was recently proposed by Brown et al. [2001]. In their approach, discrete observations of the neural firing spikes were utilized as the state to probabilistically reconstruct the position of a freely running rat in space. This approach also reflects the conceptual Bayesian filtering algorithm to predict the posterior density by a linear state, update the equation and then revise it with the next observation measurement. However, given the discrete observation, this method assumes that the posterior density of the state vector is always Gaussian distributed, which may not be the case. We proposed a probabilistic filtering algorithm to reconstruct the state from the discrete observation – the spiking event – by generating a sequential set of samples to estimate the distribution of the state posterior density without the Gaussian assumption. The posterior density is recursively propagated and revised by sequential spike observations over time. The state at each time is determined by the maximum a posterior or the expectation of the posterior density inferred by a collapsing of the mixture of Gaussian kernels when estimating the posterior density. The algorithm will be described in the next section, followed by an illustration of algorithm performance in a simulated neuron decoding example and a comparison to the probabilistic velocity reconstruction with Gaussian assumption on posterior density.

**Adaptive Algorithms for Point Processes**

In this section, we review the design of adaptive filters for point processes under the Gaussian assumption, and then introduce our method, a Monte Carlo sequential estimation, to probabilistically reconstruct the state from discrete (spiking) observation events.
Adaptive Filtering for Point Processes with Gaussian Assumption

One can model a point process using a Bayesian approach to estimate the system state by evaluating the posterior density of the state given the discrete observation [Eden & Frank et al., 2004]. This framework provides a nonlinear time-series probabilistic model between the state and the spiking event [Brown et al., 1996].

Given an observation interval \((0, T]\), the number \(N(t)\) of events (spikes) can be modeled as a stochastic inhomogeneous Poisson process characterized by its conditional intensity function \(\lambda(t \mid x(t), \theta(t), H(t))\) (i.e., the instantaneous rate of events), defined as

\[
\lambda(t \mid x(t), \theta(t), H(t)) = \lim_{\Delta \to 0} \frac{\Pr(N(t + \Delta t) - N(t) = 1 \mid x(t), \theta(t), H(t))}{\Delta t}
\]  

(2-1)

where \(x(t)\) is the system state, \(\theta(t)\) is the parameter of the adaptive filter, and \(H(t)\) is the history of all the states, parameters and the discrete observations up to time \(t\). The relationship between the single parameter Poisson process \(\lambda\), the state \(x(t)\), and the parameter \(\theta(t)\) is a nonlinear model represented by

\[
\lambda(t \mid x(t), \theta(t)) = f(x(t), \theta(t))
\]

(2-2)

Using the nonlinear function \(f(\cdot)\), assumed to be known or specified according to the application, let us consider hereafter the parameter \(\theta(t)\) as part of the state vector \(x(t)\). Given a binary observation event \(\Delta N_k\) over the time interval \((t_{k-1}, t_k)\), the posterior density of the whole state vector \(x(t)\) at time \(t_k\) can be represented by Bayes’ rule as

\[
p(x_k \mid \Delta N_k, H_k) = \frac{p(\Delta N_k \mid x_k, H_k)p(x_k \mid H_k)}{p(\Delta N_k \mid H_k)}
\]

(2-3)

where \(p(\Delta N_k \mid x_k, H_k)\) is the probability of observing spikes in the interval \((t_{k-1}, t_k)\), considering the Poisson process.
Pr(ΔNₖ | xₖ, Hₖ) = (λ(tₖ | xₖ, Hₖ)Δt)ΔNₖ exp(−λ(tₖ | xₖ, Hₖ)Δt) (2-4)

and \( p(xₖ | Hₖ) \) is the one-step prediction density given by the Chapman-Kolmogorov Equation as

\[
p(xₖ | Hₖ) = \int p(xₖ | x_{k-1}, Hₖ)p(x_{k-1} | ΔN_{k-1}, H_{k-1})dx_{k-1} \tag{2-5}
\]

where the state \( xₖ \) evolves according to the linear relation

\[
xₖ = Fₖx_{k-1} + \etaₖ \tag{2-6}
\]

\( Fₖ \) establishes the dependence on the previous state and \( \etaₖ \) is zero-mean white noise with covariance \( Qₖ \). Substituting Equations 2-4 and 2-5, in 2-3 the posterior density of the state \( p(xₖ | ΔNₖ, Hₖ) \) can be recursively estimated from the previous one based on all the spike observations.

Assuming the posterior density given by Equation 2-3 and the noise term \( \etaₖ \) in the state evolution Equation 2-6 are Gaussian distributed, the Chapman-Kolmogorov Equation 2-5 becomes a convolution of two Gaussian curves, from which the estimation of the state at each time has a closed form expression given by (see [Eden et al., 2004] for details).

\[
x_{k|k-1} = Fₖx_{k-1} \quad (2-7a)
\]

\[
W_{k|k-1} = FₖW_{k-1|k-1}Fₖ + Qₖ \quad (2-7b)
\]

\[
(Wₖ)^{-1} = (W_{k|k-1})^{-1} + \left[ \left( \frac{∂\log λ}{∂xₖ} \right) [λΔtₖ] \left( \frac{∂\log λ}{∂xₖ} \right) - (ΔNₖ - λΔtₖ) \frac{∂²\log λ}{∂xₖ^2} \right]_{x_{k|k-1}} \quad (2-7c)
\]

\[
x_{k|k} = x_{k|k-1} + Wₖ\left[ \left( \frac{∂\log λ}{∂xₖ} \right) (ΔNₖ - λΔtₖ) \right]_{x_{k|k-1}} \quad (2-7d)
\]

The Gaussian assumption was used initially because it allows one to solve analytically Equation 2-5 and therefore, for a closed form solution of Equation 2-3 as Equation 2-7.
Although the above set of equations may seem daunting, each can be interpreted quite easily. First, Equation 2-7a establishes a prediction for the state based on the previous state. Then, Equations 2-7b and 2-7c are used in Equation 2-7d to correct or refine the previous estimate, after which the recurrent process is repeated.

**Monte Carlo Sequential Estimation for Point Processes**

The Gaussian assumption applied to the posterior distribution in the algorithm just described may not be true in general. Therefore, in terms of discrete observations, a non-parametric approach is developed here which poses no constraints on the form of the posterior density.

Suppose at time instant $k$ the previous system state is $x_{k-1}$. Recall that because the parameter $\theta$ was embedded in the state, we need only the estimation of the state from the conditional intensity function Equation 2-1, since the nonlinear relation $f(\cdot)$ is assumed known. Random state samples are generated using Monte Carlo simulations [Carpenter & Clifford et al., 1999] in the neighborhood of the previous state according to Equation 2-6. Then, weighted Parzen windowing [Parzen, 1962] was used with a Gaussian kernel to estimate the posterior density. Due to the linearity of the integral in the Chapman-Kolmogorov Equation and the weighted sum of Gaussians centered at the samples, we are still able to evaluate directly from integral samples. The process is repeated recursively for each time instant, propagating the estimate of the posterior density, and the state itself, based on the discrete events over time. Notice that due to the recursive approach, the algorithm not only depends on the previous observation, but also depends on the entire path of the spike observation events.

Let $\{x_{0:k}, \omega_{i}^{j}\}_{j=1}^{N_s}$ denote a Random Measure [Arulampalam & Maskell et al., 2002] in the posterior density $p(x_{0:k} \mid N_{1:k})$, where $\{x_{0:k}, i = 1, \cdots, N_s\}$ is the set of all state samples up to time
with associated normalized weights \( \{w_k^i, i = 1, \cdots, N_s\} \), and \( N_s \) is the number of samples generated at each time index. Then, the posterior density at time \( k \) can be approximated by a weighted convolution of the samples with a Gaussian kernel as

\[
p(x_{0:k} \mid N_{1:k}) \approx \sum_{i=1}^{N_k} w_k^i \cdot k(x_{0:k} - x_{0:k}^i, \sigma)
\]

where \( N_{1:k} \) represents the spike observation events up to time \( k \) modeled by an inhomogeneous Poisson Process described in the previous section, and \( k(x - \tilde{x}, \sigma) \) is the Gaussian kernel in terms of \( x \) with mean \( \tilde{x} \) and covariance \( \sigma \). By generating samples from a proposed density \( q(x_{0:k} \mid N_{1:k}) \) according to the principle of Importance Sampling [Bergman, 1999; Doucet, 1998], which usually assumes dependence on \( x_{k-1} \) and \( N_k \) only, the weights can be defined by

\[
w_k^i \propto \frac{p(x_{0:k}^i \mid N_{1:k})}{q(x_{0:k}^i \mid N_{1:k})}
\]

(2-9)

Here, we assume the importance density obeys the properties of Markov Chain such that

\[
q(x_{0:k} \mid N_{1:k}) = q(x_k \mid x_{0:k-1}, N_{1:k})q(x_{0:k-1} \mid N_{1:k-1})
= q(x_k \mid x_{k-1}, \Delta N_k)q(x_{0:k-1} \mid N_{1:k-1})
\]

(2-10)

At each time iteration, the posterior density \( p(x_{0:k} \mid N_{1:k}) \) can be derived and approximated by the posterior density in the previous iteration as Equation 2-11.

\[
p(x_{0:k} \mid N_{1:k}) = \frac{p(\Delta N_k \mid x_{0:k}, N_{1:k-1})p(x_{0:k} \mid N_{1:k-1})}{p(\Delta N_k \mid N_{1:k-1})}
= \frac{p(\Delta N_k \mid x_{0:k}, N_{1:k-1})p(x_k \mid x_{0:k-1}, N_{1:k-1})p(x_{0:k-1} \mid N_{1:k-1})}{p(\Delta N_k \mid N_{1:k-1})}
= \frac{p(\Delta N_k \mid x_k)p(x_k \mid x_{k-1})}{p(\Delta N_k \mid N_{1:k-1})} \times p(x_{0:k-1} \mid N_{1:k-1})
\]

36
By replacing Equations 2-10 and 2-11 into Equation 2-9, the weight can be updated recursively as Equation 2-12.

\[
W_k^i \propto \frac{p(\Delta N_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i) p(x_{0:k-1} \mid N_{1:k-1})}{q(x_k^i \mid x_{k-1}^i, \Delta N_k) q(x_{0:k-1}^i \mid N_{1:k-1})} \frac{p(\Delta N_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{q(x_k^i \mid x_{k-1}^i, \Delta N_k)} W_{k-1}^i
\]

(2-12)

Usually the importance density \(q(x_k^i \mid x_{k-1}^i, \Delta N_k)\) is chosen to be the prior density \(p(x_k^i \mid x_{k-1}^i)\), requiring the generation of new samples from \(p(x_k^i \mid x_{k-1}^i)\) by Equation 2-6 as a prediction stage.

After the algorithm is applied for a few iterations, a phenomenon called degeneracy may arise, where all but one sample has negligible weight [Doucet, 1998], implying that a large computational effort is taken to update the samples that have almost no contribution to estimate the posterior density. When a significant degeneracy appears, resampling is applied to eliminate the samples with small weight and to concentrate on samples with large weights according the samples \(cdf\). In our Monte Carlo sequential estimation of the point process, Sequential Importance Resampling [Gordon & Salmond et al. 1993] is applied at every time index, so that the sample is \(i.i.d.\) from the discrete uniform density with weights \(w_{k-1}^i = 1/N_S\). The pseudo code of the scheme to resample \(\{x_k^i, w_k^i\}_{i=1}^{N_k}\) to \(\{x_k^{i*}, w_k^{i*}\}_{i=1}^{N_S}\) is the following [Arulampalam & Maskell et al., 2002].

- Initial the \(cdf\): \(c_1 = 0\);
- For \(i = 2 : N_S\)
  --construct the \(cdf\): \(c_i = c_{i-1} + w_k^i\)
- End For
- Start at the bottom of the \(cdf\): \(i=1;\)
• Draw a starting point: $u \sim U[0, N^{-1}_s]$

• For $j=1:N$
  -- move along the cdf: $u_j = u_{i+1} + N^{-1}_s (j - 1)$
  -- While $u_j > c_i$
    * $i = i + 1$
  -- end While
  -- Assign sample $x_k^{i*} = x_k^i$
  -- Assign weight $w_k^i = 1/ N_s$

• End For

The weights then change proportionally, given by

$$w_k^i \propto p(\Delta N_k \mid x_k^i)$$  \hspace{1cm} (2-13)$$

where $p(\Delta N_k \mid x_k^i)$ is defined by Equation 2-4 in this section. Using Equations 2-6, 2-13 and the resampling step, the posterior density of the state $x_k$ given the whole path of the observed events up to time $t_k$ can be approximated as

$$p(x_k \mid N_{t_k}) \approx \sum_{i=1}^{N_k} p(\Delta N_k \mid x_k^i) \cdot k(x_k - x_k^i)$$  \hspace{1cm} (2-14)$$

Equation 2-14 shows that, given the observation, the posterior density of the current state is modified by the latest probabilistic measurement of the observing spike event $p(\Delta N_k \mid x_k^i)$, which is the updating stage in adaptive filtering.

Without a close form of the state estimation, we measure the posterior density of the state given the observed spike event $p(x_k \mid N_{t_k})$ every time and apply two methods to get the state estimation $\tilde{x}_k$. One method is Maximum A Posterior (MAP), which picks out the sample $x_k^{i*}$ with maximum posterior density. The second method is to use the expectation of the posterior density as the state estimation. As we smooth the posterior density by convolving with a
Gaussian kernel, we can easily obtain the expectation $\bar{x}_k$ and its error covariance $V_k$ by collapse [Wu & Black et al., 2004].

$$\bar{x}_k = \sum_{i=1}^{N_k} p(\Delta N_k \mid x^i_k) \cdot x^i_k$$ (2-15)

$$V_k = \sum_{i=1}^{N_k} p(\Delta N_k \mid x^i_k) \cdot (\sigma + (x^i_k - \bar{x}_k)(x^i_k - \bar{x}_k)^T)$$ (2-16)

From Equations 2-15 and 2-16, we can see that without complex computation we can easily estimate the next state. Hence, the expectation by collapse is simple and elegant.

The major drawback of the algorithm is computational complexity because the quality of the solution requires many particles $\{x^i_{0:k}, i = 1, \ldots, N_S\}$ to approximate the posterior density. Smoothing the particles with kernels as in Equation 2-14 alleviates the problem in particular when collapsing is utilized, but still the computation is much higher than calculating the mean and covariance of the PDF with a Gaussian assumption.

We have to point out that both approaches assume we know the state model $F_k$ in Equation 2-6 and the observation model $f(\cdot)$ in Equation 2-2, which actually are unknown in real applications. The state model is normally assumed linear and the parameters are obtained from the data using least squares. The knowledge of the observation model is very important for decoding (deriving states from observations), because the probabilistic approach based on Bayesian estimation constructs the posterior density of each state given the spike observation at each time step from the prior density of the state. The prior density in turn is the posterior density of previous time step updated with the discrepancy between an observation model and the spike event. The observation model basically quantifies how each neuron encodes the kinematic
variables (encoding), and due to the variability of neural responses it should be carefully estimated from a training set for the purpose of Monte Carlo decoding models.

**Simulation of Monte Carlo Sequential Estimation on Neural Spike Train Decoding**

Neurons dynamically change their responses to specific input stimuli patterns through learning, which has been modeled with the help of receptive fields. Neural decoding can be used to analyze receptive field plasticity and understand how the neurons learn and adapt by modeling the tuning function of neuronal responses. In the rat hippocampus, for example, information about spatial movement can be extracted from neural decoding, such as from the activity of simultaneously recorded noisy place cells [Mehta & Quirk et al., 2000, O’Keefe & Dostrovsky, 1971] representing the spike-observed events.

In a conceptually simplified motor cortical neural model [Moran & Schwartz, 1999], the one-dimensional velocity can be reconstructed from the neuron spiking events by the Monte Carlo sequential estimation algorithm. This algorithm can provide a probabilistic approach to infer the most probable velocity as one of the components of the state. This decoding simulation updates the state estimation simultaneously and applies this estimation to reconstruct the signal, which assumes interdependence between the encoding and decoding so that the accuracy of the receptive field estimation and the accuracy of the signal reconstruction are reliable.

Let us first explain how the simulated data was generated. The tuning function of the receptive field that models the relation between the velocity and the firing rate is assumed exponential and given by

\[ \lambda(t_k) = \exp(\mu + \beta_k v_k) \]  

(2-17)

where \( \exp(\mu) \) is the background firing rate without any movement and \( \beta_k \) is the modulation in firing rate due to the velocity \( v_k \). In practice in the electrophysiology lab, this function is
unknown. Therefore, an educated guess needs to be made about the functional form, for which the exponential function is widely utilized.

The desired velocity was generated as a frequency modulated (chirp) triangle wave added with Gaussian noise (variance $2.5 \times 10^{-5}$) at each 1ms time step, as shown in Figure 2-1. The design of the desired signal enables us to check if the algorithm could track the linear evolution and the different frequency of the “movement”.

The background-firing rate $\exp(\mu)$ and the modulation parameter $\beta_k$ are set to be 1 and 3 respectively for the whole simulation time, 60s. A neuron spike is drawn as a Bernoulli random variable with probability $\lambda(t_k)\Delta t$ within each 1ms time window [Brown et al. 2002]. A realization of a neuron spike train is shown in Figure 2-2.

With the exponential tuning function operating on the velocity, we can see that when the velocity is negative, there are few spikes; while when the velocity is positive, many spikes appear. The problem is to obtain from this spike train the desired velocity of Figure 2-2, assuming the Poisson model of Equation 2-17 and one of the sequential estimation techniques discussed.

To implement the Monte Carlo sequential estimation for the point process, we regard both modulation parameter $\beta_k$ and velocity $v_k$ as the state $x_k = [v_k \beta_k]^T$. Here we set 100 samples to initialize the velocity $v_0'$ and modulation parameter $\beta_k$ respectively with a uniform and with a Gaussian distribution. Note that too many samples would increase the computational complexity dramatically, while an insufficient number of samples would result on a poor description of the non-Gaussian posterior density. The new samples are generated according to the linear state evolution Equation 2-6, where $F_k$ is obtained from the data using least squares for $v_k$ and 1 for
\( \beta_k \) (implicitly assuming that the modulation parameter would not change very fast). The \( i.i.d. \) noise for velocity state in Equation 2-6 was drawn from the distribution of the error between the true velocity and the linear predicted results by \( F_k \). The \( i.i.d. \) noise for estimating the modulation parameter \( \beta_k \) is approximated by a zero mean Gaussian distribution with variance \( Q_k \) (default \( 10^{-7} \)). The kernel size utilized in Equation 2-14 to estimate the maximum of the posterior density (thru MAP) obeys Silverman’s rule [Silverman 1981]. Because the spike train is generated according to the Poisson model, there is stochasticity involved. We then generate 10 sets of the spike train from the same time series of the firing rate by the tuning function Equation 2-17 from the desired velocity. The averaged performances evaluated by NMSE between the desired trajectory and the model output are shown in Table 2-1, for different runs of the covariance matrices of the state generation \( Q_k \). Notice that the noise variance should be small enough to track the unchanged \( \beta_k \) set in the data. In general, if \( Q_k \) is too large, the continuity constraint of the whole sequential sample generated has little effect. If it is too small, this constraint may become too restrictive and the reconstructed velocity may get stuck in the same position while the real velocity moves away by a distance much larger than \( Q_k \).

In order to obtain realistic performance assessments of the different models (Maximum a posteriori and collapse), the state estimations \( \tilde{v}_k, \tilde{\beta}_k \) for the duration of the trajectory are drawn 10 different times. The best velocity reconstruction is shown in Figure 2-3. The Normalized Mean Square Error (MSE normalized by the power of the desired signal) between the desired trajectory and the model output for the adaptive filtering with Gaussian assumption is 0.3254. NMSE for sequential estimation by MAP is 0.2352 and by collapse is 0.2140.
From Figure 2-3, we can see that compared with the desired velocity (dash-dotted red line), all the methods obtain close estimation when there are many spikes (i.e., when the velocity is at the positive peaks of the triangle wave). This is because the high likelihood of spikes corresponds to the range of the exponential tuning function where the modulation of the high firing probability is easily distinguished and the posterior density is close to the Gaussian assumption. However, in the negative peaks of the desired velocity the sequential estimation algorithm (using collapse for expectation or MAP) performs considerably better. This is primarily because the modulation of the firing rate is nonlinearly compressed by the exponential tuning function, leading to non-Gaussian posterior densities, and thus violating the Gaussian assumption the adaptive filtering method relies on. Although there is nearly no neuronal representation for negative velocities and therefore both algorithms are inferring the new velocity solely on the previous state, the non-parametric estimation of the pdf in the sequential estimation algorithm allows for more accurate inference. As an example in Figure 2-4A, the posterior density at time 6.247s (when the desired velocity is close to the positive peak) is shown (dotted pink line) Gaussian-like shape, all the methods provide similar estimations close to the true value (red star). In Figure 2-4B, the posterior density at time 35.506s (when the desired velocity is close to the negative peak) is shown (dotted pink line) non-symmetric with 2 ripples and is obviously not Gaussian distributed. The adaptive filtering on point process under a Gaussian assumption provides poor estimation (gray dotted line), not only because of its Gaussian assumption but also because the algorithm propagates the poor estimation from previous time resulting in an accumulation of errors. The velocity estimated by the sequential estimation with collapse denoted by the blue circle is the closest to the desired velocity (red star). Notice also that in all cases the tracking performance gets progressively worse as the frequency increases. This is
because the state model is fixed from the whole set of the data by a linear model, which tracks
the velocity state at the average frequency. If a time-variant state model is used on a segment-by-
segment basis, we could expect better reconstructions.

In summary, the Monte Carlo sequential estimation on point processes seems promising to
estimate the state from the discrete spiking events.

Interpretation

Point process adaptive filtering is a two-step Bayesian approach based on the Chapman-
Kolmogorov Equation to estimate parameters from discrete observed events. However, the
Gaussian assumption of posterior density of the state, upon observation, may not accurately
represent state reconstruction due to the less accurate evaluation of posterior density. We present
in this paper a Monte Carlo sequential estimation to modify the amplitude of the observed
discrete events by the probabilistic measurement, posterior density. A sequence of samples is
generated to estimate the posterior density more accurately. Through sequential estimation and
weighted Parzen windowing, we avoid the numerical computation of the integral in the C-K
Equation. By smoothing the posterior density with the Gaussian kernel from Parzen windowing,
we can collapse to easily derive the expectation of the posterior density, leading to a better result
of state estimate than noisy Maximum a posterior. The Monte Carlo estimation shows better
capability to probabilistically estimate the state because it better approximates the posterior
density than does the point process adaptive filtering algorithm with Gaussian assumption.
Figure 2-1. The desired velocity generated by triangle wave with Gaussian noise

Figure 2-2. The simulated neuron spike train generated by an exponential tuning function

Table 2-1. Comparison results of all algorithms with different $Q_k$

<table>
<thead>
<tr>
<th>$Q_k$</th>
<th>Adaptive filtering of point process</th>
<th>Sequential estimation Collapse</th>
<th>Sequential estimation MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-5}$</td>
<td>0.4434</td>
<td>0.3803</td>
<td>0.3881</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>0.3940</td>
<td>0.3575</td>
<td>0.3709</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>0.3583</td>
<td>0.2956</td>
<td>0.3252</td>
</tr>
</tbody>
</table>
Figure 2-3. The velocity reconstruction by different algorithms

Figure 2-4. $p(v_{i,k} | \Delta N_{i,k})$ at different time. A) At time 6.247s. B) At time 35.506s
Figure 2-4. Continued
CHAPTER 3
INFORMATION THEORETICAL ANALYSIS OF INSTANTANEOUS MOTOR CORTICAL NEURON ENCODING

Experimental Setups

In Chapter 2, we presented a Monte Carlo sequential estimation algorithm to reconstruct the continuous state variable directly from point process observations. In the one-neuron spike train decoding simulation, this algorithm provided a better estimate of the state recursively without Gaussian distribution. The Monte Carlo sequential estimation in spike domain is a promising signal processing tool to decode the continuous kinematics variable directly from neural spike trains in Brain Machine Interfaces. With this method, spike binning window size is no longer a concern, as one can directly utilize the spike timing event. The online state estimation is suitable for real-time BMIs decoding without the desired signal; however, both the neural activity recoding and desired trajectories are required to estimate the neuron tuning function. The decoding results by Monte Carlo estimation could be different between realizations because of the random manner in which samples are generated to construct the posterior density.

Data Recording

The Brain-Machine Interface paradigm was designed and implemented in Dr. Miguel Nicolelis laboratory at Duke University. Chronic, neural ensemble recordings were collected from the brain of an adult female Rhesus monkey named Aurora, and synchronized with task behaviors.

Several micro-electrode arrays were chronically implanted in five of the monkey’s cortical neural structures, right dorsolateral premotor area (PMA), right primary motor cortex (MI), right primary somatosensory cortex (S1), right supplementary motor area (SMA), and the left primary motor cortex (MI). Each electrode array consisted of up to 128 microwires (30 to 50 μm in diameter, spaced 300 μm apart), distributed in a 16 × 8 matrix. Each recording site occupied a
total area of 15.7 mm² (5.6 × 2.8 mm) and was capable of recording up to four single cells from each microwire for a total of 512 neurons (4 × 128) [Sanchez, 2004].

After the surgical procedure, a multi-channel acquisition processor cluster (MAP, Plexon, Dallas, TX) was used in the experiments to record the neuronal action potentials simultaneously. Analog waveforms of the action potential were amplified and band pass filtered from 500 Hz to 5 kHz. The spikes of single neurons from each microwire were discriminated based on time-amplitude discriminators and a principal component analysis (PCA) algorithm [Nicolelis et al., 1997; Wessberg et al. 2000]. The firing times of each spike were stored. Table 3-1 shows the assignment of the sorted neural activity to the electrodes for different motor cortical areas [Kim 2005].

The monkey performed a two-dimensional target-reaching task to move the cursor on a computer screen by controlling a hand-held joystick to reach the target (Figure 3-1). The monkey was rewarded when the cursor intersected the target. The corresponding position of the joystick was recorded continuously for an initial 30-min period at a 50 Hz sampling rate, referred to as the “pole control” period [Carmena & Lebedev et al. 2003].

The monkey performed a two-dimensional target-reaching task to move the cursor on a computer screen by controlling a hand-held joystick to reach the target (Figure 3-1). The monkey was rewarded when the cursor intersected the target. The corresponding position of the joystick was recorded continuously for an initial 30-min period at a 50 Hz sampling rate, referred to as the “pole control” period [Carmena et al., 2003].
**Simulation vs. Vivo Recordings**

BMI data provides us with 185 neural spike train channels and 2-dimensional movement trajectories for about 30 minutes. Compared to the one-neuron decoding simulation in Chapter 2, there are big differences.

At first glance, it is remarkable that the time resolution for the neural spike train is about a millisecond, while the movement trajectories have a sampling frequency 50Hz. The neural spike trains allow us to more closely observe the true random neural behavior. Consequently, however the millisecond scale requires more computational complexity. We must bridge the disparity between the microscopic neural spikes and the macroscopic kinematics.

The tuning function provides a basis on which to build a simultaneously functional relationship. In the simulation, we simply assume that the tuning function characterizes the exponentially increasing firing rate conditioned on the velocity. For the real BMI data, is this tuning function still valid and cogent? As presented in Chapter 2, our Monte Carlo sequential estimation algorithm works as probabilistic approach directly in the spike domain. The major assumption supporting the entire algorithm is that we have enough knowledge of both the system model and the observation model. This assumption establishes a reliable base to propagate the posterior density leading to the state estimation at each time iteration. How can we obtain the knowledge? The work by Georgopoulos and Schwartz *et al.* [1986] provides some guidance. The population coding presented in their paper analyzed the individual neural activities tuned broadly to a particular direction. Based through trials on the weighted distribution of individual neurons to the preferred direction, the direction of movement was found to be uniquely predicted. The principle behind this work is *letting the data speak for itself!* We gain insight into neural tuning properties by analyzing the existing neuron and kinematics data. This analysis leads to better kinematics decoding from neural activities in the future.
Another issue to resolve is dealing with multi-channel neural spike trains when there is only one neural channel in the simulation. In the real BMI data, how can we account for the association between channels? In Chapter 1, we reviewed the work done by many researchers in this field with multiple outcomes. Most of the work focused on the exclusive relationship between neural activities, such as the correlation between neurons characterized by the neural firing, or between neuron microscopic spiking and field potentials. With regard to both external kinematics and neural activities, neural spike trains between channels are usually assumed to be conditionally independent of kinematics. In other words, spike generation is determined once the kinematics and parameters of the neuron tuning are known. We should emphasize that the assumption of conditional independence does not conflict with the association analysis between neurons. If the firing rates of two neurons are generated independently through two similar tuning functions in a certain time period, similar firing patterns are expected during this time period, and the analysis on the correlation between them is still valid.

**Review of Tuning Analysis**

The probabilistic approach based on Bayesian estimation constructs the posterior density of each kinematic state given the spike trains at each time step from the prior density of the state. The prior density in turn is the posterior density of previous time step updated with the discrepancy between an observation model and the spike train. The observation model linking the measurement of the noisy neural activity to the kinematics implicitly utilizes the tuning characteristics of each neuron. In our newly proposed Monte Carlo sequential estimation algorithm operating directly on point processes [Wang et al., 2006b], the Bayesian approach analyzes and infers the kinematics as a state variable of the neural dynamical system without the constraints of linearity and Gaussianity. Accurate modelling of the neuron tuning properties in
the observation model is critical to decode the kinematics by expectation of the posterior density or by maximum a posterior.

The tuning, also called the encoding function, mathematically models how a neuron represents behavioral consequences or how it responds to a stimulus. The parameterization of a tuning function requires an understanding of three interconnected aspects 1) What is the behavior/stimulus? 2) How does the neuron encode it? 3) What is the criterion for quantifying the quality of the response? For neurons located in the motor cortex, researchers first developed the static descriptions of movement-related activity by applying electrical stimuli to motor areas to elicit muscle contraction [Fritsch & Hitzig, 1870; Leyton & Sherrington, 1917; and Schafer 1900]. Later, movement direction was correlated with cortical firing in a center-out task where the tuning function was initially modelled as a cosine curve [Georgopoulos et al., 1982]. The peak discharge rate of a cell is called preferred direction. To quantify the degree of tuning, the tuning depth has been proposed as a metric and it is defined as the difference between the maximum and minimum values in the firing rates, normalized by the standard deviation of the firing rate [Carmena et al., 2003; Sanchez et al., 2003]. As a scalar, the tuning depth summarizes the statistical information contained in the tuning curve to evaluate the neural representation, which indicates how modulated the cell’s firing rate is to the kinematic parameter of interest. However, this metric has some shortcomings since it can exaggerate the value of tuning depth when the neuron firing rate standard deviation is close to 0. Additionally, it depends on the binning window size to calculate the firing rate of the neuron. The tuning depth also relates to the scale of the behavior/stimulus and makes the analysis not comparable among neurons as we will see. A more principled metric, allowing comparisons among neurons and among kinematic variables, is necessary to mathematically evaluate the information encoded by neurons about the
kinematics variables. If this is achieved, the new tuning depth metric can be utilized to distinguish the neuron’s tuning ability in BMI.

In addition to tuning depth, Researchers have also proposed a variety of parametric models to describe the motor representation neurons. Linear relationships from motor cortical discharge rate to speed and direction have been constructed [Moran & Schwartz, 1999]. The linear filter took into account the sensitivity of preferred direction, the position and speed of the movement to represent the firing rate in cortical activity [Roitman et al., 2005]. However, linear encoding captures only a fraction of the overall information transmitted because the neuron exhibits nonlinear behavior with respect to the input signal. Brown et al. [2001] used a Gaussian tuning function for the hippocampal pyramidal neurons. Brockwell et al. [2003] assumed an exponential tuning function for their motor cortical data. These nonlinear mathematical models are not optimal for dealing with real data because the tuned cells could have very different tuning properties. Simoncelli and Paninski et al. [2004] further improved the linear idea and proposed a Liner-Nonlinear-Poisson (LNP) model to cascade the linear stage with a nonlinear transformation as the second stage, which gives a conditional instantaneous firing rate to the Poisson spike generating model at the third stage.

In the LNP model, the position or velocity at all relevant times within a temporal window was utilized to extract the information between neuronal activity and animal movement trajectories. During a continuous target tracking task, Paninski et al. [2004b] studied the temporal dynamics of M1 neurons for position and velocity of hand motion given the firing rate. The linear filter in the LNP model averages the temporal position or velocity within the window and so it smoothes the statistical curves on the stimulus distribution and provides the widely known exponential increasing nonlinearity that relates neuronal firing rate to the projected
kinematics. Unfortunately, the averaging builds up an $N$ to 1 temporal mapping between the kinematic variables (position or velocities) and the neural spikes that negatively impacts our goal of building sequential estimation algorithms. Indeed, sequential models require inferring the estimation of the kinematics from current neuronal spike times. Therefore, the instantaneous one to one functional tuning relationship between the kinematics and neuron activities are needed to decode the kinematics online and to avoid the error accumulation within the windowed kinematic vector. Moreover, the analysis of the receptive fields of motor cortex neurons is different from the stimulus-response analysis in sensory cortices, because there is always a time delay between the initiation of the neuron spiking and the movement response. This delay must be taken into consideration in BMI decoding algorithms. The estimation of instantaneous tuning parameters is more difficult and more prone to errors, therefore we will have to evaluate how much nonlinearity properties still holds or changes compared to the temporal kinematic vectors.

In the literature, mutual information has been used to differentiate the raw stimulus ensemble from the spike-triggered stimulus distribution [Simoncelli et al., 2004; Sharpee et al., 2002], as well as to estimate the minimal number of delay samples in the temporal kinematics needed to represent the information extracted by the full preferred trajectory of a given cell [Paninski et al., 2004b]. In this Chapter 3, we also apply an information theoretical analysis but on the instantaneous tuning properties of the motor cortical neurons. We propose the concept of mutual information to estimate tuning depth to analyze the information that neurons in different cortical areas share with respect to the animal’s position, velocity and acceleration. This criterion is first tested in synthetic data, and then applied to motor cortex data. We elaborate how to build our instantaneous tuning function of motor cortical neurons for BMIs. The information theoretical analysis is applied to the projective nonlinear-Poisson encoding analysis to estimate
the causal time delay. The nonlinearity of the instantaneous tuning curves is compared to the
method computed from windowed kinematics.

**Visual Inspection of a Tuning Neuron**

Neurophysiologic evidence suggests that neurons encode the direction of hand movements
with cosine shaped tuning curves [Georgopoulos *et al.*, 1982]. For each neuron, the polar plot of
the neuron activity with regard to a kinematic vector, such as hand position, hand velocity, or
hand acceleration, is investigated to compute the kinematic direction as an angle between 0 and
360 degrees. 45 degree bins are chosen to coarsely classify all the directions into 8 bins. For each
direction, the average neuron firing rate obtained by binning defines the magnitude of the vector
in a polar plot. For a tuned neuron, the average firing rate in each direction is expected to be
quite different. The preferred direction is computed using circular statistics [Jammalamadaka &
SenGupta, 1999] as

\[
\text{circular mean} = \arg\left(\sum_N r_N e^{i\theta_N}\right)
\]  

(3-1)

where \( r_N \) is the neuron’s average firing rate for angle \( \theta_N \), and \( N \) covers all the angle range.

Figure 3-2 shows the polar plot of neuron 72. The direction of the vector on the polar plot
indicates the direction of velocities, and the magnitude of the vector is the average firing rate,
marked as a blue circle, for each direction. The computed circular mean, estimated as the firing
rate weighted direction, is shown as a solid red line on the polar plot. This indicates clearly that
neuron 72 fired most frequently toward the preferred direction.

**Metric for Tuning**

A metric is necessary to evaluate the neural tuning. A comparative analysis between the
neural firing and the kinematics based on the metric could provide a better understanding of the
neuron receptive field properties. A metric would also present a way to select the tuning neuron
subset that contributes most to movement generation, potentially reducing the decoding complexity. In this section, we review the previous tuning metric and then compare it to our newly proposed tuning metric.

**Tuning Depth**

The metric for evaluating the tuning property of a cell is the tuning depth of the cell’s tuning curve. This quantity is defined as the difference between the maximum and minimum values in the cellular tuning normalized by the standard deviation of the firing rate [Carmena et al., 2003; Sanchez, 2004]. The tuning depth is normalized between 0 and 1 through all the channels, which looses the scale for comparisons among different neurons.

\[
\text{tuning depth} = \frac{\text{max}(r_N) - \text{min}(r_N)}{\text{std}(\text{firing rate})}
\]  

(3-2)

The normalization in Equation 3-2 used to equalize the firing of different cells can wrongly evaluate a shallow tuned neuron as a deeply tuned neuron, when both fire with a small variance. The normalization inaccurately exaggerates the tuning depth when the standard deviation is close to 0. In fact, the tuning metric to evaluate neuronal participation in kinematics should depend not only upon the mean firing rate in a certain direction, but also on the distribution of the neural spike patterns. Normalizing by the firing rate alone may not be the best way to evaluate neuron tuning.

A counterexample using tuning depth as the metric is shown in Figure 3-3. Neuron 72 is plotted on the left, and neuron 80 is on the right. Neuron 72 fires less in other directions than the preferred one, while neuron 80 not fire at most of the directions expected for the preferred one. By visually inspecting the plots, we can infer that neuron 80 is more “tuned” than neuron 72. However, by tuning depth metric, neuron 80 was assigned smaller tuning depth, 0.93, than neuron 72’s tuning depth of 1. This may be due to normalization by the standard derivation of
the firing rate, which inaccurately exaggerates the tuning depth for some neurons with stable activities (standard derivation is close to 0). In fact, the tuning metric to evaluate differences between neuron reactions to kinematics depends not only upon the mean firing rate on a certain direction, but also on the distribution of the neural spike patterns. Normalizing by only the firing rate does not appear to be a very cogent or effective way to evaluate neuron tuning.

**Information Theoretic Tuning Metric**

The traditional tuning curves do not intrinsically allow us to measure information content. We have used indirect observational methods such as tuning depth but they are not optimal. An information theoretic tuning depth as a metric for evaluating neuron instantaneous receptive properties is based on information theory and would capture much more of the neuronal response [Paninski et al., 2004b; Wang et al., 2007b]. Define a tuned cell as a cell that extracts more information between the stimulus direction angle and its spiking output. If a cell is tuned to a certain angle, the well-established concept of mutual information [Reza, 1994] can mathematically account for an information theoretic metric between the neural spikes and direction angles, which is given by

\[
I(spk; \theta) = \sum_\theta p(\theta) \sum_{spk=0,1} p(spk | \theta) \log_2 \left( \frac{p(spk | \theta)}{p(spk)} \right)
\]

\[
= \sum_{spk=0,1} p(spk) \log_2 (p(spk)) - \sum_\theta p(\theta) \sum_{spk=0,1} p(spk | \theta) \log_2 (p(spk | \theta))
\]

(3-3a)

where \( p(\theta) \) is the probabilistic density of all the direction angles, which can be easily estimated by Parzen window [Parzen 1962]. The direction angles of the kinematic vectors are evaluated between \(-\pi\) and \(\pi\). \( p(spk) \) can be calculated simply as the percentage of the spike count during the entire spike train. \( p(spk | \theta) \) is the conditional probability density of the spike given the direction angle.
For each neuron, the conditional probability density \( p(spk \mid \theta) \) was estimated directed from the data by an intuitive nonparametric technique [Chichilnisky 2001, Simoncelli et al. 2004], as the fraction of the two kernel smoothed histograms of marginal \( p(\theta) \) and joint distribution \( p(spk = 1, \theta) \). The histogram of the spike-triggered angle is smoothed by a Gaussian kernel according to Silverman’s rule [Silverman, 1981] and normalized to approximate the joint probability \( p(spk = 1, \theta) \), depicted as the solid red line in upper plot of Figure 3-4. In other words, the direction angle is accounted for in the histogram during the corresponding direction angle bin only when there is a spike. Then the conditional probability density \( p(spk = 1 \mid \theta) \), depicted as solid blue line in the bottom plot of Figure 1, is approximated by dividing the kernel-smoothed histogram of \( p(spk = 1, \theta) \) by the kernel-smoothed histogram of \( \theta \) (blue dot line in the upper plot of Figure 3-4), which is in fact Bayesian rule,

\[
p(spk = 1 \mid \theta) = \frac{p(spk = 1, \theta)}{p(\theta)} \tag{3-4}
\]

where \( p(spk = 0 \mid \theta) = 1 - p(spk = 1 \mid \theta) \). When \( p(\theta) \) is 0, \( p(spk = 1, \theta) \) is set to be 0. Note that because \( p(spk, \theta) \) is always not greater than \( p(\theta) \), this actually does not share the same problem as Equation 3-2.

The traditional computation of \( r_N \) in the tuning depth, which is the average firing rate for certain angle \( \theta_N \), is actually a rough approximation of Equation 3-3 because

\[
r(\theta) = \frac{\sum_{i=1}^{M(\theta)} \lambda_i(\theta)}{M(\theta)} = \frac{\#_{\text{spike}}(\theta)}{\#(\theta)} \approx \frac{p(spk, \theta)}{p(\theta)} = p(spk \mid \theta) \tag{3-5}
\]

where \( M(\theta) \) is the total sample number at angle \( \theta \) from the whole data set, as well as \( \#(\theta) \), and \( \lambda_i(\theta) \) is the firing rate corresponding to sample \( i \) of angle \( \theta \). \( \#_{\text{spike}}(\theta) \) is the total number of
spike counts when the movement angle is $\theta$. The conditional probability density $p(spk \mid \theta)$ can be regarded as the non-linear functional relationship between instantaneous neuron firing probability and movement directions. We can see that the traditional tuning depth analysis actually works only with the difference between the maximum and minimum of the nonlinear tuning curve, scaled by the binning window. During the experiment the monkey very likely will not explore all the possible angles equally so it will achieve different prior distributions for $p(\theta)$.

The uniformly distributed $p(\theta)$ provides the ideal estimation for tuning curves. When there is insufficient data to estimate the accurate shape of $p(spk \mid \theta)$, the traditional tuning depth will certainly provide a bias. In the experiment, there is no guarantee of the data sufficiency. Its effect will be tested in synthetic data. The normalization by the standard deviation of the firing rate in Equation 3-2 brings the concern of binning window size as well. The information theoretical tuning depth works directly on the spike train. It takes into account not only the spike nature of the data, which we can tell from the first term in Equation 3-3b, but also the every point of the nonlinearity $p(spk \mid \theta)$ and the prior distribution $p(\theta)$ as well, which is shown in the second term in Equation 3-3b.

**Simulated Neural Recordings**

We first test our information theoretical criterion on synthetic data using a single random realization of the spike train. Three sets of 2-dimensional movement kinematics are generated. The magnitude and the direction of first dataset are both uniformly distributed within the range $[0, 1]$, $[-\pi, \pi]$ respectively. The second dataset has magnitude uniformly distributed while the direction is Gaussian distributed, centered at $2/3 \pi$ with standard deviation $0.1 \pi$. The third data set has Gaussian distributed magnitude centered at $0.7$ with standard deviation $0.1$, and Gaussian
distributed direction centered at $2/3 \pi$ with standard deviation $0.1 \pi$. The velocity train is passed through a LNP model with the assumed nonlinear tuning function in Equation 3-6.

$$\lambda_i = \exp(\mu + \beta \vec{v}_i \cdot \vec{D}_{\text{prefer}})$$  

(3-6)

where $\lambda_i$ is the instantaneous firing probability, $\mu$ is the background firing rate, $\beta$ represents the modulation factor to a certain preferred direction, which is represented by a unit vector $\vec{D}_{\text{prefer}}$. The spike train is generated by an inhomogeneous Poisson spike generator, once we have the knowledge of $\lambda_i$.

We generate each velocity dataset with 100 Hz sampling frequency and 100 sec duration (10000 samples totally) or 10 sec duration (1000 samples totally) to test the reliability of the tuning criterion when there is fewer data. The background-firing rate $\mu$ is set to 0. The preferred direction is set as $1/3 \pi$. We implemented 10 synthetic neurons distinguished by their modulation factor $\beta$ varying from 1 to 10, which hints at a monotonically increasing tuning. The first uniformly distributed data set is supposed to give full perspective of the tuning curve, since it explores all possible direction angles. The Gaussian distributed direction in the second data set favors samples at a certain direction. It won’t change the information about the tuning curves in terms of direction angle when compared to the first dataset. The third data set have the Gaussian distribution magnitude with center at 0.7, which means for given direction angle the instantaneous firing probability is higher than the uniformly distributed magnitude with mean at 0.5. Since randomness is involved in the generation of the velocity and spike trains, we will evaluate the tuning depth criterion for 100 Monte Carlo trials.

Figure 3-5 shows the average tuning information with standard deviation across 100 Monte Carlo trials evaluated for 10 neurons with 100 sec duration. The dotted line group is the tuning
information estimated by traditional tuning depth for all 3 datasets. In order to get the statistical evaluation between Monte Carlo runs, the traditional tuning depth were not normalized to [0, 1] for each realization as normally done in real data. The solid dot group is the tuning information estimated by information theoretical analysis for all 3 datasets. Both groups show higher information amount evaluated for each neuron from dataset 3 than the other 2 datasets as expected. However, the 2 lines evaluated from dataset 1 and dataset 2 are grouped much closer, which means less bias affected by prior distribution by information theoretical analysis than by traditional tuning depth. Since the more samples on the certain direction angle should not affect the information amount, the information theoretical analysis provides the estimation that makes more sense.

The tuning criterion is expected to steadily represent the tuning information amount across different Monte Carlo trials. However, for each neuron directly comparing the standard deviation through Monte Carlo trials between 2 methods is not fair, since their scales are quite different. We use correlation coefficient to measure the similarity of the tuning information curve along 10 neurons between each trial and the average performance. The statistical similarly results through 100 trials for 3 datasets evaluated by 2 methods with both durations are shown in Table 3-2. For each data set, pair-wise student t-test was performed to see if the results are statistically different from the traditional tuning depth. The test is performed against the alternative specified by the left tail test \( t_{\text{Tuning depth}} < t_{\text{Information theoretical Analysis}} \). All the tests are performed on the null hypothesis at \( \alpha = 0.01 \) significance level. Under the null hypothesis, the probability of observing a value as extreme or more extreme of the test statistic, as indicated by the \( p \)-value, is also shown in Table 3-2.
For each dataset, the tuning information criterion by information theoretical analysis shows steadily information representation with higher correlation and less standard deviation in terms of similarity to average tuning information. All the $t$-test confirms the statistical performance improvement. In the real data analysis, there is no guarantee that we always have sufficient data to estimate the tuning abilities. Note that with less duration (1000 samples), the information theoretical criterion performs better than the traditional one.

To distinguish the 10 neurons, we expect the criterion will be able to accurately rank the neurons monotonically related to the modulation factor $\beta$ from 1 to 10, even for the single realization of the spike train. Throughout the 100 Monte Carlo trials, the monotonicity of the tuning depth along 10 neurons for 3 datasets by both methods for the two durations is shown in Table 3-3. For example, among 100 Monte Carlo trials on 1000 sample simulation, only 7 trials show monotonicity by traditional tuning depth, while 62 trials show monotonicity by information theoretical analysis.

Note that the traditional tuning depth shows much poorer monotonicity for all the dataset compared to information theoretical analysis. It even fails on monotonicity test in dataset 3. This is because the normalization term in the traditional tuning depth (the standard deviation of the firing rate) is exponentially increasing when both modulation factor $\beta$ and mean of speed increases. When there is enough data, all the datasets show 100% monotonicity of tuning information across the 10 neurons evaluated by the information theoretical analysis. Even with insufficient data, the information theoretic tuning again shows much greater monotonicity percentage than the traditional tuning depth. Thus the information theoretical tuning depth is more reliable to rank neurons.
In Vivo Neural Recordings

Since we have tested the reliability of the information theoretical analysis on the tuning information in synthetic data, we now implement this criterion for our BMI data, where the neural activity is processed as binary spike trains sampled at 100 Hz. All the kinematics variables, hand position, velocity and acceleration, are upsampled to be synchronized with the neuron spikes trains. The traditional tuning depth for all 185 neurons is computed from each of the kinematic variables and normalized into [0,1] as shown in Figure 3-6. The top plot is tuning depth computed from position, the middle from velocity, and the bottom from acceleration. The cortical areas where the micro-arrays were placed are also marked in the Figure. We can see clearly that most tuned neurons are in the primary motor cortex regardless of which kinematic vectors are used to calculate the tuning depth.

Figure 3-7A shows the information theoretic depth calculated from all 3 kinematic directions for all the neurons. Compared to Figure 3-6, in which the tuning depths are usually normalized to [0, 1] for all kinematics, the mutual information shows clearly that the velocities (the middle plot) relatively conveys more tuning information than position or acceleration, as reported in the literature [Paninski et al., 2004a] Since mutual information is a distance (it is self-normalized) it allows the relative assessment of tuning across different kinematics. For example, we found that neuron 121 is tuned more to position, while neuron 149 is tuned more to velocity. In Figure 3-7A, with the exception of the M1 cortical area, the neuronal information theoretic tuning depths seem almost flat, which could be erroneously interpreted as meaning that these neurons have little or no tuning. Actually, the mutual information is a nonlinear measure, emphasizing the large distances. Due to the large dynamic range of the mutual information, it is preferable to display the results in logarithmic scale. The difference between neurons in other cortical area is much clearly depicted in Fig 3-7B.
Information Theoretical Neural Encoding

This section implements an information theoretical methodology to address *instantaneous* neuronal encoding properties. The analysis is based on a statistical procedure for quantifying how neuronal spike trains directly encode arm kinematics. All of the evaluation is performed directly with the neural spike times, which preserves the fine time structure of the representation without determining a rate code and its associated window size commonly chosen by the experimenter.

**Instantaneous Tuning Function in Motor Cortex**

The literature contains many different types of tuning functions (*i.e.*, linear, exponential, Gaussian) [Moran & Schwartz, 1999; Eden & Frank *et al.*, 2004]. These nonlinear mathematical models are not optimal for dealing with the real data because each neuron very likely has different tuning properties [Wise *et al.*, 1998]. The accuracy of the tuning function estimation will directly affect the Bayesian decoding approach and, therefore, the results of the kinematic estimation in BMIs.

The spike-triggered average (STA) is one of the most commonly used white noise analysis [deBoer & Kuyper, 1968; Marmarelis & Naka, 1972; Chichilnisky, 2001], applicable when data is uncorrelated. It is applied for instance in the study of auditory neurons [Eggermont *et al.*, 1983], retinal ganglion cells [Sakai & Naka, 1987; Meister *et al.*, 1994], lateral geniculate neurons [Reid & Alonso, 1995], simple cells in primary visual cortex (V1) [Jones & Palmer, 1987; McLean & Palmer, 1989; DeAngelis *et al.*, 1993]. STA provides an estimate of the first linear term in a polynomial series expansion of the system response function with the assumptions that the raw stimulus distribution is spherically symmetric or elliptically symmetric (whitening operation is then necessary), and the raw stimuli and the spike-triggered stimuli distribute differently in terms of the mean. If the system is truly linear, STA provides a complete
characterization. This linear approximation was improved by Simoncelli, Paninski and colleagues [Simoncelli et al., 2004]. By parametric model identification, the nonlinear property between the neural spikes and the stimuli was directly estimated from data, which is more reliable than just assuming linear or Gaussian dependence. In our sequential estimation for BMI studies [Wang et al., 2007b] it provides a very practical way to acquire the prior knowledge (the tuning function) for decoding purposes.

This technique estimates the tuning function by a Linear-Nonlinear-Poisson (LNP) model [Simoncelli et al., 2004], which is composed of a linear filter followed by a static nonlinearity then followed by a Poisson model, as shown in Figure 3-8.

The linear filter projects the multi-dimensional kinematic vector into its weight vector \( \vec{k} \) (representing a direction in space), which produces a scalar value that is converted by a nonlinear function \( f \) and applied to the Poisson spike-generating model as the instantaneous conditional firing probability \( p(\text{spike} | \vec{k} \cdot \vec{x}) \) for that particular direction in the high dimensional space. In our work the optimal linear filter actually projects the multi-dimensional kinematic vector \( \vec{x} \) built from the position, velocity and acceleration in \( x \) and \( y \) along the direction where they differ the most from the spike triggered kinematic vectors. This projection could represent the transformation between kinematics to muscle activation [Todorov, 2000]. The nonlinear function \( f \) represents the neuron nonlinear response, which accounts for all of the processing of the spinal cord and deep brain structures to condition the signal for activation operations [Todorov, 2000]. The Poisson model, which encodes the randomness of neural behavior, generates spike trains with an instantaneous firing probability defined by the nonlinear output. This modeling method assumes that the generation of spikes depends only on the recent stimulus and is historically independent of previous spike times.
Previous work [Paninski et al., 2004a, Paninski et al., 2004b] utilized a window in time approach to build a smoother statistical tuning function from temporal kinematics to instantaneous neural firing rate. In the encoding stage, the kinematic variable within a window that embeds temporal information before and after the current neuron firing time is used as a high dimensional input vector. The linear-nonlinear stage of the LNP model generates a one-dimensional output as the estimated firing rate for the Poisson stage. However, the sequential estimation model of our BMI requires just the opposite (i.e., we need to predict from the current neural activity event a sequence of kinematics), especially for the neurons in M1. When we infer the kinematics during a certain window length with respect to a particular spike, the state estimation error can accumulate easily as the estimation is recursively propagated into the next time iteration to build the vector during the window. Thus, a one-to-one mapping between the instantaneous kinematics and the neural activities is of paramount importance for the online decoding purpose. The other issue is to estimate appropriately the optimal delay in the instantaneous functional mapping. Due to the decrease in the amounts of data, the instantaneous decoding is expected to be noisier (fewer data to identify the transfer function), but there are also possible advantages. Compared to the windowed method of Paninski et al. [2004b], instantaneous estimation works directly in the dynamic range of the kinematic signals instead of being affected by all the temporal information embedded within the window. To deal with the sensitivity issue for neural tuning identification, the method works with the full kinematic vector containing the instantaneous position, velocity and acceleration to include the information that each kinematic variable conveys for tuning, which ultimately is what is needed in BMI decoding.

Estimation of the instantaneous encoding depends upon the ability to estimate the appropriate time delay between motor cortical neuron activity and kinematics [Wu et al., 2006].
Due to the propagation effects of signals in the motor and peripheral nervous system and to preserve causality, the intended movement is executed after the motor cortical neuron fires (Figure 3-9).

In the temporal kinematic encoding by LNP models, a window that usually samples 300 msec before and 500 msec after the current neural firing rate [Paninski et al., 2004b] is used to construct the high dimensional kinematic vector. Although the causal time delay is already taken into account, the temporal kinematic information before the neuron fires actually has no causal relation with respect to the current spike. For the instantaneous kinematic encoding model, the optimum time delay has to be estimated to draw as much information as possible. The instantaneous motor cortical neural activity can be modelled as

\[
\lambda_i = f(\vec{k} \cdot \vec{x}_{t+\text{lag}})
\]  \hspace{1cm} (3-7)

\[
\text{spike}_i = \text{Poisson}(\lambda_i)
\]  \hspace{1cm} (3-8)

where \(\vec{x}_{t+\text{lag}}\) is the instantaneous kinematics vector defined as \([p_x \ v_x \ \ddot{a}_x \ p_y \ v_y \ \ddot{a}_y]^{T}\) with 2-dimensional information of position, velocity and acceleration with causal time delay. \(\vec{k}\) is a linear filter, representing the preferred instantaneous direction in high-dimensional kinematics space. The weight estimation of the linear filter is based on the standard technique of spike-triggered regression.

\[
\vec{k} = (E[\vec{x}_{t+\text{lag}}^{T} \vec{x}_{t+\text{lag}}] + \sigma)^{-1} E_{x_{t+\text{lag}}|\text{spike}_i}[\vec{x}_{t+\text{lag}}]
\]  \hspace{1cm} (3-9)

Equation 3-9 represents the least square solution for the linear adaptive filter, where

\(E[\vec{x}_{t+\text{lag}}^{T} \vec{x}_{t+\text{lag}}]\) gives the autocorrelation matrix \(R\) of the input vector considering causal time delay. \(\sigma\) is a regularization factor, which avoids ill-conditioning in the inverse. In the experiment, \(\sigma\) is chosen to maximize the linear filter performance. From a statistical
perspective, \( E_{\tilde{x}_{t+\text{lag}|\text{spike}}} \tilde{x}_{t+\text{lag}} \) mimics the role of the cross-correlation vector \( P \) between the input and the binary spike train considering a causal time delay. Therefore, Equation 3-9 reduces to a conditional expectation of the binary spike train (i.e., this linear filter gives the spike-triggered average instantaneous kinematic vector \( E_{\tilde{x}_{t+\text{lag}|\text{spike}}} \tilde{x}_{t+\text{lag}} \)) scaled by the decorrelated kinematic data \( (E[\tilde{x}_{t+\text{lag}}^T \tilde{x}_{t+\text{lag}}] + \sigma)^{-1} \).

\( \lambda \) is the instantaneous firing rate in an inhomogeneous Poisson spike generator. For the time interval selected for the spike analysis (i.e. the time interval valid for a Poisson assumption in the collected data, which has to be experimentally determined), a number is randomly drawn from a normalized uniform distribution (i.e., 0 to 1) and compared with the instantaneous conditional firing probability. If the number is smaller than the probability, then a spike is generated in this time interval. This modelling approach is therefore intrinsically stochastic, which carries implications (large variance) to on-line real time implementations.

\( f \) is the nonlinear function estimated by an intuitive nonparametric technique [Chichilnisky 2001; Simoncelli et al., 2004] as the conditional probability density \( p(spk | \tilde{k} \cdot \tilde{x}) \) directed from the data. It is the fraction of the two kernel smoothed histograms of marginal \( p(\tilde{k} \cdot \tilde{x}) \) and joint distribution \( p(spk, \tilde{k} \cdot \tilde{x}) \). It is the same way when we describe in Figure 3-4. The only difference is that we are plot the joint and marginal pdf in term of filtered kinematics \( \tilde{k} \cdot \tilde{x} \). The histogram of the spike-triggered angle is smoothed by a Gaussian kernel according to Silverman’s rule [Silverman, 1981] and normalized to approximate the joint probability \( p(spk, \tilde{k} \cdot \tilde{x}) \), depicted as the solid red line in upper plot of Figure 3-10. In other words, the direction angle is accounted for in the histogram during the corresponding direction
angle bin only when there is a spike. Then the conditional probability density \( p(spk \mid \vec{k} \cdot \vec{x}) \), depicted as the line in the bottom plot of Figure 3-10, is obtained by dividing the kernel-smoothed histogram of \( p(spk, \vec{k} \cdot \vec{x}) \) by the kernel-smoothed histogram of \( \vec{k} \cdot \vec{x} \) (dotted line in the upper plot of Figure 3-10), which in fact implements Bayes rule,

\[
p(spk = 1 \mid \vec{k} \cdot \vec{x}) = \frac{p(spk = 1, \vec{k} \cdot \vec{x})}{p(\vec{k} \cdot \vec{x})}
\]

(3-10)

where \( p(spk = 0 \mid \vec{k} \cdot \vec{x}) = 1 - p(spk = 1 \mid \vec{k} \cdot \vec{x}) \). When \( p(\vec{k} \cdot \vec{x}) \) is 0, \( p(spk = 1, \vec{k} \cdot \vec{x}) \) is set to be 0.

The peak in the conditional probability of Figure 3-10 is associated with the maximal firing probability, which is linked with specific values of the kinematic variables, and produces an increase in the firing rate of the neuron. Likewise, the region of low probability shows a deviation from the spontaneous firing rate for the neuron. These two portions of the curve (the most difficult to estimate well because they are at the tails of the distribution) are responsible for the modulation that is seen in the rasters of the spike train data when observed along with the kinematic variables, and that are fundamental for BMI decoding performance.

**Information Theoretic Delay Estimation**

The causal time delay can also be estimated by information theoretical analysis. Here, we are interested in the optimum time lag, which extracts the most instantaneous kinematic information corresponding to the neural spike event. The well-established concept of mutual information [Reza, 1994] as a metric for evaluating neuron instantaneous receptive properties is based on information theory and would capture much more of the neuronal response [Paninski et al., 2004b; Wang et al., 2007b]. Define a tuned cell as a cell that extracts more information between the linear filtered kinematics and its spiking output. If a neuron is tuned to a preferred direction in high-dimensional space, the mutual information between the spike and the delayed
linear filter kinematics vector is first drawn simply as the function of the time lag after a spike as in Equation 3-11.

$$I_{(\text{spk}; \tilde{k}, \tilde{x})}(\text{lag}) = \sum_{k} p(\tilde{k} \cdot \tilde{x}(\text{lag})) \sum_{\text{spike}=0,1} p(\text{spk} | \tilde{k} \cdot \tilde{x}(\text{lag})) \cdot \log_2 \left( \frac{p(\text{spk} | \tilde{k} \cdot \tilde{x}(\text{lag}))}{p(\text{spk})} \right)$$

(3-11)

where $p(\tilde{k} \cdot \tilde{x}(\text{lag}))$ is the probabilistic density the linear filtered kinematics as a function of time lag, which can be easily estimated by Parzen window [Parzen, 1962]. $p(\text{spk})$ can be calculated simply as the percentage of the spike count during the entire spike train. $p(\text{spk} | \tilde{k} \cdot \tilde{x})$ is exactly the nonlinear function $f$ in LNP model.

The time delay with the highest mutual information is assigned as the optimum time lag for each neuron. The kinematics at the optimum time lag carries maximally the causal information of the neural spike. In the encoding stage, the 6-dimentional kinematic vectors are first synchronized at the optimum delay for each neuron, then input to the LNP tuning model to generate the estimated firing rates according to Equation 3-7. To test the encoding ability of the instantaneous tuning model, the neuron firing rate is obtained by smoothing the real spike train with a Gaussian kernel. The correlation coefficient is then calculated between two firing rates to measure the quality of encoding.

As we mentioned in the previous section, the windowed kinematic vector is usually chosen as 300 msec before and 500 msec after the current neural spike, which already takes into account the causal time delay of the motor cortical neurons. We selected a possible delay range from 0 to 500ms after a neuron spikes to estimate the optimum time delay for our instantaneous tuning function. The regularization factor $\sigma^2$ in the spike-triggered average stage is experimentally set as $10^{-7}$, and the kernel size to smooth the histogram of probability density is set according to Silverman’s rule [Silverman, 1981]. For all 185 neurons, the mutual information as a function of
time delay was obtained from 10,000 continuous samples (100 seconds) during movement. The
time delay with highest mutual information was assigned as the best time lag for each neuron.
Since neurons in M1 show more tuning information than other cortical areas, here we study 5
neurons that show the highest tuning, neurons 72, 77, 80, 99, and neuron 108. Figure 3-11 shows
the mutual information as a function of time delay after spike occurrence. The best time lags are
marked by a cross on each curve, and are 110ms, 170 ms, 170ms, 130 ms and 250ms
respectively. It is interesting to observe that not all the neurons have the same time delay,
although all of these neurons are in M1. During the analysis, different time delay is used for each
neuron respectively. The average best time delay for all 185 neurons was 220.108ms, which is
close to the results mentioned in the literature [Wu et al., 2006].

**Instantaneous vs. Windowed Tuning Curves**

The windowed encoding approach yields a widely accepted exponential increasing
nonlinear function f after linear projection [Paninski et al., 2004a, Paninski et al., 2004b].
However for BMIs we are proposing an instantaneous and global (i.e., across kinematic
variables) tuning estimation, therefore it is important to compare and evaluate the two tuning
methodologies. For each neuron, we chose 7 different window sizes to filter the kinematic vector
\[ \begin{bmatrix} \ddot{p}_x & \ddot{v}_x & \ddot{a}_x & \ddot{p}_y & \ddot{v}_y & \ddot{a}_y \end{bmatrix}_{t+\text{lag}} \] and calculate the nonlinearity using the methods described in Figure 3-10. The biggest window size is 300 ms before and 500 ms after the current neural spike, noted as
[-300, 500], which has been used in [Paninski et al., 2004b] for motor cortical neuron tuning
analysis. Then each window shrinks 50ms at left and right extremes, such as [-250, 450], [-200,
400], ... until the smallest window [0, 200] ms. Figure 3-12 shows the nonlinearity of the 4 MI
neurons estimated by windowed kinematics with 7 different window sizes, each plotted in
different colors. The instantaneous nonlinear tuning with optimum delay is emphasized in a thick red line.

As we can observe from the figures, the tuning curves vary with different window sizes, particularly in the high tuning region. However, the middle part of the nonlinearity is very stable across all the window sizes, including the instantaneous estimation. Compared to the windowed tuning, the instantaneous model produces a smaller dynamic range of projected values (x-axis) because it directly works in the dynamics range of the kinematics without involving time-embedded information. We chose the Correlation Coefficient (CC) as the criterion to evaluate the similarity between the nonlinear tuning curves estimated from each windowed kinematics and the instantaneous one within the range specified by the instantaneous model. Seven histograms of correlation coefficients are shown in Figure 3-13, where the y-axis shows the percentage of neurons (out of 185) with a given CC. We can see that 98.92% of neurons have instantaneous tuning curves with a similarity over 0.9 compared to the one by window size [-300, 500] ms. More than half (58.38%) of the neurons have a similarity over 0.9 for the [-50, 250] ms window. However, less than half (41.62%) of the neurons have a similarity over 0.9 for the [0, 200] ms window because this window is not big enough to include the optimum causal delay, which is on average 220 ms. Since the summation for the same window size (color bar) is 100%, the similarity of the less similar neurons (CC<0.9) is distributed across other CC bins.

Also notice that from the windowed methods the one with the smallest window, when it includes the optimum time delay, is the closest to the instantaneous estimated tuning. The similarity amongst windowed and instantaneous methods is rather surprising, and builds confidence that in spite of its simplicity in computation it is quantifying appropriately neural tuning properties.
One possible reason for the differences at both extremes of the tuning curves is insufficient data to provide an accurate estimation at both extremes, in particular because of the division in Equation 3-10. Recall that this is actually the important part of the tuning curve for BMIs because it is in this portion that the neuron firing shows modulation with the kinematic variable. In particular, neurons 80 and 99 (as many others) show a large mismatch at the high firing rate level (right end of the curve). Both neurons demonstrate a lower firing probability in the instantaneous curve compared to the windowed curves. Neuron 80 also shows a saddle-like behavior very different from the exponential increase. Therefore these behaviors need to be further investigated.

**Instantaneous vs. Windowed Encoding**

Since the ultimate goal of the tuning analysis is to transform spike timing information into the kinematic space, here we compare both tuning methods in our experimental data set. Here we select neuron 80 and neuron 99 to compare the encoding ability between the windowed and the instantaneous tuning model with the real kinematics signals (Figure 3-14A and 3-14B). From previous studies, these 2 neurons are known to be among the most sensitive neurons for BMI modeling [Sanchez *et al.*, 2003; Wang *et al.*, 2007b], and they are also amongst the ones that show the larger mismatch at the high firing probability range (right extreme end of Figure 3-12).

In each plot, the pink bars in the first and second rows represent the neural spike train. The red dash line superimposed on the spike train is the firing rate estimation by kernel smoothing. In the top panel, the blue solid line superimposed on the spike train is the estimated firing rate by instantaneous tuning, while in the second panel, the green solid line superimposed on the spike train is the estimated firing rate by windowed tuning with 300 ms before and 500 ms after the current neuron firing. To check the animal’s behavior simultaneously with the neural activity, the
third and fourth panels show the re-scaled 2D position and velocity (blue for $x$, green for $y$) after synchronization at the optimum delay.

We can clearly observe that, for both neurons (Figure 3-14A and Figure 3-14B), the instantaneous model gives a smoother estimated firing rate than the noisy estimation by the windowed model. We found that the linear filter outputs in the windowed model are very noisy, because it is a projection of the high dimensional time-embedded kinematic vector, which increases the range of the independent variable and so creates larger variability in the spike rates. Moreover, the over-estimation at high firing rate of the nonlinearity curve leads to the extraneous large peaks on the green line. As can be expected, since the tuning is higher, there will be more spikes and so the intensity function estimation is very high and noisier as seen in the green curve.

It is also very interesting to notice that after causal alignment both neurons demonstrate clear time alignment (negative correlation) between the hand velocity trajectory and the peaks of firings, which reinforces the evidence for neural kinematic encoding.

To quantify the encoding comparisons, the correlation coefficient between the neuron’s firing rate and the estimated firing rates from the windowed and instantaneous models. The kernel size smooths the spike train to enable the estimation of CC but it will affect the results of the similarity measure. Figure 3-15A and B shows results comparing the CC for the same 2 neurons vs. different kernel sizes. Correlation coefficients for the instantaneous model are always greater than the ones by windowed model across kernel sizes. Here we choose to display the kernel size that maximizes the similarity. For neuron 99, the correlation coefficient between the instantaneous model and the firing rate is 0.6049, which is greater than 0.4964 for the windowed model. For neuron 80, the correlation coefficient between the estimated firing rate with the instantaneous model and the firing rate from real spike train is 0.6393, which is greater than
0.5841 given by the windowed model. Therefore, the instantaneous model shows better encoding ability.

**Discussion**

The traditional criterion of estimating tuning depth from windows of data does not seem the most appropriate in the design of BMIs using sequential estimation algorithms on spike trains. Here we present instead an information theoretical tuning analysis of instantaneous neural encoding properties that relate the instantaneous value of the kinematic vector to neural spiking. The proposed methodology is still based on the LNP model, and an information theoretic formulation provides a more detailed perspective when compared with the conventional tuning curve because it statistically quantifies the amount of information between the kinematic vectors triggered off by the spike train. As a direct consequence, it can estimate the optimum time delay between motor cortex neurons and behavior caused by the propagation effects of signals in the motor and peripheral nervous system.

The similarities and differences between the windowed and instantaneously evaluated tuning functions were also analyzed. We conclude that the instantaneous tuning curves for most of the neurons show over 0.9 correlation coefficients in the central region of the tuning curve, which unfortunately is not the most important for BMI studies. There are marked differences in the high tuning region of the curves, both in the dynamic range and in the estimated value. The windowed model works on a time-embedded vector, which spreads the linear output \(\vec{k} \cdot \vec{x}\) to a wider range. Since the pdf integral is always 1, the windowed model flattens the marginal distribution of \(p(\vec{k} \cdot \vec{x})\). In the time segment when the neuron keeps firing, the overlapping windows make the linear filter output \(\vec{k} \cdot \vec{x}\) change slowly. It results in more spike-triggered samples in the small neighborhood of \(\vec{k} \cdot \vec{x}\). Therefore, the estimation on the joint distribution
\( p(spk, \vec{k} \cdot \vec{x}) \) becomes higher. Both consequences contribute to the overestimation of tuning at the high firing rate part of the windowed nonlinear curve.

The instantaneous model works directly in the dynamic range of the kinematics that is sensitive only to the corresponding neuron spike timings. It estimates more accurately the firing probability without distortions from the temporal neighborhood information. However, we create a vector with all of the kinematics (position, velocity, acceleration) to estimate better (i.e., to obtain more sensitivity) the tuning from the data. This has the potential to mix tuning information for the different kinematics variables and different directions if they are not exactly the same. When the different kinematic variables display different sensitivities in the input space, after projection by the weight filter direction they will peak at different values of \( \vec{k} \cdot \vec{x} \) in the nonlinear curve, which then results in the saddle-like feature observed in Figure 3-12. The other potential shortcoming is that less data is used, so the variability may be higher. However, at this time one still does not know which tuning curve provides a better estimate for the instantaneous tuning model required in the encoding and decoding stages of BMIs. Ultimately, the instantaneous model can produce equivalent or better encoding results compared to existing techniques. This outcome builds confidence to directly implement the instantaneous tuning function into the future online decoding work for Brain-Machine Interfaces.
Table 3-1. Assignment of the sorted neural activity to the electrodes

<table>
<thead>
<tr>
<th>Aurora (left handed)</th>
<th>Right PMA</th>
<th>Right MI</th>
<th>Right S1</th>
<th>Right SMA</th>
<th>Left MI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-66(66)</td>
<td>67-123(57)</td>
<td>123-161(38)</td>
<td>162-180(19)</td>
<td>181-185(5)</td>
</tr>
</tbody>
</table>

Figure 3-1. The BMI experiments of 2D target reaching task. The monkey moves a cursor (yellow circle) to a randomly placed target (green circle), and is rewarded if a cursor intersects the target.

Figure 3-2. Tuning plot for neuron 72
Figure 3-3. A counterexample of neuron tuning evaluated by tuning depth. The left plot is a tuning plot of neuron 72 with tuning depth 1. The right plot is for neuron 80 with tuning depth 0.93.

Figure 3-4. The conditional probability density estimation.
The average tuning information across Monte Carlo trials for different neurons

Figure 3-5. The average tuning information across trials by different evaluation

Table 3-2. The statistical similarity results comparison

<table>
<thead>
<tr>
<th>Sample# Method</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional tuning depth</td>
<td>0.9705 ± 0.0186</td>
<td>0.9775 ± 0.0133</td>
<td>0.9911 ± 0.058</td>
</tr>
<tr>
<td>$10^3$ Information theoretical analysis</td>
<td>0.9960 ± 0.0024</td>
<td>0.9964 ± 0.0021</td>
<td>0.9988 ± 0.0008</td>
</tr>
<tr>
<td>$t$-test ($p$ value)</td>
<td>1(9.52×10^{-26})</td>
<td>1(1.37×10^{-26})</td>
<td>1(5.68×10^{-24})</td>
</tr>
<tr>
<td>Traditional tuning depth</td>
<td>0.9976 ± 0.0013</td>
<td>0.9977 ± 0.0014</td>
<td>0.9991 ± 0.0005</td>
</tr>
<tr>
<td>$10^4$ Information theoretical analysis</td>
<td>0.9997 ± 0.0002</td>
<td>0.9996 ± 0.0002</td>
<td>0.9999 ± 0.0001</td>
</tr>
<tr>
<td>$t$-test ($p$ value)</td>
<td>1(1.60×10^{-26})</td>
<td>1(4.57×10^{-25})</td>
<td>1(6.00×10^{-19})</td>
</tr>
</tbody>
</table>

Table 3-3. The comparison of percentage of Monte Carlo results in monotonically increasing

<table>
<thead>
<tr>
<th>Sample# Method</th>
<th>Dataset 1</th>
<th>Dataset 2</th>
<th>Dataset 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^3$ Traditional tuning depth</td>
<td>7%</td>
<td>3%</td>
<td>0%</td>
</tr>
<tr>
<td>Information theoretical analysis</td>
<td>62%</td>
<td>57%</td>
<td>76%</td>
</tr>
<tr>
<td>$10^4$ Traditional tuning depth</td>
<td>76%</td>
<td>84%</td>
<td>0%</td>
</tr>
<tr>
<td>Information theoretical analysis</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Figure 3-6. Traditional tuning depth for all the neurons computed from three kinematics
Figure 3-7. Information theoretic tuning depth for all the neurons computed from 3 kinematics plotted individually. A) In regular sale. B) In logarithmic scale
Figure 3-8. Block diagram of Linear-Nonlinear-Poisson model

Figure 3-9. Sketch map of the time delay between neuron spike train (bottom plot) and the kinematics response (upper plot)
Figure 3-10. The conditional probability density estimation

Figure 3-11. Mutual information as function of time delay for 5 neurons
Figure 3-12. Continued
Figure 3-13. Correlation coefficient between the nonlinearity calculated from windowed kinematics and the instantaneous kinematics with optimum delay
Figure 3-14. Comparison of encoding results by instantaneous modeling and windowed modeling. A) Neuron 99. B) Neuron 80
Figure 3-15. Comparison of encoding similarity by instantaneous modeling and windowed modeling across kernel size. A) Neuron 99. B) Neuron 80
CHAPTER 4
BRAIN MACHINE INTERFACES DECODING IN SPIKE DOMAIN

The Monte Carlo Sequential Estimation Framework for BMI Decoding

We have thus far presented background on the difference between simulation and BMI real data, and have elaborated on the Monte Carlo sequential estimation algorithm. Based on this information, we now present a systematic framework for BMI decoding using a probabilistic approach.

The decoding of Brain Machine Interfaces is intended to infer the primate’s movement from the multi-channel neuron spike trains. The spike times from multiple neurons are the multi-channel point process observations. The kinematics is the state that needs to be derived from the point process observation through the tuning function by our Monte Carlo sequential estimation algorithm. Figure 4-1 provides a schematic of the basic process.

The decoding schematic for BMIs is shown in Figure 4-1 as the right to left arrow. The signal processing begins by first translating the neuron spike times collected from the real data into a sequence of 1 (there is a spike) and 0 (no spike). A time interval small enough should be chosen to guarantee the Poisson hypothesis (i.e., only few intervals have more than one spike). If the interval is too small, however, the computational complexity is increased without any significant improvement in performance. One must also be careful when selecting the kinematic state (position, velocity, or acceleration) for the decoding model since the actual neuron encoding is unknown. The analysis presented here will consider a vector state with all three kinematic variables. The velocity is estimated as the difference between the current and previous recoded positions, and the acceleration is estimated by first differences from the velocity. For fine timing resolution, all of the kinematics are interpolated and time synchronized with the neural spike trains.
It is interesting to note that in black box modeling, the motor BMI is posed as a decoding problem \( i.e. \), a transformation from motor neurons to behavior. However, when we use the Bayesian sequential estimation, decoding is insufficient to solve the modeling problem. In order to implement decoding it is important to also model how each neuron encodes movement, which is exactly the observation model \( f(\cdot) \) in tuning analysis in Chapter 3. Therefore, one sees that generative models do in fact require more information about the task and are therefore an opportunity to investigate further neural functionality. Here we use the instantaneous motor cortical neural activity modeled in Chapter 3 as

\[
\lambda_i = f(k \cdot \vec{x}_{t+\text{lag}}) \tag{4-1}
\]

\[
\text{spike}_i = \text{Poisson}(\lambda_i) \tag{4-2}
\]

where, as before, \( \vec{x}_{t+\text{lag}} \) is the instantaneous kinematics vector defined as

\[
[\vec{p}_x, \vec{v}_x, \vec{a}_x, \vec{p}_y, \vec{v}_y, \vec{a}_y, 1]^T_{t+\text{lag}} \text{ with 2 dimensional information of position, velocity, acceleration and bias with causal time delay depending on the data. For BMI, the kinematic vector in Linear-Nonlinear-Poisson model must be read from the experiment for every spike occurrence since the task is dynamic, taking into consideration the causal delay between neural firings and kinematic outputs [Wang et al., 2007b]. The linear filter projects the kinematics vector \( \vec{x} \) into its weight vector \( \vec{k} \) (representing a preferred direction in space), which produces a scalar value that is converted by a nonlinear function \( f \) and applied to the Poisson model as the instantaneous conditional firing probability \( \lambda_i \) for that particular direction in space \( p(\text{spike} | \vec{k} \cdot \vec{x}) \). The filter weights are obtained optimally by least squares as \( \vec{k} = (E[\vec{x}_{t+\text{lag}}^T \vec{x}_{t+\text{lag}}] + \alpha)^{-1} E_{\vec{x}_{t+\text{lag}}|\text{spike}_i}[\vec{x}_{t+\text{lag}}] \),

where \( E_{\vec{x}_{t+\text{lag}}|\text{spike}_i}[\vec{x}_{t+\text{lag}}] \) is the conditional expectation of the kinematic data given the spikes. The
parameter $\alpha$ is a regularization parameter to properly condition the inverse. The optimal linear filter actually projects the multi-dimensional kinematic vectors along the direction where they differ the most from the spike triggered kinematic vectors.

The nonlinear encoding function $f$ for each neuron was estimated using an intuitive nonparametric technique [Chichilnisky, 2001; Simoncelli et al., 2004]. Given the linear filter vector $\vec{k}$, we drew the histogram of all the kinematics vectors filtered by $\vec{k}$, and smoothed the histogram by convolving with a Gaussian kernel. The same procedure was repeated to draw the smoothed histogram for the outputs of the spike-triggered velocity vectors filtered by $\vec{k}$. The nonlinear function $f$, which gives the conditional instantaneous firing rate to the Poisson spike-generating model, was then estimated as the ratio of the two smoothed histograms. Since $f$ is estimated from real data by the nonparametric technique, it provides more accurate nonlinear properties than just assuming the exponential or Gaussian function. In practice, it can be implemented as a look up table, for its evaluation in testing as

$$p(\text{spike} | \vec{k} \cdot \vec{x}'_{\text{test}}) = \frac{\sum_{j} k(\vec{k} \cdot \vec{x}'_{\text{test}} - \vec{k} \cdot \vec{x}^{j}_{\text{spike,training}})}{\sum_{i} k(\vec{k} \cdot \vec{x}_{\text{test}} - \vec{k} \cdot \vec{x}^{i}_{\text{training}})}$$

(4-3)

where $k$ is the Gaussian kernel, $\vec{x}'_{\text{test}}$ is a possible sample we generate at time $t$ in the test data. $\vec{x}^{i}_{\text{training}}$ is one sample of velocity vector in the training data, and $\vec{x}^{j}_{\text{spike,training}}$ is corresponding spike-triggered sample. In our calculation, we approximate the nonlinearity for each neuron by a 2-layer MLP with a 10 hidden logsig PEs trained by Levenberg-Marquardt back-propagation.

The causal time delay is obtained by maximizing the mutual information as a function of time lag for each neuron from 10000 continuous samples of the kinematic variables [Wang et al., 2007b], as we described in Chapter 3. Here we further assume that the firing rates of all the
neuron channels are conditionally independent in implementing whole Monte Carlo sequential estimation (SE) algorithm with the encoding and decoding process on BMI.

First, the neuron activity data and kinematics are preprocessed. The only information we stored on the neural activities is the spiking time. In our preprocessing, we check every time interval with a spiking time, and assign 1 if there is a spike; otherwise, we assign 0. The interval should be small enough so that only a few intervals have more than one spike. In this case, we will still assign 1. The multi-channel spike trains are generated as our point process observations. We identify the kinematic variable as the state we are interested in reconstructing, or the one that carries the most information as determined by the information theoretical tuning depth. This variable could be a kinematic vector during a window, which contains both spatial and temporal information. It could also be an instantaneous kinematic variable resulting from a spike with some time delays specific to the motor cortex. The velocity is derived as the difference between the current and previous recoded positions, and the acceleration is derived the same way from the velocity. All the kinematics are interpolated to be synchronized with neural spike trains.

Secondly, the kinematics dynamic system model $F_k$, as stated in Equation 2-6 in Chapter 2, and the tuning function between the neural spike train and the primate’s kinematics are estimated from the existing (training) data. The system model is used to linearly predict the next kinematic value from the current one as $x_k = F_k x_{k-1} + \eta_k$. Since the kinematics are continuous values, $F_k$ can be estimated easily by the least square solution. The tuning function $\lambda_i = f(\vec{k} \cdot x_i)$ is designed as a linear-nonlinear-Poisson model for each neuron to describe the conditional firing rate as a function that encodes the kinematics that we are interested in reconstructing. The details of linear parameter $\vec{k}$ and nonlinear function $f$ estimation are already discussed in Chapter 3.
Provided the pre-knowledge of the system model and tuning function, we can implement the Monte Carlo sequential estimation adaptive filtering algorithm for point process. By generating a sequential set of samples, the posterior density \( p(\Delta N_k^j \mid x_k) \) is recursively estimated given spike train of neuron \( j \). At each time iteration \( k \), the joint posterior density \( p(\Delta N_k \mid x_k) \) is approximated by the product of all the marginal \( p(\Delta N_k^j \mid x_k) \), which assumes the conditional independence between neurons. The state is determined by the maximum a posterior or the expectation by collapsing the Gaussian kernel on the set of samples. The following steps represent the entire process.

**Step 1**: Preprocess and analysis.
1. Generate spike trains from stored spike times.
2. Synchronize all the kinematics with the spike trains.
3. Assign the kinematic vector \( \tilde{x} \) to be reconstructed.

**Step 2**: Model estimation (encoding).
1. Estimate the kinematic dynamics of the system model
   \[
   F_k = (E[\tilde{x}_{k-1}^T \tilde{x}_{k-1} + \alpha I])^{-1} E[\tilde{x}_{k-1} \tilde{x}_k]
   \]
2. For each neuron \( j \), estimate the tuning function
   1) Linear model \( \tilde{k}^j = (E[x^T_x] + \alpha I)^{-1} E_{x_{spike}}[\tilde{x}] \)
   2) Nonlinear function \( f^j(\tilde{k}^j, \tilde{x}) = \frac{p(\text{spike}^j, \tilde{k}^j \cdot \tilde{x})}{p(\tilde{k}^j \cdot \tilde{x})} \)
   3) Implement the inhomogeneous Poisson generator

**Step 3**: Monte Carlo sequential estimation of the kinematics (decoding)
For each time \( k \), a set of samples for state \( \tilde{x}_k \) are generated, \( i=1:N \)
1. Predict new state samples \( \tilde{x}_k = F_k \tilde{x}_{k-1} + \eta_k, i=1:N \)
2. For each neuron \( j \),
   1) Estimate the conditional firing rate \( \lambda_{ij}^k = f^j(\tilde{k}^j \cdot \tilde{x}_k), i=1:N \)
   2) Update the weights \( w_{ij}^k \propto p(\Delta N_k^j \mid \lambda_{ij}^k), i=1:N \)
3. Draw the weight for the joint posterior density \( W_k = \prod_j w_{ij}^k, i=1:N \)
4. Normalize the weights $W^i_k = \frac{W^i_k}{\sum_i W^i_k}$, $i=1:N$

5. Draw the joint posterior density $p(\tilde{x}_k \mid N_{1:k}) \approx \sum_{i} W^i_k \cdot k(\tilde{x}_k - \tilde{x}^i_k)$

6. Estimate the state $\tilde{x}^*_k$ from the joint posterior density by MAP or expectation.

7. Resample $\tilde{x}^i_k$ according to the weights $W^i_k$.

**Monte Carlo SE Decoding Results in Spike Domain**

In this section, we show the BMI decoding results directly in the spike domain by implementing the Monte Carlo sequential estimation framework.

We first preprocessed the 185 channels of the neuron spiking time as a 0, 1 point process. For each neuron in the ensemble, an optimum time interval of 10 ms was selected to construct the point process observation sequence. With this interval, 94.1% of the intervals with spikes had only a single spike. For each time interval and in each channel, 1 was assigned when there were one or more spikes, otherwise 0 was assigned. 185 multi-channel spike trains were generated 1750 seconds long. The recorded 2-D position vector $\tilde{p}$ is interpolated to be synchronized with the spike trains. The velocity $\tilde{v}$ is derived as the difference between the current and previous positions, and the acceleration $\tilde{a}$ is derived the same way from the velocity.

Here, the state vector is chosen as the instantaneous kinematics vector $\tilde{x} = \begin{bmatrix} \tilde{p}_x & \tilde{v}_x & \tilde{a}_x & \tilde{p}_y & \tilde{v}_y & \tilde{a}_y \end{bmatrix}^T$ to be reconstructed directly from the spike trains, rather than choosing only the velocity during a window when a spike appears. Therefore, the kinematics vector contains more information about positions, velocities and accelerations. As we discussed in the tuning analysis section, the information theoretical tuning depths computed from each kinematics can be different, indicating that there are neurons tuned specifically to a particular
kind of kinematics. Using only one kinematic variable might leave out important information
between the neural spikes and other kinematics.

After data preprocessing, the kinematics model $F_k$ can be estimated using the least squares
solution as shown in Equation 2-6. Notice that carefully choosing the parameters in the noise
estimation (the noise distribution $p(\eta)$ in Monte Carlo SE) could affect the algorithm
performance. However, since we have no access to the desired kinematics in the test data set, the
parameter of both algorithms were estimated from the training data sets. In the Monte Carlo SE
model, the noise distribution $p(\eta)$ is approximated by the histogram of $\eta_k = x_k - F_k x_{k-1}$. The
resolution parameter was experimentally set to 100 to approximate the noise distribution. The
regularization factor $\alpha$ in the tuning function was experimentally set at $10^{-7}$ for this analysis.
The remaining parameters in Monte Carlo SE include the kernel size $\sigma$ selected at 0.02 and the
number of particles $x_n$ experimentally set at 1000, for a reasonable compromise between
computational time and estimation performance. This kernel size should be chosen carefully to
not lose the characteristics of the tuning curve, while still minimizing ripples in the estimated
density.

Monte Carlo SE algorithm produces stochastic outputs because of the Poisson spike
generation model. It also introduces variations between realizations even with fixed parameters
due to the estimation of the posterior distribution with the particles.

Table 4-1 shows reconstruction results on a 1000 sample of test segment (time index from
25401 to 26400) of neural data. Correlation Coefficients (CC) and Normalized Mean Square
Error (MSE normalized by the power of the desired signal) between the desired signal and the
estimations are evaluated for the Monte Carlo SE using 20 realizations. We show the mean and
the standard derivation among realizations, together with the best and the worst performance obtained by single realization.

Our approach resulted in reasonable reconstructions of the position and the velocity. The position shows the best correlation coefficient with the true trajectory. This result may be due to the fact that the velocity and the acceleration were derived as differential variables, where the noise in the estimation might be magnified. The Monte Carlo SE obtains the tuning function nonlinearity for each neuron from the training data and estimates the kinematics without any restriction on the posterior density. The average correlation for the position along $x$ is $0.8058 \pm 0.0111$ and along $y$ is $0.8396 \pm 0.0124$. The average correlation for velocity along $x$ is $0.7945 \pm 0.0104$ and along $y$ is $0.7381 \pm 0.0057$. We notice that although Monte Carlo SE introduces differences on the reconstruction among realizations due to stochasticity, the variance of the results is pretty small.

Figure 4-2 zooms in the first 100 samples of the reconstructed kinematics to show better the modeling accuracy. The left and right column plots display the reconstructed kinematics for $x$-axis and $y$-axis. The 3 rows of plots illustrate from top to bottom the reconstructed position, the velocity and the acceleration. In each plot, the red dash line is the desired signal. The blue line is the reconstructed kinematics by one trial of Monte Carlo SE. The gray area in each plot represents the posterior density estimated by the algorithm over time where the darker areas represent a higher value. As the value of the posterior density decreases to 0, the color of the dots will fade to white. Figure 4-2 shows the Monte Carlo SE effectiveness to generate samples whose density follows the trajectory. The desired signal falls almost always within the high probability range of the posterior density, which demonstrates the good tracking ability of Monte Carlo SE.
Since the desired signal in the test set data is formally unknown, it is not reasonable to just pick the best realization to present the reconstruction results. Here, we choose the averaged performance among realizations as the reconstruction results by Monte Carlo SE.

Figure 4-3 shows the averaged performance by Monte Carlo SE to reconstruct kinematics from all 185 neuron spike trains for 1000 test samples. The left and right column plots display the reconstructed kinematics for $x$-axis and $y$-axis. The 3 rows of plots illustrate from top to bottom the reconstructed position, the velocity and the acceleration. In each subplot, the red line indicates the desired signal, and the blue line indicates the expectation estimation. The correlation coefficients between the desired signal and the estimations were shown in Table 4-2.

We further compared the statistical performance of both algorithms on 8000 test data samples (80 seconds) of neural data. The performance averaged among the decoding results from 20 Monte Carlo trials is chosen as the reconstruction result by Monte Carlo SE. CC and NMSE were both evaluated with an 800 sample-long window with 50% overlap. The reconstruction performance is shown in Table 4-3.

As for the Figure of merit for reconstruction, the correlation coefficient has been the preferred metric to compare movement reconstruction between different experimental data sets in BMIs [Wessberg et al. 2000]. However, it may not be sufficient to evaluate the accuracy of BMI algorithm, since a bias in position means that a different point in the external space will be targeted, so the rating criterion should take this bias into consideration to properly compare reconstruction models. Notice also that the correlation coefficient obtained from the acceleration is pretty low. However, if we visually check the reconstruction results in Figure 4-3, the algorithm actually follows the trend of the desired signal closely. The problem with the NMSE for BMIs is that the results do not “look as good”, with errors sometimes bigger than the power
of the trajectory. This can be observed in Figure 4-3, where the reconstructed position seems to have a different scale of the desired trajectory. Therefore, NMSE is also chosen as another criterion to evaluate the tracking accuracy of the animal’s true movement trajectory.

**Parameter Study for Monte Carlo SE Decoding in Spike Domain**

Although the results are interesting, Monte Carlo SE for spike modeling needs to be further developed. They are substantially more complex than the ones for random processes, and many parameters are assumed and need to be estimated with significant design expertise. There are 4 parameters in Monte Carlo SE for point process in need to be tuned. Three of them during the encoding process (training stage), regularization factor $\alpha$ in kinematics correlation matrix inverse (default $10^{-7}$), kernel size $\sigma$ in nonlinearity smooth (default 0.02), and the resolution parameter in approximation of noise distribution $p(\eta)$ in state dynamic model (default 100). The fourth parameter occurs in the decoding process and relates to the number of samples $x_n$ of particle $\vec{x}_i$ in posterior density estimation (default 1000). Therefore we will evaluate the encoding/decoding performance as a function of these parameters. For each parameter, 5 different values are tried with all the other parameters set at the default values.

**Regularization factor $\alpha$.** It is used to calculate the inverse of the correlation matrix of the kinematics $(E[\vec{x} \vec{x}^T] + \alpha I)^{-1}$. The parameter $\alpha$ is supposed to be a small positive number in order to properly condition the inverse of the correlation matrix of the kinematics, when the minimal eigenvalue is close to 0. However, it should be insignificant compared to the maximal eigenvalue of the correlation matrix, otherwise it would disturb the eigenvalue structure. Notice that one way to experimentally set the proper $\alpha$ is to check how $\alpha$ affects the linear model error between linear output and desired signal. Here we set $\alpha = [0 \ 10^{-7} \ 10^{-5} \ 10^{-3} \ 10^{-1}]$. The error between the linear model output and the desired signal in terms of different $\alpha$ is shown in
Figure 4-4. As before, the left and right column plots display the reconstructed kinematics for $x$-axis and $y$-axis. The 3 rows of plots illustrate from top to bottom the error for position, the velocity and the acceleration. We can see that when $\alpha$ is smaller than $10^{-5}$, there is almost no significant difference between the errors. However we have only access to training data, a very small value will be safer ($10^{-7}$) for the test data.

**The resolution parameter $p(\eta)$.** It is in approximating the noise distribution in the state dynamic model $\eta_k = x_k - F_k x_{k-1}$. Density is the number of samples to approximate the $cdf$ of the noise distribution during the training. The greater the density is, the better the $cdf$ approximates to the true one, together with more computation. Here we set density $= [20, 50, 100, 200, 500]$. Figure 4-5 shows the $cdf$ of the noise distribution obtained from training set using different density values. We can see that when the density is larger than 100, the $cdf$ lines overlap. Therefore 100 is a proper choice to approximate the $cdf$ of the noise distribution in our experimental data.

**Kernel size $\sigma$.** It is used to smooth the nonlinearity in tuning estimation. Here we only study the kernel size for the important neurons, which contribute most to shape the posterior density of the kinematics. If the kernel size is too small, there will be ripples on the conditional $pdf$, which brings a large variance in nonlinearity estimation. If the kernel size is too big, it will smooth out the difference between joint $pdf$ and marginal $pdf$, which results in the under-estimation of the conditional $pdf$. Here we set $\sigma = [0.005, 0.01, 0.02, 0.05, 0.1]$. Figure 4-6 shows the nonlinearity of neuron 72 (one of the important tuning neurons) smoothed by different kernel sizes. We can see that when $\sigma$ is 0.005, there are a few ripples on the nonlinear tuning curve. Even when $\sigma$ is 0.01, there are still ripple at both extreme ends due to insufficient samples. When $\sigma$ is too big (0.05 and 0.1), the tuning curve is underestimated. We check $\sigma$ for all
neurons, especially focus on the important tuning neurons. 0.02 is an empirical middle ground to smooth the nonlinearity in tuning.

The sample number $x_n$. It refers to the number of particle $\tilde{x}_k$ in posterior density estimation is the only free parameter during the decoding process. This parameter describes the accuracy of the posterior density estimation at each time index. It also brings the main drawback of the approach, the high computational complexity, because each of the samples will be evaluated to construct the shape of the posterior density. Here we set the sample number $x_n = [200, 500, 1000, 1500, 2000]$. Figure 4-7 shows the averaged decoding results through 20 Monte Carlo trials of the kinematics reconstruction with different $x_n$. The left and right column plots display the reconstructed kinematics for $x$-axis and $y$-axis. The 3 rows of plots illustrate from top to bottom the reconstructed performance of the position, the velocity and the acceleration. In each plot, the $x$-axis shows the value of $x_n$. The blue solid line is CC between the reconstruction and desired signal. The green dash line is NMSE between the reconstruction and desired signal. We can see that CCs don’t change obviously for all kinematics even using much higher $x_n$, but the NMSE clearly shows the decrease trend when $x_n$ is bigger. Although the performance convergences with very large value of $x_n$, it would also bring a large computational burden to decoding. To comprise between the accuracy and computational complexity, we choose 1000 samples where the decoding of most of the kinematic variables start to converge.

Synthesis Averaging by Monte Carlo SE Decoding in Spike Domain

The Monte Carlo sequential estimation for point processes contains two sources of stochasticity, the generation of the samples to reconstruct the posterior density and the very nature of the single neuron firings that is modeled as a Poisson point process. While the former was dealt with the Monte Carlo method (averaging several realizations), the later is still present
in our results due to the coarse spatial sampling of neural activity produced by the limited number of electrodes. This coarse sampling has two basic consequences. First, the multi electrode array collects activity from only some of these neural assemblies, which means that the Monte Carlo sequential estimation model output will have an error produced by not observing all the relevant neural data. This problem will always be present due to the huge difference in the number of motor cortex neurons and electrodes. Second, even when a given neural assembly is probed by one or a few neurons, it is still not possible to achieve accurate modeling due to the stochasticity embedded in the time structure of the spike trains. To remove it, one would have to access the intensity function of neural assemblies that are transiently created in motor cortex for movement planning and control, which are deterministic quantities.

This means that every neuron belonging to the same neural assembly will display slightly different spike timing, although they share the same intensity function. Since each probed neuron drives an observation model in the BMI, there will be a stochastic term in the output of the BMI (kinematics estimation) that can only be removed by averaging over the neural assembly populations. However, we can attempt to decrease this variance by estimating the intensity function from the probed neuron and from it generate several synthetic spike trains, use them in the observation model and average the corresponding estimated kinematics. Since this averaging is done in the movement domain (and if the process would not incur a bias in the estimation of the intensity function) the time resolution would be preserved, while the variance would be decreased. We call this procedure synthetic averaging and it attempts to mimic the population effect in the cortical assemblies. This averaging is rather different from the time average that is operated in binning, which looses time resolution in the reconstructed kinematics.
The synthetic spike trains are generated by an inhomogeneous Poisson process with a mean value given by the estimated intensity function obtained by kernel smoothing. This is repeated for each neuron in the array. During testing these synthetic spike trains play the same role as the true spike trains to predict the kinematics on-line. Of course this will increase the computation time proportionally to the number of synthetic spike trains created. In a sense we are trying to use computer power to offset the limitations of probing relatively few neurons in the cortex. Since the errors in prediction have a bias and a variance which are not quantified, it is unclear at this point how much better performance will become, but this will be addressed in the validation.

As we analyzed in the previous section, in order to deal with the intrinsic stochasticity due to the randomness of the spike trains, we proposed the synthetic averaging idea to mimic the neuron population effect. Instead of decoding only from current spike trains, we use a Poisson generator to obtain 20 sets of spike trains from each neuron as synthetic plausible observations to represent the neuron ensemble firing with the same intensity function. This firing intensity function is estimated by kernel smoothing from each recorded spike trains. The kernel size is experimentally set as 0.17. In order to preserve the timing resolution the averaging is performed across the estimated kinematics of each group (including the output of the true spike train). Table 4-4 shows the comparison results of the performance by Monte Carlo SE averaged among 20 realizations on recorded real spike train and the “deterministic” averaged performance over Monte Carlo and synthetic data (20 sets re-generated spike trains, 20 Monte Carlo trials for each set) in the same segment of test data (time index 215401 to 216400).

Both approaches as well as the deterministic performance resulted in reconstruction with similar correlation coefficients. However, the average over synthetic data shows smoother
kinematics reconstruction with reduced NMSE comparing to the averaged performance through 20 Monte Carlo trials on original spike trains. NMSE reduces 26% for position along $x$, 18% for position along $y$, and on average 15% for all 6 kinematic variables. Therefore we can conclude that the reconstruction accuracy measured by NMSE has a large component due to the variance intrinsic in the spike firing, but does not affect the general trend of the reconstructed signal as measured by the CC.

We further compared the statistical performance of both algorithms on 8000 test data samples of neural data. The performance averaged among the decoding results from 20 sets regenerated spike trains is chosen as the reconstruction result by Monte Carlo SE. We run the decoding process for 20 Monte Carlo trials on each set of synthetic spike trains. CC and NMSE were both evaluated with an 800 sample-long window with 50% overlap. For each segment of data, pair-wise student $t$-test was performed to see if the synthetic averaging (SA) results are statistically different from the averaged performance by recorded neuron spike train alone (MCSE). The test is performed against the alternative specified by the left tail test $CC_{MCSE} < CC_{SA}$ for each kinematic variable. Comparing NMSE by both approaches, the test is performed against the alternative specified by the right tail test $NMSE_{MCSE} > NMSE_{SA}$ for each kinematic variable. All the tests are performed on the null hypothesis at $\alpha = 0.05$ significance level. Under the null hypothesis, the probability of observing a value as extreme or more extreme of the test statistic, as indicated by the $p$-value, is shown in Table 4-5.

Except the position $x$ and the velocity $y$ from this first case, we could not see CC by synthetic averaging is significantly larger than the one Monte Carlo SE ($p \leq 0.05$), as statistically verified using the $t$-test. In terms of NMSE, however, the $t$-test verifies that the
Monte Carlo SE reconstruction is statistically better than the Monte Carlo SE for most kinematic variables.

This result demonstrates that using the simulated neuron population attenuates the variability intrinsic in the coarse sampling of a given neural population, effectively trading computation for lack of more neural channels belonging to the same neural population. However, this procedure only reduces the kinematics estimation error that is due to the variance of the recorded spike train. It cannot do anything against the lack of information produced by the coarse sampling of other neural population involved in the movement but not sampled at all. On the other hand, the procedure creates a modeling bias because the intensity function is estimated from a single neuron, but it is very difficult to quantify. Since the results improve as measured by NMSE, overall the synthetic averaging method gains more than it looses. When compared with the averaging done in time by binning, the averaging in the kinematics domain bypasses the lack of resolution problem and still smoothes the reconstruction.

**Decoding Results Comparison Analysis**

Several signal-processing approaches have been applied to predict movements from neural activities. Many decoding methodologies use binned spike trains to predict movement based on linear or nonlinear optimal filters [Wessberg et al., 2000; Sanchez et al., 2002b; Kim et al., 2003]. These methods avoid the need for explicit knowledge of the neurological dynamic encoding properties, and standard linear or nonlinear regression is used to fit the relationship directly into the decoding operation. Yet another methodology can be derived probabilistically using a state model within a Bayesian formulation [Schwartz et al., 2001; Wu et al., 2006; Brockwell et al., 2004] as we did in our Monte Carlo SE for point process. The difference is all the previous algorithms are coarse approaches that do not exploit spike timing resolution due to binning and may exclude rich neural dynamics in the modeling. Monte Carlo SE for point
process decodes the movement in spike domain. It is important to compare our algorithm to other Bayesian approaches that have been applied to BMI in terms of their different assumptions and decoding performance.

**Decoding by Kalman**

The Kalman filter has been applied to BMIs [Wu et al., 2006] to reconstruct the kinematics as the state from continuous representation of neural activities (*i.e.*, using binned data). When seen as a Bayesian approach, the 2 basic assumptions of the Kalman filter are the linearity and Gaussian distributed posterior density. In another word, both the kinematic dynamic model and the tuning function are assumed to be strictly linear, and the posterior density of the kinematics state given current neural firing rates are Gaussian distributed at each time index. In this way, the posterior density can be represented in close form with only 2 parameters, mean and variance of *pdf*. To apply Kalman filter on our BMI data, the state dynamic remains the same as

\[ x_k = F_k x_{k-1} + \eta_k \]  

(4-4)

where \( F_k \) establishes the dependence on the previous state and \( \eta_k \) is zero-mean Gaussian distributed noise with covariance \( Q_k \). \( F_k \) is estimated from training data by the least square solution. \( Q_k \) is estimated as the variance of the error between the linear model output and the desired signal. The tuning function is linearly defined as

\[ \lambda_t = H \cdot \tilde{x}_{t+\text{lag}} + n_k \]  

(4-5)

where \( \lambda_t \) is the firing rate by 100ms window binning. \( \tilde{x}_t \) is the instantaneous kinematics vector defined as \( [\vec{p}_x \, \vec{v}_x \, \vec{a}_x \, \vec{p}_y \, \vec{v}_y \, \vec{a}_y \, 1]^T \) with 2-dimensional information of position, velocity, acceleration and bias term. The variable \( \text{lag} \) refers to the causal time delay between motor cortical neuron activity and kinematics due to the propagation effects of signals thru the motor
and peripheral nervous systems. Here it is experimentally set as 200 ms [Wu et al., 2006, Wang et al., 2007b]. \( n_k \) is zero-mean Gaussian distributed noise with covariance \( R_k \). The weight estimation of the linear filter \( \mathbf{H} \) is given from training data by

\[
\mathbf{H} = (E[\mathbf{x}_{t+\text{lag}}^T \mathbf{x}_{t+\text{lag}}])^{-1} E[\mathbf{x}_{t+\text{lag}} \mathbf{\lambda}_t]
\] (4-6)

Equation 4-6 represents the least square solution for the linear tuning function. The kinematics vector is then derived as the state from the observation of firing rate in test by Equations 4-7 a-e.

\[
\mathbf{x}_{k|k-1} = F_k \mathbf{x}_{k-|k-1}
\] (4-7 a)

\[
P_{k|k-1} = F_k P_{k-|k-1} F_k^T + Q_k
\] (4-7 b)

\[
K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}
\] (4-7 c)

\[
P_{k|k} = F_k^T P_{k-|k-1} F_k + Q_k
\] (4-7 d)

\[
\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + K_k (\lambda - H_k \mathbf{x}_{k|k-1})
\] (4-7 e)

**Decoding by Adaptive Point Process**

Adaptive filtering of point processes provides an analytical solution to the state estimation in the spike domain. Therefore, it requires a parametric model for the neuron tuning in closed form. Many different functional forms of tuning have been proposed, consisting mostly of linear projections of the neural modulation on 2 or 3 dimensions of kinematic vectors and bias. Moran and Schwartz [1999] also introduced a linear relationship from motor cortical spiking rate to speed and direction. Brockwell et al. [2003] assumed an exponential tuning function for their motor cortical data. Here we have tried both tuning functions for our BMI data.

**Exponential tuning**

The exponential tuning function is estimated from 10000 samples of the training data as

\[
\lambda_t = \exp(\mathbf{H} \cdot \mathbf{x}_{t+\text{lag}})
\] (4-8)
where $\lambda_i$ is the firing probability for each neuron, obtained by smoothing the spike train with a Gaussian kernel. The kernel size is empirically set to be 0.17 in the experiment [Wang et al., 2007c]. $\tilde{x}_t$ is the instantaneous kinematics vector defined as $[\tilde{p}_x, \tilde{v}_x, \tilde{a}_x, \tilde{p}_y, \tilde{v}_y, \tilde{a}_y, 1]^T$ with 2-dimensional information of position, velocity acceleration, and bias. The variable lag refers to the causal time delay between motor cortical neuron activity and kinematics due to the propagation effects of signals thru the motor and peripheral nervous systems. Here it is experimentally set as 200 ms as well [Wu et al., 2006; Wang et al., 2007c]. The weight estimation of the linear filter $\tilde{H}$ is given from the training data by

$$
\tilde{H} = (E[\tilde{x}_{t+\text{lag}}^T \tilde{x}_{t+\text{lag}}])^{-1} E[\tilde{x}_{t+\text{lag}} \log(\lambda_i)]
$$

Equation 4-10 represents the least square solution for the linear adaptive filter in log likelihood form. During operation, most likely some firing rates are close to 0, which results in extremely negative numbers. Therefore, we add a small positive number, defined as 10% of the mean firing rate during training for each neuron, which makes the firing rate always positive. The exponential tuning function in Equation 4-8 defines the first and second derivative terms in Equations 2-7c and 2-7d as

$$
\frac{\partial \log \lambda_i}{\partial \tilde{x}_{t+\text{lag}}} = \tilde{H}^T
$$

$$
\frac{\partial^2 \log \lambda_i}{\partial \tilde{x}_{t+\text{lag}} \partial \tilde{x}_{t+\text{lag}}} = 0
$$

The kinematics vector is then derived as the state from the observation of multi-channel spikes train for the test samples by Equations 2-7a-d in Chapter 2.
Kalman point process

Notice that when a linear tuning function is selected for the observation model together with a Gaussian assumption for the posterior density, the end result is actually a Kalman filter in the spike domain and will be called Kalman filter for point process (PP). Here the linear tuning function is estimated from 10000 samples of the training data as

\[ \lambda_i = \vec{h} \cdot \vec{x}_{t+\text{lag}} + \vec{B} \]  \hspace{1cm} (4-13)

\[ \text{spike}_i = \text{Poisson}(\lambda_i) \]  \hspace{1cm} (4-14)

where \( \lambda_i \) is the firing probability for each neuron, obtained by smoothing the spike train with a Gaussian kernel. The kernel size is empirically set to be 0.17 in the experiment [Wang et al., 2007c]. \( \vec{x}_t \) is the instantaneous kinematics vector defined as \([\vec{p}_x, \vec{v}_x, \vec{a}_x, \vec{p}_y, \vec{v}_y, \vec{a}_y]^T\) with 2-dimensional information of position, velocity and acceleration. The variable lag refers to the causal time delay between motor cortical neuron activity and kinematics due to the propagation effects of signals thru the motor and peripheral nervous systems. Here it is experimentally set as 200 ms [Wu et al., 2006; Wang et al., 2007c]. We extend the kinematics vector as \([\vec{p}_x, \vec{v}_x, \vec{a}_x, \vec{p}_y, \vec{v}_y, \vec{a}_y, 1]^T\) to include a bias \( \vec{B} \), which can be regarded as part of the weights of the linear filter \( \vec{H} \). The tuning function is then \( \lambda_i = \vec{H} \cdot \vec{x}_{t+\text{lag}} \). The weight estimation of the linear filter \( \vec{H} \) is given by

\[ \vec{H} = (E[\vec{x}_{t+\text{lag}}^T \vec{x}_{t+\text{lag}}])^{-1} E[\vec{x}_{t+\text{lag}} \lambda_i] \]  \hspace{1cm} (4-15)

Equation 4-15 represents the least square solution for the linear adaptive filter, where

\( E[\vec{x}_{t+\text{lag}}^T \vec{x}_{t+\text{lag}}] \) gives the autocorrelation matrix \( R \) of the input kinematics vector considering a causal time delay. \( E[\vec{x}_{t+\text{lag}} \lambda_i] \) gives the cross-correlation vector \( P \) between the input and the
firing probability. The linear tuning function in Equation 4-13 defines the first and second
derivative terms in Equations 2-7c and 2-7d in Chapter 2 as

$$\frac{\partial \log \lambda_t}{\partial \tilde{x}_{t+\text{lag}}} = \frac{\tilde{H}^T}{\lambda_t}$$  \hspace{1cm} (4-16)$$

$$\frac{\partial^2 \log \lambda_t}{\partial \tilde{x}_{t+\text{lag}} \partial \tilde{x}_{t+\text{lag}}} = -\frac{\tilde{H} \cdot \tilde{H}^T}{\lambda_t^2}$$  \hspace{1cm} (4-17)$$

The kinematics vector is then derived as the state from the observation of multi-channel
spike trains for the test samples by Equation 2-7a-d in Chapter 2.

**Performance Analysis**

Our Monte Carlo SE for Point Process is designed to estimate the kinematics state directly
from spike trains. The posterior density is estimated non-parametrically without Gaussian
assumptions, which allows the state model and the observation model to be nonlinear. It is
important to compare the performance of the Monte Carlo SE with the other algorithms on the
same data set to validate all the assumptions. First, to evaluate the performance advantages of a
nonlinear & non-Gaussian model, we compare it with the Kalman PP, which works in spike
domain with linear tuning function and assumes the posterior density Gaussian distributed.
Secondly, the Monte Carlo SE is utilizes a tuning function that is estimated non-parametrically
directly from data. It would be interesting to compare the decoding performances with the
different tuning models, such as the Gaussian tuning curve and the exponential tuning curve.
Thirdly, all the algorithms assume stationary tuning function between the training and test
datasets. To study the decoding performance separately in training and testing would provide us
some idea how the tuning function could be changing over the time. Fourthly, the following
question should be asked. How is the performance in the spike domain compared to working on
the conventional spike rates? The above questions will be analyzed in details in the following sections.

**Nonlinear & non-Gaussian vs. linear & Gaussian**

The point process adaptive filtering with linear observation model and Gaussian assumption (Kalman filter PP) and the proposed Monte Carlo SE framework were both tested and compared in a BMI experiment for the 2-D control of a computer cursor using 185 motor cortical neurons [Nicolelis et al., 1997; Wessberg et al., 2000] as before.

After data preprocessing, the kinematics model $F_k$ for both algorithms can be estimated using the least squares solution. Notice that carefully choosing the parameters in the noise estimation (covariance $Q_k$ in Kalman PP and the noise distribution $p(\eta)$ in Monte Carlo SE) could affect the algorithm performance. However, since we have no access to the desired kinematics in the test data set, the parameter estimations of both algorithms were obtained from the training data sets. For the Kalman filter PP, the noise in the kinematics model (Equation 2-6) is approximated by a Gaussian distribution with covariance $Q_k$. In the Monte Carlo SE model, the noise distribution $p(\eta)$ is approximated by the histogram of $\eta_k = x_k - F_k x_{k-1}$. The resolution parameter was experimentally set to 100 to approximate the noise distribution. The regularization factor $\alpha$ in the tuning function was experimentally set at $10^{-7}$ for this analysis.

The remaining parameters in Monte Carlo SE include the kernel size $\sigma$ selected at 0.02 and the number of particles $x_n$ experimentally set 1000, for a reasonable compromise between computational time and estimation performance. This kernel size is chosen carefully to not lose the characteristics of the tuning curve as we study before.

As we have analyzed before, both algorithms produce stochastic outputs because of the Poisson spike generation model. However, the Kalman filtering PP has an analytical solution
with recursive close form equations. We set the initial state \( \tilde{x}_0 \) to be the zero vectors and the state variance \( P_{00} \) is estimated from the training data. Once the initial condition and parameters are set, the state estimation is determined uniquely by the spike observations. However, the Monte Carlo SE approach introduces variations between realizations even with fixed parameters due to the estimation of the posterior distribution with the particles. Since the desired signal in the test set data is formally unknown, it is not reasonable to just pick the best realization to present the reconstruction results. Here, we choose the averaged performance among realizations as the reconstruction results by Monte Carlo SE, and compare with the Kalman filter PP results.

Table 4-6 shows reconstruction results on a 1000 sample of a test segment (shown in Figure 4-7) of neural data. Correlation Coefficients (CC) and Normalized Mean Square Error between the desired signal and the estimations are evaluated for the Kalman filter PP as well as for the Monte Carlo SE using 20 realizations of the posterior. For the second approach we also show the mean and the standard derivation among realizations, together with the best and the worst performance obtained by single realization.

Both approaches resulted in reasonable reconstructions of the position and the velocity. The position shows the best correlation coefficient with the true trajectory. This result may be due to the fact that the velocity and the acceleration were derived as differential variables, where the noise in the estimation might be magnified. Although the Kalman filter PP assumes a Gaussian posterior and a simple linear model for both the kinematic dynamic system and the tuning function, it obtains a reasonable reconstruction of the position and the velocity. For the position CC = 0.7422 for the \( x \) direction and CC = 0.8264 for the \( y \) direction. The velocity shows a CC = 0.7416 for \( x \) and CC = 0.6813 for \( y \). The Monte Carlo SE obtains the tuning function nonlinearity for each neuron from the training data and estimates the kinematics without any
restriction on the posterior density. The average correlation for the position along $x$ is $0.8058 \pm 0.0111$ and along $y$ is $0.8396 \pm 0.0124$. The average correlation for velocity along $x$ is $0.7945 \pm 0.0104$ and along $y$ is $0.7381 \pm 0.0057$. The Monte Carlo SE is better than the Kalman filter PP in terms of both CC and NMSE.

Figure 4-8B shows the reconstructed kinematics using both algorithms from all 185 neurons for 1000 testing samples. As before, the left and right panels depict respectively the reconstructed kinematics for $x$-axis and $y$-axis. The 3 rows of plots from top to bottom display respectively the reconstructed position, the velocity and the acceleration. In each subplot, the red dash line indicates the desired signal, the blue solid line indicates the estimation by Monte Carlo SE, and green dotted line indicates the estimation by Kalman filtering PP. For clarity, Figure 4-8B also shows the 2D reconstructed position for a segment of the testing samples by two methods. The Monte Carlo approach offers the most consistent reconstruction in terms of both correlation coefficient and normalized mean square error.

The simulation of both models with synthetic data provides important hints on how to interpret the results with real neural data. The linear tuning model by the Kalman filter PP provides less accuracy in the nonlinear region of the tuning function, which in turn affects the decoding performance. Moreover, the Kalman filter PP also assumes the posterior density is Gaussian, therefore both algorithms provide similar velocity estimation along $y$ when both assumptions are verified. When the estimation from the two algorithms are different (often occur at the peak of the desired signal), the Monte Carlo SE model usually provides better performance, which is due to either its better modeling of the neuron’s nonlinear tuning or its ability to track the non-Gaussian posterior density better.
We further compared the statistical performance of both algorithms on 8000 test data samples of neural data. The performance averaged among the decoding results from 20 sets regenerated spike trains (20 realizations each set) is chosen as the reconstruction result by Monte Carlo SE. CC and NMSE were both evaluated with an 800 sample-long window with 50\% overlap. For each segment of data, pair-wise student \( t \)-test was performed to see if the results are statistically different from the Kalman filter PP. The test is performed against the alternative specified by the left tail test \( CC_{\text{Kalman}} < CC_{\text{MCSE}} \) for each kinematic variable. Comparing NMSE by both approaches, the test is performed against the alternative specified by the right tail test \( \text{NMSE}_{\text{Kalman}} > \text{NMSE}_{\text{MCSE}} \) for each kinematic variable. All the tests are performed on the null hypothesis at \( \alpha = 0.05 \) significance level. Under the null hypothesis, the probability of observing a value as extreme or more extreme of the test statistic, as indicated by the \( p \)-value, is shown in Table 4-7.

Except the \( x \) position and the \( y \) acceleration from this first case, the CC of the Monte Carlo SE of all other kinematic variables is significantly larger than the Kalman filter PP (\( p \leq 0.05 \)), as statistically verified using the \( t \)-test. In terms of NMSE, however, the \( t \)-test verifies that the Monte Carlo SE reconstruction is statistically better than the Kalman filter PP for all kinematic variables.

**Exponential vs. linear vs. LNP in encoding**

We have shown 2 tuning models in implementing the adaptive filtering on point process. Comparing the decoding performance of these 2 different encoding (tuning) models with the Gaussian distributed posterior density could show the importance of choosing an appropriate tuning model for the decoding methodology.
Both tuning models were implemented as BMI decoders in the spike domain. The point process generation is the same as we described for Kalman PP in previous section. After data preprocessing, the parameter estimation of both algorithms were obtained from the training data sets. For the exponential filter PP, the noise in the kinematics model is the same as the one in Kalman PP. We set the initial state $\mathbf{x}_0$ to be the zero vectors and the state variance $P_{00}$ is estimated from the training data. Once the initial condition and parameters are set, the state estimation is determined uniquely by the spike observations.

Table 4-8 shows the statistical reconstruction results on 8000 samples of test neural data. NMSE between the desired signal and the estimations by exponential PP and Kalman PP are evaluated with an 8 sec window with 50% overlap, together with the performance by Monte Carlo SE.

Kalman filter PP gives better performance in position $y$ but worse performance in position $x$ comparing to exponential PP. For all the other kinematic variables, both encodings give similar performances. We can infer that the proper tuning function to decode the kinematics on-line would be somehow between linear and exponential curves. The performance comparing to one by Monte Carlo SE shows that the instantaneous tuning curves we evaluate directly from the data catches more information than both linear and exponential curves, which provides the best decoding results. However, it is a very time consuming operation as described before.

**Training vs. testing in different segments – nonstationary observation**

As we have mentioned before, all the parameters of the tuning curves were estimated from the training data and remain the same in the testing segments. The big assumption behind this methodology is stationary of the tuning properties over time, which may not be true. One way to test this assumption is to see the performance comparison among the training data and different
testing data. Here time index of the training set is from 113500 ms to 193500 ms. The time index for testing set 1 is from 213500 ms to 293500 ms, which is right after the training data. The second testing set is chosen from 1413500 ms to 1493500 ms, which is far from the training data. For each data set, the statistical reconstruction results on 8000 samples of neural data. Both CC and NMSE between the desired signal and the estimations by exponential PP, Kalman PP and Monte Carlo PP are evaluated with an 8 sec window with 50% overlap. Figure 4-9 A and 4-9 B shows the performance trends between the training and different test sets in terms of CC and NMSE respectively. The left and right panels depict respectively the reconstructed kinematics for x-axis and y-axis. The 3 rows of plots from top to bottom display respectively the reconstructed performances for position, velocity and acceleration. In each subplot, the green bar indicates the mean and variance of the estimation performance for 3 different data sets by Kalman filtering PP, the cyan line indicates the statistical estimation performance by Exponential filtering PP, and the blue line indicates the statistical estimation performance by Monte Carlo SE.

For both criteria, all the algorithms show clearly similar trends of statistical performance. The reconstruction on test data 1 is slightly worse than the reconstruction in training data. However, in the test data 2, which is quite far from the training data, the performance is much worse. It means the stationary assumption in both training and testing is questionable. It might be allowed in the testing segment close right after the training because the change of the tuning property is not obvious. But it would result in poor estimation when the tuning properties change after some time. Therefore, the study on the non-stationary tuning property and the corresponding tracking in the decoding algorithm is necessary.

**Spike rates vs. point process**

One way to test the decoding difference between continuous variables (spike rates) and point processes is to compare the performance of the Kalman filter and Kalman PP on the same
segment of test data, because both filters have linear tuning and Gaussian distributed posterior density assumptions. The difference is Kalman filtering reconstructs the kinematics state from continuous representation of neural activities – the binning firing rate, while Kalman PP directly works in spike domain. For Kalman filtering, 100 msec binning window is used to process the spike times into continuous firing rates for each neuron in the ensemble. For Kalman PP, the preprocessing to construct the point process observation sequence remains the same as Monte Carlo SE does.

After data preprocessing, the kinematics model \( F_k \) for both algorithms can be estimated using the least squares solution. Notice that carefully choosing the parameters in the noise estimation (covariance \( Q_k \) in both Kalman and Kalman PP) could affect the algorithm performance. However, since we have no access to the desired kinematics in the test data set, the parameter estimations of both algorithms were obtained from the training data sets. The noise in the kinematics model is approximated by a Gaussian distribution with covariance \( Q_k \).

The Kalman filtering PP algorithms produce stochastic outputs because of the Poisson spike generation model. Both have analytical solutions with recursive close form equations. We set the initial state \( \tilde{x}_0 \) to be the zero vectors and the state variance \( P_{00} \) is estimated from the training data. Once the initial condition and parameters are set, the state estimation is determined uniquely by the spike observations.

Table 4-9 shows the statistical reconstruction results on 8000 samples of training and 8000 samples of test segment of neural data. Since the desired signals of Kalman filter and Kalman PP are obtained differently, here only Correlation Coefficients (CC) between the desired signal and the estimations are evaluated with an 8 sec window with 50% overlap.
Both approaches resulted in reasonable reconstructions of the position and the velocity. The position shows the best correlation coefficient with the true trajectory. This result may be due to the fact that the velocity and the acceleration were derived as differential variables, where the noise in the estimation might be magnified. It is interesting to first notice that Kalman filter gets pretty good results on training set while the performance drop much more in testing comparing to the decrease between training and test. In the Kalman filter for BMI, the firing rates obtained by binning techniques blur the exact time information of the spike trains. The binning techniques also serve as averaging which makes the noise terms more Gaussian. This may make the Kalman filter over-fit the training set, while loose the generality in test because of a lack of information in spike timing. Comparing the performance difference between training and testing by Kalman PP, it shows no phenomenon of model over-fitting even with position $y$ better predicted in testing than in training. This is because Kalman works directly in the spike train, which involves higher resolution time information of neural activity. However, working in spike domain without averaging makes the assumption of the Gaussian distributed posterior density less satisfied in Kalman PP than Kalman. This is why the Kalman PP doesn’t show better decoding performance, which does not necessarily mean point process brings less information for decoding. When we compare the Kalman performance to Monte Carlo SE for point process in Table 4-10, where we have no Gaussian assumption, Monte Carlo SE has better decoding results in position and velocity as we expected. The smaller CC of reconstructed acceleration in point process might be due to large peaks of the desired acceleration as we explained before, which is different from the desired acceleration of Kalman filter.

**Monte Carlo SE Decoding in Spike Domain Using a Neural Subset**

The performance of BMI hinges on the ability to exploit information in chronically recorded neuronal activity. Since during the surgical phase there are no precise techniques to
target the modulated cells, the strategy has been to sample as many cells as possible from multiple cortical areas with known motor associations. In the experiment, we collected activities of 185 neurons from 5 motor cortical areas and regard them contributing equally to the current decoding process. Research has shown that different motor cortical areas play different roles in terms of the movement plan and execution. Moreover, the time-consuming computation on all the neuron information would bring significant computational burden to implement BMI in low-power, portable hardware. We can’t help making a guess that groups of neurons have different importance in BMI decoding as suggested in previous work [Sanchez et al., 2003]. In Chapter 3, we have shown that the information theoretical analysis on the neuron tuning function could be a criterion to evaluate the information amount between the kinematics and neural spike trains, therefore it weights the importance among neurons in term of certain task or movement. Moreover, if the decoding algorithm only calculates the subset of the important neuron associated with movement behavior, it will improve the efficiency of BMI on large amount of the brain activity data.

Neural Subset Selection

As we have shown in Chapter 3, the information theoretic tuning depth we proposed as a metric for evaluating neuron instantaneous receptive properties is based on information theory and would capture much more of the neuronal response. Define a tuned cell as a cell that extracts more information between the kinematics and its spiking output. The well-established concept of mutual information [Reza 1994] can mathematically account as an information theoretic metric from the neural spikes for each neuron based on the instantaneous tuning model, which is given by

$$I^{(j)}(spk^{(j)}, \tilde{k} \cdot \tilde{x}_{lag}) = \sum_{k \cdot x} p(\tilde{k} \cdot \tilde{x}_{lag}) \sum_{spk^{(j)}=0,1} p(spk^{(j)} | \tilde{k} \cdot \tilde{x}_{lag}) \cdot \log_2 \left( \frac{p(spk^{(j)} | \tilde{k} \cdot \tilde{x}_{lag})}{p(spk^{(j)})} \right) \quad (4-18)$$
where \( j \) is the neuron index. \( p(\bar{k} \cdot \bar{x}_{lag}) \) is the probabilistic density the linear filtered kinematics evaluated at the optimum time lag, which can be easily estimated by Parzen window [Parzen 1962]. \( p(spk) \) can be calculated simply as the percentage of the spike count during the entire spike train. \( p(spk | \bar{k} \cdot \bar{x}) \) is exactly the nonlinear function \( f \) in LNP model

The information theoretical tuning depth statistically indicates the information between the kinematic direction and neural spike train. By setting a threshold, as shown in Figure 4-10, it could help determine which subsets of the tuned neuron to include in the model to reduce the computation complexity. For example, the first 30 most tuned neurons could be selected as candidates to decode the movements in the BMI model. The distribution of the selected neuron is shown in Figure 4-11. Here the 5 different cortical areas are shown in different color bar with the corresponding mutual information estimated by Equation 4-18. The selected 30 neurons are labeled as read stars. We can see that here are 1 neuron in PMd-contra, 21 neurons in M1-contra, 6 neurons in S1-contra, and 2 neurons in SMA-contra. The most tuned neurons are in M1 as we expected.

**Neural Subset vs. Full Ensemble**

As we have the criterion to select the neural subset, we compare the reconstruction performance by different neural subsets, which have 60, 40, 30, 20, 10 most important neurons associated with the movement, to the decoding results by the full ensemble of 185 neurons. The statistical performances evaluated by both CC and NMSE with an 8 sec window with 50% overlap are shown in Table 4-11.

We also plot the statistical decoding performance (mean and standard derivation) by CC and NMSE respectively with different neuron subsets in Figure 4-12 A and 4-12 B. The performance difference evaluated by CC among the neural subsets is not as clear as the one by
NMSE. Decoding performances along $x$ evaluated by CC and NMSE increase and converge as the neuron number of neuron subset increases. The decoding performances along $y$ reach the maximum (CC) or the minimum (NMSE) when the neuron subset has 30 neurons.

The study of decoding performance by neuron subset shows the possibility to evaluate the neuron tuning importance associated with the movement task. With only 30 neurons (bolded row in Table 4-11) out of the full ensemble 185 (italic row in Table 4-11), we could achieve similar or even better performance in terms of NMSE. It means that not all the neuron activities in motor cortex are closely related to the movement task. Some of the neurons’ activities might contribute as noise in term of certain task, which reduces the decoding performance. At the same time, computation by only 30 neurons saves 84% running time comparing to the one by 185 neurons.
Figure 4-1. Schematic of relationship between encoding and decoding processes for Monte Carlo sequential estimation of point processes

Table 4-1. The kinematics reconstructions by Monte Carlo SE for segment of test data

<table>
<thead>
<tr>
<th>Method</th>
<th>Criterion</th>
<th>Position</th>
<th></th>
<th>Velocity</th>
<th></th>
<th>Acceleration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>Monte</td>
<td>Mean ± Std</td>
<td>0.81 ± 0.01</td>
<td>0.83 ± 0.01</td>
<td>0.79 ± 0.01</td>
<td>0.74 ± 0.01</td>
<td>0.45 ± 0.01</td>
<td>0.25 ± 0.01</td>
</tr>
<tr>
<td>Carlo SE</td>
<td>Best</td>
<td>0.83</td>
<td>0.84</td>
<td>0.80</td>
<td>0.74</td>
<td>0.47</td>
<td>0.26</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>0.79</td>
<td>0.83</td>
<td>0.78</td>
<td>0.73</td>
<td>0.44</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>Mean ± Std</td>
<td>0.44 ± 0.03</td>
<td>0.98 ± 0.14</td>
<td>0.45 ± 0.02</td>
<td>0.55 ± 0.01</td>
<td>0.82 ± 0.01</td>
<td>1.03 ± 0.01</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>0.40</td>
<td>0.74</td>
<td>0.45</td>
<td>0.54</td>
<td>0.81</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>0.43</td>
<td>1.30</td>
<td>0.44</td>
<td>0.54</td>
<td>0.81</td>
<td>1.02</td>
</tr>
</tbody>
</table>
Figure 4-2. The posterior density of the reconstructed kinematics by Monte Carlo SE
Figure 4-3. The reconstructed kinematics for 2-D reaching task

Table 4-2. Averaged performance by Monte Carlo SE of the kinematics reconstructions for segment of test data

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
</tr>
<tr>
<td>CC</td>
<td>0.81</td>
<td>0.84</td>
<td>0.80</td>
</tr>
<tr>
<td>NMSE</td>
<td>0.43</td>
<td>0.93</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Table 4-3. Statistical performance of the kinematics reconstructions using 2 criteria

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
</tr>
<tr>
<td>CC</td>
<td>$0.762</td>
<td>$0.757$</td>
<td>$0.751$</td>
</tr>
<tr>
<td></td>
<td>$\pm 0.078$</td>
<td>$\pm 0.128$</td>
<td>$\pm 0.075$</td>
</tr>
<tr>
<td>NMSE</td>
<td>0.563</td>
<td>0.964</td>
<td>0.515</td>
</tr>
<tr>
<td></td>
<td>$\pm 0.186$</td>
<td>$\pm 0.322$</td>
<td>$\pm 0.126$</td>
</tr>
</tbody>
</table>
Figure 4-4. Linear model error using different $\alpha$
Figure 4-5. *cdf* of noise distribution using different density

Figure 4-6. Nonlinearity of neuron 72 using different $\sigma$
Figure 4-7. Decoding performances by different $x_n$

Table 4-4. Results comparing the kinematics reconstructions averaged among Monte Carlo trials and synthetic averaging

<table>
<thead>
<tr>
<th>Criteri on</th>
<th>Method</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
</tr>
<tr>
<td>CC</td>
<td>Average among 20 Monte Carlo trials</td>
<td>0.811</td>
<td>0.837</td>
<td>0.799</td>
</tr>
<tr>
<td></td>
<td>Average among 20 Synthetic spikes, 20 Monte Carlo trials each</td>
<td>0.843</td>
<td>0.852</td>
<td>0.822</td>
</tr>
<tr>
<td>NMSE</td>
<td>Average among 20 Monte Carlo trials</td>
<td>0.429</td>
<td>0.933</td>
<td>0.439</td>
</tr>
<tr>
<td></td>
<td>Average among 20 Synthetic spikes, 20 Monte Carlo trials each</td>
<td>0.319</td>
<td>0.768</td>
<td>0.330</td>
</tr>
</tbody>
</table>
Table 4-5. Statistical performance of the kinematics reconstructions by Monte Carlo SE and synthetic averaging

<table>
<thead>
<tr>
<th>Method</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>Monte Carlo SE</td>
<td>0.762</td>
<td>±0.078</td>
<td>0.751</td>
</tr>
<tr>
<td>(synthetic averaging)</td>
<td>±0.128</td>
<td>±0.063</td>
<td>±0.154</td>
</tr>
<tr>
<td>t-test</td>
<td>H₁: CC&lt;CCₐ</td>
<td>p-value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1(0.027)</td>
<td>0(0.618)</td>
<td>1(0.004)</td>
</tr>
<tr>
<td>Monte Carlo SE</td>
<td>0.563</td>
<td>±0.186</td>
<td>±0.515</td>
</tr>
<tr>
<td>(synthetic averaging)</td>
<td>±0.322</td>
<td>±0.126</td>
<td>±0.171</td>
</tr>
<tr>
<td>t-test</td>
<td>H₁: NMSE&gt;NMSEₐ</td>
<td>p-value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1(0)</td>
<td>1(0)</td>
<td>1(0)</td>
</tr>
</tbody>
</table>

Table 4-6. Results comparing the kinematics reconstruction by Kalman PP and Monte Carlo SE for a segment of data

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Method</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>CC</td>
<td>Kalman filter PP</td>
<td>0.74</td>
<td>0.83</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>Monte Carlo SE</td>
<td>0.81</td>
<td>0.84</td>
<td>0.80</td>
</tr>
<tr>
<td>NMSE</td>
<td>Kalman filter PP</td>
<td>0.81</td>
<td>1.51</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>Monte Carlo SE</td>
<td>0.43</td>
<td>0.93</td>
<td>0.44</td>
</tr>
</tbody>
</table>
Figure 4-8. The reconstructed kinematics for a 2-D reaching task. A) Plot individually. B) Position reconstruction in 2D
Figure 4-8. Continued
### Table 4-7. Statistical performance of the kinematics reconstructions by Kalman PP and Monte Carlo SE (synthetic averaging)

<table>
<thead>
<tr>
<th>Method</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
</tr>
<tr>
<td><strong>CC</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kalman filter PP</td>
<td>0.763 ± 0.073</td>
<td>0.717 ± 0.133</td>
<td>0.702 ± 0.114</td>
</tr>
<tr>
<td>Monte Carlo SE (synthetic averaging)</td>
<td>0.777 ± 0.089</td>
<td>0.755 ± 0.154</td>
<td>0.753 ± 0.083</td>
</tr>
<tr>
<td><strong>t-test</strong> $H_1$: $CC_{Kalman} &lt; CC_{MCSE}$ ($p$-value)</td>
<td>0(0.172)</td>
<td>1(0.028)</td>
<td>1(0.001)</td>
</tr>
<tr>
<td><strong>NMS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kalman filter PP</td>
<td>± 0.305 ± 0.245</td>
<td>± 0.271 ± 0.172</td>
<td>± 0.187 ± 0.385</td>
</tr>
<tr>
<td>Monte Carlo SE (synthetic averaging)</td>
<td>± 0.171</td>
<td>± 0.321</td>
<td>± 0.127</td>
</tr>
<tr>
<td><strong>t-test</strong> $H_1$: $NMSE_{Kalman} &gt; NMSE_{MCSE}$ ($p$-value)</td>
<td>1(0)</td>
<td>1(0)</td>
<td>1(0)</td>
</tr>
</tbody>
</table>

### Table 4-8. Statistical performance of the kinematics reconstructions by different encoding models

<table>
<thead>
<tr>
<th>Method</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
</tr>
<tr>
<td><strong>NMS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exponential PP</td>
<td>0.6673 ± 0.2024</td>
<td>1.4976 ± 0.6547</td>
<td>0.6690 ± 0.2090</td>
</tr>
<tr>
<td>Kalman filter PP</td>
<td>0.897 ± 0.097</td>
<td>1.043 ± 0.637</td>
<td>0.673 ± 0.086</td>
</tr>
<tr>
<td><strong>t-test</strong> $H_1$: $NMSE_{Kalman} &gt; NMSE_{MCSE}$ ($p$-value)</td>
<td>1(0)</td>
<td>1(0)</td>
<td>1(0)</td>
</tr>
</tbody>
</table>
Figure 4-9. The decoding performance by algorithms in PP for different data sets. A) CC. B) NMSE
Figure 4-9. Continued.
Table 4-9. Statistical performance of the kinematics reconstructions Kalman filter and Kalman PP

<table>
<thead>
<tr>
<th>Method</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>Kalman filter</td>
<td>Training</td>
<td>± 0.039</td>
<td>± 0.061</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>± 0.070</td>
<td>± 0.100</td>
</tr>
<tr>
<td>CC</td>
<td>Training</td>
<td>± 0.061</td>
<td>± 0.182</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>± 0.073</td>
<td>± 0.133</td>
</tr>
<tr>
<td>Kalman filter PP</td>
<td>Training</td>
<td>± 0.061</td>
<td>± 0.182</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>± 0.073</td>
<td>± 0.133</td>
</tr>
</tbody>
</table>

Table 4-10. Statistical performance of the kinematics reconstructions by spike pates and by point process

<table>
<thead>
<tr>
<th>Method</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>Kalman filter</td>
<td>± 0.0703</td>
<td>± 0.1003</td>
<td>± 0.0601</td>
</tr>
<tr>
<td>Monte Carlo</td>
<td>0.7776</td>
<td>0.7545</td>
<td>0.7530</td>
</tr>
<tr>
<td>SE</td>
<td>± 0.0886</td>
<td>± 0.1543</td>
<td>± 0.0830</td>
</tr>
</tbody>
</table>

Figure 4-10. Threshold setting for sorted information theoretic tuning depths for 185 neurons
Figure 4-11. Selected neuron subset (30 neurons) distribution
Table 4-11. Statistical performance of the kinematics reconstructions by neuron subset and full ensemble

<table>
<thead>
<tr>
<th>Neuron Subset</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td><strong>Full Ensemble</strong></td>
<td>0.7619</td>
<td>0.7574</td>
<td>0.7511</td>
</tr>
<tr>
<td></td>
<td>±0.0784</td>
<td>±0.1275</td>
<td>±0.0749</td>
</tr>
<tr>
<td>60</td>
<td>0.7554</td>
<td>0.7721</td>
<td>0.7473</td>
</tr>
<tr>
<td></td>
<td>±0.0942</td>
<td>±0.1105</td>
<td>±0.0787</td>
</tr>
<tr>
<td>40</td>
<td>0.7449</td>
<td>0.7782</td>
<td>0.7373</td>
</tr>
<tr>
<td></td>
<td>±0.0954</td>
<td>±0.1058</td>
<td>±0.0848</td>
</tr>
<tr>
<td>CC</td>
<td>0.7456</td>
<td>0.7730</td>
<td>0.7420</td>
</tr>
<tr>
<td></td>
<td>±0.1027</td>
<td>±0.1084</td>
<td>±0.0823</td>
</tr>
<tr>
<td>30</td>
<td>0.7213</td>
<td>0.7568</td>
<td>0.7227</td>
</tr>
<tr>
<td></td>
<td>±0.1036</td>
<td>±0.1238</td>
<td>±0.0884</td>
</tr>
<tr>
<td>20</td>
<td>0.7181</td>
<td>0.6487</td>
<td>0.6824</td>
</tr>
<tr>
<td></td>
<td>±0.1141</td>
<td>±0.1752</td>
<td>±0.0924</td>
</tr>
<tr>
<td><strong>Full Ensemble</strong></td>
<td>0.5628</td>
<td>0.9643</td>
<td>0.5145</td>
</tr>
<tr>
<td></td>
<td>±0.1861</td>
<td>±0.3222</td>
<td>±0.1259</td>
</tr>
<tr>
<td>60</td>
<td>0.5330</td>
<td>0.8925</td>
<td>0.5031</td>
</tr>
<tr>
<td></td>
<td>±0.1908</td>
<td>±0.2256</td>
<td>±0.1345</td>
</tr>
<tr>
<td>40</td>
<td>0.5335</td>
<td>0.8003</td>
<td>0.5173</td>
</tr>
<tr>
<td></td>
<td>±0.1818</td>
<td>±0.1585</td>
<td>±0.1385</td>
</tr>
<tr>
<td><strong>Full Ensemble</strong></td>
<td>0.5339</td>
<td>0.8022</td>
<td>0.4985</td>
</tr>
<tr>
<td></td>
<td>±0.2047</td>
<td>±0.2555</td>
<td>±0.1440</td>
</tr>
<tr>
<td>30</td>
<td>0.5828</td>
<td>0.7273</td>
<td>0.5334</td>
</tr>
<tr>
<td></td>
<td>±0.1858</td>
<td>±0.2674</td>
<td>±0.1538</td>
</tr>
<tr>
<td>20</td>
<td>0.5770</td>
<td>0.7304</td>
<td>0.5697</td>
</tr>
<tr>
<td></td>
<td>±0.2167</td>
<td>±0.3408</td>
<td>±0.1550</td>
</tr>
<tr>
<td>10</td>
<td>0.5628</td>
<td>0.9643</td>
<td>0.5145</td>
</tr>
<tr>
<td></td>
<td>±0.1861</td>
<td>±0.3222</td>
<td>±0.1259</td>
</tr>
</tbody>
</table>
Figure 4-12. Statistical performance of reconstructed kinematics by different neuron subsets. A) CC. B) NMSE
Figure 4-12. Continued
Conclusions

Brain-Machine Interfaces (BMI) is an emerging field inspired by the need to restore motor function and control in individuals who have lost the ability to control the movement of their limbs. Researchers seek to design a neuron-motor system that exploits the spatial and temporal structure of neural activity in the brain to bypass spinal cord lesions and directly control a prosthetic device by intended movement. In human and animal experiments, neuronal activity has been collected synchronously from microelectrode arrays implanted into multiple cortical areas while subjects performed 3-D or 2-D target-tracking tasks. Several signal processing approaches have been applied to extract the functional relationship between the neural recordings and the animal’s kinematic trajectories. The resulting models can predict movements and control a prosthetic robot arm or computer to implement them.

Many decoding methodologies, including Wiener filter and neural networks, use binned spike trains to predict movement based on standard linear or nonlinear regression. Alternative methodologies, such as Kalman filter or particle filter, were derived using a state model within a Bayesian formulation. From a sequence of noisy observations of neural activity, the probabilistic approach analyzes and infers the kinematics as a state variable of the neural dynamical system. The neural tuning property relates the measurement of the noisy neural activity to the animal’s behaviors, and builds up the observation measurement model. Consequently, a recursive algorithm based on all available statistical information can be used to construct the posterior probability density function of each kinematic state given the neuron activity at each time step from the prior density of that state. The prior density in turn is the posterior density of the previous time step updated with the discrepancy between an observation model and the neuron
firings. Movements are then recovered probabilistically from the multi-channel neural recordings by estimating the expectation of the posterior density or by maximum a posterior. The differences among the above approaches reflect the following challenges in BMI modeling.

**Linear or nonlinear?** Wiener and Kalman filters are both linear fitting methods that can be used to reflect the functional relationship between neural firing and movements. A linear model is intuitive and is not computationally complex, so is simple to calculate. However, the assumption of linearity is very strict, and although it may be valid for binned data due to the averaging effects, most neuroscientists do not agree with this approach at the neural level. Adding to this concern is that neuron behavior exhibits saturation, thresholding, and refractory attributes, thus reflecting nonlinearity. To improve the performance of these models, neural networks and particle filters were added to build nonlinear relationships, but this also increases the computational complexity. On the other hand, a standard method to accurately estimate or model the neural nonlinearity is still in development, since the ground truth is not fully understood even by neuroscientists. Evaluation of the model nonlinearity by comparing several algorithm performances in the BMI reconstruction accuracy is one of the feasible ways to rate different hypothesis. At issue is whether or not the performance will improve enough to justify the complicated nonlinear modeling and computation.

**Gaussian or non-Gaussian?** Gaussianity is one of engineering’s most preferred assumptions to describe the error distribution when we build models for stochastic signals. In the Bayesian approach, the assumption of Gaussianity is also present in the Kalman filter to describe the posterior density. However, if we agree on the nonlinearity relation of the neuron behavior tuning to preferred movement, the Gaussian assumption at all the time is always questionable because the pdf is reshaped by the nonlinear tuning. An algorithm that is not bound to this
assumption (i.e., which utilizes the full information in the pdf) is necessary to help us understand how much performance hit is tied to the Gaussian assumption. Particle filtering is a general sequential estimation method that works with continuous observation through a nonlinear observation model without the Gaussian assumptions. However, in terms of a practical application, we should not over-claim the non-Gaussian assumption for performance evaluation because the computational complexity of both methods (Particle and Kalman filters) is drastically different. The proper framework is to realize that the Gaussian assumption is a simplifying assumption, and then ask how much improvement over the Kalman can the Particle filter provide. For instance, if the local pdf can be approximated by a Gaussian distribution very well for certain segment of experimental data, the algorithm with non-Gaussian assumption would have the equivalent performance without showing its advantage, which, at the same time, comes with more computation complexity.

**Black box or gray box?** Wiener and neural networks are black box models that operate without physical insight into the important features of the motor nervous system. However, in the Bayesian approach, the observation model enables us to have more insight into the neural tuning property which relates the measurement of the noisy neural activity to the animal’s behaviors. Although the Kalman filter is still (and controversially) linear, it would be an excellent entry point to incorporate the knowledge of neural tuning into modeling. Enhancing the black box model to the gray box model is expected to increase performance and in turn to test the knowledge we incorporate into the model. Notice however, that both the particle filtering and the Kalman filter both still assume a fixed and known state and observation models. In actuality, these remain unknown for BMI data.
All of the computational models described above are intended to efficiently and accurately translate neural activity into the intention of movement. Depending on different animals and tasks, well-established adaptive signal processing algorithms have achieved reasonable kinematics predictions (average correlation coefficient of around 0.8 [Sanchez, 2004]). These algorithms provide an attractive engineering solution for evaluating and characterizing the temporal aspects of a system. However, a successful realization of BMI cannot be dependent entirely on improvement of methodologies. We must develop a better understanding of brain signal properties. Brain signals are believed to be very complicated. They contain a huge amount of data, they are noisy, non-stationary, and interact with each other in ways not fully understood.

When designing the computational model, the following should be carefully considered.

**What is the proper signal scale for BMIs?** To fit into the traditional signal processing algorithm which works with a continuous value, early BMI research frequently employed a binning process on action potentials to obtain the neural firing rate as a continuous neural signal. However, single unit activity is completely specified by the spike times. The weakness of the binning technique, as a coarse approach, is finding the optimal window size. The loss of spike timing resolution might exclude rich neural dynamics from the model. How to extract effectively the information hidden in the spike timing brings challenges not only in signal-processing algorithm development but also in the accurate modeling of the neuron physiologic properties. Moreover, if the signal-processing techniques enable us to look closer into the neural spike train, we will have to face another challenge not encountered in BMI for spike rates.

**Time resolution gap between neural activity and movement.** Although spike trains are a very good indicator of neuronal function, they are also far removed from the time and macroscopic scales of behavior. Therefore, a central question in modeling brain function in
behavior experiments is how to optimally bridge the time scale between spike events (milliseconds) and the time scale of behavior (seconds). Most often, the relatively rudimentary method of time averaging (binning spikes) is used to bridge the gap, but much of the resolution of the spike representation is wasted. Therefore, to model the hierarchy of scales present in the nervous system, a model-based methodology must link the firing times to movement in a principled way. It remains to be seen under what conditions the spike timing is relevant for motor BMIs because as stated the kinematics exist at a much longer time scale, which may indicate that the exact timing of spikes is not important.

**Non-stationary neuron behavior.** Studies show that the response of individual neurons to the same stimulus changes frequently. Even the cortical areas used in BMI experiments can vary considerably from day to day. Neuroscientists have used the average of peri-event neuron spiking patterns across trials/times in order to eliminate noise contamination and observe the same stationary neuron behaviors. However, this statistical analysis is not feasible for the reconstruction of trajectory time series in motor BMI. Current signal processing modeling still assumes that neuron behaviors are stationary between the training and testing data. This assumption is questionable and affects the performance on the test data.

**Association among the neurons.** Evidence shows that neuron spikes are synchronized as groups along time. Some researchers even claimed that in order to understand brain function, the signals should be recorded from areas all over the brain since they are dynamically correlated as a network. Imagine the computational complexity when we have about 200 neurons interacting with each other. Researchers have applied statistics and data mining techniques to evaluate the synchronization of multi-channel spikes in terms of the accuracy and the efficiency. A better understanding of neuron recordings, especially the causal correlation between the different
recording areas, would be achieved by dynamic modeling in the probability domain neural
dependence across channels. Therefore, it is very important to involve a dependence study
among the neurons into our BMI study. Unfortunately the sequential estimation models for point
processes assume independence among neurons to avoid estimating the joint distribution, so this
is one of their most important shortcomings.

Computational complexity. BMI performance hinges on the ability to exploit information
in chronically recorded neuronal activity. Since there are no precise techniques to target the
modulated cells during the surgical phase, the strategy has been to sample as many cells as
possible from multiple cortical areas with known motor associations. This time-consuming
computational burden would significantly impair the use of BMI in low-power, portable
hardware. Therefore channel selection methodologies should be applied to the neural vector to
estimate the channels that are more relevant for the task.

With all of these issues in mind, we proposed and validated a Monte Carlo sequential
estimation framework to reconstruct the kinematics directly from the neural spike trains. There
are two main steps to apply this idea to neural data from BMI experiments. First, we must
validate our physiologic knowledge of neural tuning properties by analysis and modeling using
statistical signal processing. Second, based on the knowledge we have gained, we must
implement the adaptive signal filtering algorithm to derive the kinematics directly from the
neuron spike trains.

Our intention is to reduce the randomness of the neuron spiking in probabilistic models.
Faced with a tremendous amount of neural recoding data, we proposed using the mutual
information between the neuron spike and kinematic direction as a new metric to evaluate how
much information the neuron spike encodes. This well-established concept in information theory
provides a statistical measure to gauge neuron tuning depth. As a non-unit measure, the proposed metric provides a means to compare information in terms of tuning, not only among different kinematics, positions, velocities and accelerations; but also among neurons in different cortical areas. The primary motor cortex contained most of the tuned neurons, and therefore is a potential location to elicit a neuron subset for movement reconstructions.

In addition to its informative value for importance, the tuning function was also mathematically estimated by a parametric Linear-Nonlinear-Poisson model. The traditional criterion of estimating tuning depth from windows of data does not seem the most appropriate in the design of BMIs using sequential estimation algorithms on spike trains. Here we presented instead an information theoretical tuning analysis of instantaneous neural encoding properties that relate the instantaneous value of the kinematic vector to neural spiking. The proposed methodology is still based on the Linear-Nonlinear-Poisson model of Paninski. Using a spike-triggered averaging technique, the linear filter finds the preferred direction of a high-dimensional kinematics vector, which could involve both spatial (2-D) and temporal information if evaluated in a window. The nonlinear filter demonstrates the neuron’s nonlinear property, such as saturation, thresholding, or refractory period. As the function of the filtered kinematic vectors, the neuron’s nonlinear property is approximated by the conditional probability density of the spikes according to the Bayesian rule. Although most of the statistical nonlinear neuron properties are expressed as exponentially increasing curves, we also found diversity among these properties. This might indicate varying functional tuning roles among neurons. The prescribed inhomogeneous model embodies the randomness and nonstationary aspects of neural behaviors, which finally connects the continuous kinematics to the point process. An information theoretic formulation provides a more detailed perspective when compared with the conventional tuning
curve because it statistically quantifies the amount of information between the kinematic vectors triggered off by the spike train. As a direct consequence, it can estimate the optimum time delay between motor cortex neurons and behavior caused by the propagation effects of signals in the motor and peripheral nervous system.

The similarities and differences between the windowed and instantaneously evaluated tuning functions were also analyzed. The instantaneous tuning function displayed over 0.9 correlation in the central region w.r.t. to the windowed tuning function. The differences in the high tuning region of the curves, both in the dynamic range and in the estimated value were much higher and resulted from the overestimation of tuning by the window method at the high firing rate part of the curve. The instantaneous model works directly in the dynamic range of the kinematics therefore it estimates more accurately the firing probability without distortions from the temporal neighborhood information and produce equivalent or better encoding results compared to existing techniques. This outcome builds confidence to directly implement the instantaneous tuning function into the future online decoding models for Brain-Machine Interfaces.

The instantaneous tuning function based on the Linear-Nonlinear-Poisson model builds a non-linear functional relationship from the kinematics to the neuron activity, which is estimating neural physiologic tuning directly from the spike timing information. This solution is working to a certain extent, but it might not describe totally how the neuron actually fires corresponding to the certain kinematics. For example, it assumes a stationary linear filter and nonlinear tuning curve; the current modeling is done independently for each neuron without considering the interactions. Since the accuracy of the encoding model will impact the performance of the
kinematic decoding from the neural activity, further development and validation of the encoding model is an important aspect to consider.

With the knowledge gained from the neuron physiology function analysis with this signal processing algorithm, we proposed a Monte-Carlo sequential estimation for point process (PP) adaptive filtering to convert the Brain Machine Interfaces decoding problem to state sequential estimation. We reconstruct the kinematics as the state directly from the neural spike trains. The traditional adaptive filtering algorithms were well established to represent the temporal evolution of a system with continuous measurements on signals, such as Kalman filter, least square solution and gradient decent searching. They are of limited use when it comes to BMI decoding in the spike domain, where only the recorded neural spiking time matters and the amplitude information of the signals is absent. A recently proposed point process adaptive filtering algorithm uses the probability of a spike occurrence (which is a continuous variable) and the Chapman-Kolmogorov Equation to estimate parameters from discrete observed events. As a two-step Bayesian approach, it assumes posterior density of the state given the observation Gaussian distrusted with less accuracy. We presented a Monte Carlo sequential estimation to modify the amplitude of the observed discrete events by the probabilistic measurement – posterior density. We generated a sequence of samples to estimate the posterior density more precisely, avoiding the numerical computation of the integral in the C-K Equation through sequential estimation and weighted Parzen windowing. Due to the smoothing of the posterior density with the Gaussian kernel from Parzen windowing, we used collapse to easily obtain the expectation of the posterior density, which leads to a better result of state estimate than noisy Maximum A Posterior. In a simulation of a one-neuron encoding experiment, the Monte Carlo estimation showed better
capability to probabilistically estimate the state, better approximating posterior density than the point process adaptive filtering algorithm with Gaussian assumption.

The Monte Carlo sequential estimation PP algorithm enables us to use signal-processing techniques to directly draw information from timing of discrete event without a Gaussian assumption. Although it is proposed for the BMI application on motor cortical neurons in this dissertation, it is theoretically a general non-parametric approach that can infer continuous signals from point process without constraints, which can be utilized in many other neuroscience applications (e.g. visual cortex processing), in communications (network traffic) and in process optimization. We have to point out that the implementation of this algorithm would not always bring us better performance. It depends how the user assign the proper state and build the models. In addition, the advantage of the approach will be only shown when the posterior density of the state given observation can’t be well approximated by Gaussian distribution, for example, multi-modes or highly skewed. On the other hand, since the pdf information is fully stored and propagated for each time index, the computation complexity is one trade off that the user must weight. Moreover, we were able to pin point and quantify for motor BMIs the performance paid by the Gaussian assumption. Towards this goal, we compared performance with the Kalman filter PP applied to a cursor control task, and concluded that the Monte Carlo PP framework showed statistically better results (all the p value of the pair-wise t-test on NMSE is smaller than 0.02) between the desired and estimated trajectory. We should mention that this improvement in performance is paid by much more demanding computation and also by much more detailed information about the decoding model for each neuron.

Although spike trains are very telling of neuronal function, they are also very removed from the macroscopic time scales of behavior. Therefore, a central question in modeling brain
function in behavior experiments is how to optimally bridge the time scale between spike events (milliseconds) and the time scale of behavior (seconds). Most often, the relatively rudimentary method of time averaging (binning spikes) is used to bridge the gap, but excludes the rich information embedded in the high resolution of the spike representation. Model-based methodologies including an encoding model linking the firing times to state variables as the ones presented here seem to be a much more principled way to model the hierarchy of scales present in the nervous system. However, these models are intrinsically stochastic with the encoding models in use today, so they pose difficulties for real time operation of BMI models.

Although the results are interesting, the signal processing methodologies for spike train modeling need to be further developed. Many parameters are assumed and need to be estimated with significant design expertise as we studied in terms of decoding performance. They are substantially more complex than the ones for random processes. Therefore, we choose the averaged kinematics estimation among many Monte Carlo trials as the algorithm performance. Still, the results are still intrinsically stochastic due to the randomness of the generated spike trains. In order to achieve more reliable results, we propose a synthetic averaging idea to generate several sets of spike trains from the estimated firing intensity probability to simulate the population effects in the cortex. Instead of the coarse binning techniques on the neural activity, the model is implemented several times from regenerated spike observation to reconstruction the kinematics. The performance is averaged among the decoding results in the movement domain to bypass the possible distortion by nonlinear tuning function due to the binning in spike domain. The synthetic averaging idea provided smoother kinematics reconstruction, which is a promising result for improved performance.
However, synthetic averaging is effectively averaging the timing information that one seeks in this class of methods in the first place. Therefore, the interesting observation is that it seems to indicate that spike timing has no effect in performance, otherwise when we use the synthetic examples performance should decrease. This issue is hard to quantify due to the many factors at play and the lack of ground truth to compare absolute performance. We briefly explain the issues below, but this is an open problem that deserves much more research.

First, the way we generate the synthetic spike trains is to obtain an estimate of the intensity function (firing probability) of a single neuron by kernel smoothing. This obviously will always produce a bias estimate of the intensity function that will be present in all the realizations. However, the averaging of kinematic responses will decrease the variance of the estimated kinematics as we have seen in the results. NMSE reduces 26% for position along $x$, 18% for position along $y$, and on average 15% for all 6 kinematic variables. But this process of averaging effectively puts us back into the realm of rate models if we look at the input side (spike trains). We think that further analysis is necessary distinguishing the linear and the nonlinear models. If we do synthetic averaging in the Kalman PP, where the neuron tuning function is linear, the synthetic averaging would be indeed equivalent as inputting the continuous firing rates when the number of realizations is infinite. However, since the neuron tuning function is developed based on LNP model, the averaging in the neuron activity, binning or smoothed spike rates, would be conceptually different from the averaging on the nonlinear outputs of the tuning. As a simple example, in general $f(E[x]) \neq E[f(x)]$, where $E[\bullet]$ is expectation operation. Besides, synthetic averaging is coupled with LNP encoding model designed just for the spike train, which models the kinematics simply triggered by the spike timing. This quantity can not be currently estimated.
on the continuous firing rates inputs since there is no corresponding encoding modeling method available.

The synthetic averaging is an attempt not only to bridge the time-resolution difference between neuron activity and the kinematics, but also to reduce the variance of the spike timing introduced by single realization of the neuron recordings. Alternate methods that can reduce the variance of the estimate without reducing temporal resolution need to be investigated, but are not known to us at this moment.

In addition to the comparing our Monte Carlo SE to Kalman PP to evaluate effect of the linear/nonlinear tuning and Gaussian/nonGaussian distributed posterior density; we further investigated the decoding performance by comparison to other decoding methods. The difference between the statistical reconstruction results by Kalman PP and adaptive filtering on point process with exponential tuning function shows the importance of accurate encoding model. The linear tuning curve works better for kinematics along $y$ (e.g. NMSE of position $y$ linear vs. exponential: $1.043 \pm 0.245$ vs. $1.498 \pm 0.655$). While the exponential tuning curves works better for kinematics along $x$ (e.g. NMSE of position $x$ exponential vs. linear: $0.667 \pm 0.202$ vs. $0.897 \pm 0.305$). However, both of the encoding models couldn’t catch more information than Monte Carlo SE, which provides the best decoding results (e.g. NMSE of position $x$ and $y$: $0.563 \pm 0.186$ vs. $0.964 \pm 0.322$). This is because the Monte Carlo SE using the instantaneous encoding estimated directly for data without close-form assumptions.

Let’s come back to the motivation of developing signal processing techniques on the point process, where we wonder if the spike timing contains richer information than the conventional spike rates. One straightforward way is to compare decoding performances between the spikes rates and point process domain. Since our algorithm is developed based on the state-observation
model, it would be comparable to first start with Kalman filter and Kalman PP. Both methods have linear tuning and assume the posterior density Gaussian distributed. The big performance drop between the training and test by Kalman filter shows the over-fit parameter in tuning model because of the blurred time information of the neural activity. Kalman PP works directly on point process, which overcomes the problem with less performance difference between training and testing set. However, the closer resolution on neural activity results in poor estimation on posterior density results approximated Gaussian, which produces not necessarily better results. Compare the performance by Monte Carlo SE, which estimated posterior density more accurately, the performance in spike domain is slightly better (CC of 2D position $0.7776 \pm 0.0886$, $0.7545 \pm 0.1543$) than the one in continuous spike rates (CC of 2D position $0.7463 \pm 0.0703$, $0.7397 \pm 0.1003$).

The slightly better performance is not as good as we expected to corroborate the hypothesis that richer dynamic information from spike timing is needed in motor BMIs. By only checking values of the performance criterion, it would be too quick to come to the conclusion that the spike trains contain no more information than spike rates. We should look into carefully how the 2 different methods are implemented and under what circumstances each shows the advantage. The Kalman filtering infers the kinematics from continuous spike rates within closed form simply and analytically with linear model and Gaussian assumption on posterior. Our proposed Monte Carlo sequential estimation enable to filter on point process, while it would show clearly better performance only if the pdf of the state given experimental observation is multi-modal or highly skewed for most of the time. One of the possible reasons of the slightly better performance here could be that state variable we are modeling on. Currently we build the probabilistic approach to inferring the 2D position, velocity and acceleration, which are a final
representation of a combination of complicated muscle movements that are initialized by the motor neuron spiking. Those combinations can be regarded as low-pass filtering or weighted averaging operations from the neural activities, which might make linear function and Gaussian assumptions easily satisfied in Kalman filter. Plus the bigger time resolution gap from spike timing rather than spike rates brings more difficult decoding job for Monte Carlo SE. If we would have access to synchronous EMG (electromyopgraphic), signals which have a much higher time resolution than the kinematics because they respond to motor neuron firing without too much averaging and less time resolution gap, it might be a better case for Monte Carlo sequential estimation to show its advantages on decoding.

Comparing to the Kalman filter with fixed linear model, our proposed approach, as a non-parametric method without constrains, enables us to build the neuron physiologic tuning knowledge estimated simply from spike timing into the decoding framework. The instantaneous LNP model we currently use may not be optimal, which could also result in the “slightly” but not “obviously” better performance. The better encoding model should bring the potentials to improve the BMI decoding performance, therefore evaluate more fairly if the spike timing contains more information comparing to the spike rates.

In the efforts to reduce the computational complexity for multi-channel BMI, we proposed mutual information based on the instantaneous tuning function to select the neuron subset in term of the importance related to the movement task. Among the 30 selected neurons, 70% of neurons distribute in M1. The decoding performance has close or even less NMSE comparing to the full neuron ensemble with much less computational complexity.

**Future Work**

As we have described the challenges in BMI, Monte Carlo SE is design to derive kinematics directly from spike domain without linear and Gaussian assumptions. The
instantaneous encoding model tries to evaluate the tuning property directly from the data without a closed form assumption such as linear or exponential. We have also developed the synthetic averaging idea in efforts to bridge the time gap between the neural activates and the movement. The information theoretical criterion is proposed to reduce the computation complexity by decoding with only subset of the neurons. There are still some aspects we could work on in the future 1) the association among neurons, and 2) the non-stationary tracking of the neuron tuning properties during decoding process.

In our current approach, the posterior density of the kinematics given multi-channel spike observations is obtained with the conditional independent assumption among all the neurons. This opposes the concerns on neuron associations. One solution might be to modify the neuron tuning function such that it takes into account not only the kinematics but also the neurons with synchronized behavior. In this way, we also build the functional structure between the neuron firing information and improve our approach in a more realistic way.

In our preliminary BMI decoding results, we used the statistically fixed tuning function to reconstruct the monkey’s movements from the multi-channel neuron spike trains. The preferred kinematic direction, which is represented by the linear filter in the tuning function model, is constant for each neuron. The nonlinearity of the neuron tuning curve remains constant throughout the decoding. As we analyzed the decoding performance in training and testing data in different segments, it clearly shows that the reconstruction in the testing segment, which is far away from the training set, is poor. It is because the stable assumptions could conflict with regard to nonstationary neuron firing patterns. If we can analyze the amount of information that a neuron conveys by firing changes, could we deal with it in the decoding?
Awareness of the nonstationary properties of neuron firing behaviors should alter the parameters in the tuning function modeling along the time step. The preferred kinematic direction could deviate slightly from the direction at the previous time iteration. Approximating both movements and linear filter weights is a duel estimation problem. In the dual extended Kalman filter [Wan & Nelson, 1997] and the joint extended Kalman filter [Matthews, 1990], the dual estimation problem was addressed with differing solutions. In the dual extended Kalman filter, a separate state-space representation is used for both the signal and the weights. At every time step, the current estimation of the weights is used as a fixed parameter in the signal filter, and vice versa. The joint extended Kalman Figure combines signal and weights into a single joint state vector, and runs the estimation simultaneously. Since there are 185 neurons recorded simultaneously with the movement task, to explore the joint state vector with both signal and weights within such a high dimensional space could require huge amount of samples. We apply here the dual methods to our BMI decoding to deal with the nonstationary neuron tuning function.

We started with the simplest case, Kalman filter working on the continuous binning spike rates to show preliminary results of the dual idea. To apply Kalman filter on our BMI data, the state dynamic remains the same as

\[ x_t = F_t x_{t-1} + \eta_t \]  \hspace{1cm} (5-1)

where \( F_t \) establishes the dependence on the previous state and \( \eta_t \) is zero-mean Gaussian distributed noise with covariance \( Q_{\eta_t} \). \( F_t \) is estimated from training data by the least square solution. \( Q_{\eta_t} \) is estimated as the variance of the error between the linear model output and the desired signal. The tuning function is linearly defined as

\[ \lambda_t = H_t \cdot \tilde{x}_{t+\text{lag}} + n_{\lambda_t} + n_{2t} \] \hspace{1cm} (5-2)
where $\lambda_t$ is the firing rate by 100ms window binning. $\tilde{x}_t$ is the instantaneous kinematics vector defined as $[\tilde{p}_x \tilde{v}_x \tilde{a}_x \tilde{p}_y \tilde{v}_y \tilde{a}_y]_t^T$ with 2-dimensional information of position, velocity, acceleration and bias term. The variable lag refers to the causal time delay between motor cortical neuron activity and kinematics due to the propagation effects of signals thru the motor and peripheral nervous systems. Here it is experimentally set as 200 ms [Wu et al., 2006; Wang et al., 2007c].

In traditional Kalman filter, the weight estimation of the linear tuning function $\tilde{H}_t$ is given from training data by

$$\tilde{H}_t = (E[\tilde{x}_{t+\text{lag}}]^T \tilde{x}_{t+\text{lag}})^{-1} E[\tilde{x}_{t+\text{lag}} \lambda_t]$$

(5-3)

Different from traditional Kalman filter, the linear filter weights in the tuning function, which represent the preferred kinematic direction, are modeled as a slowly changing random walk in dual Kalman filter. In this way, the dual estimation on tuning function parameters would demonstrate the transformation of the neuron encoding.

$$\tilde{H}^T_t = \tilde{H}^T_{t-1} + u_t^j$$

(5-4)

where $\tilde{H}^j_t$ represents the linear tuning parameters of neuron $j$ at time index $t$. Here we only model the tuning parameter of the first 10 most important neurons as we selected in Chapter 4 by the information theoretical criterion. The tuning parameters of the 10 neurons change over time with the dependence on the previous tuning parameters. $(\cdot)^T$ represents the transformation operation. $u_t^j$ is zero-mean Gaussian distributed noise with covariance $Q_{2k}$.

$n_{1k}$ is zero-mean Gaussian distributed noise with covariance $R_{1k}$, which is contributed by the noisy kinematics states. $n_{2k}$ is zero-mean Gaussian distributed noise with covariance $R_{2k}$,
which is contributed by the changing tuning parameters. At each time index, the kinematics vector is first derived as the state from the observation of firing rate in test by Equations 5-5 a-e.

\[ x_{k|k-1} = F_k x_{k-1|k-1} \]  \hspace{1cm} (5-5 a)

\[ P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_{1k} \]  \hspace{1cm} (5-5 b)

\[ K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_{1k})^{-1} \]  \hspace{1cm} (5-5 c)

\[ P_{k|k} = F_k P_{k-1|k-1} F_k^T + Q_{1k} \]  \hspace{1cm} (5-5 d)

\[ x_{k|k} = x_{k|k-1} + K_k (\lambda_k - H_k x_{k|k-1}) \]  \hspace{1cm} (5-5 e)

After the kinematics state is estimated from the observation, the tuning parameters for each neuron are then estimated by another Kalman filter by Equations 5-6 a-d.

\[ Ph_{k|k-1} = Ph_{k-1|k-1} + Q_{2k} \]  \hspace{1cm} (5-6 a)

\[ Kh_k = Ph_{k|k-1} x_k (x_k^T Ph_{k|k-1} x_k + R_{2k})^{-1} \]  \hspace{1cm} (5-6 b)

\[ Ph_{k|k} = Ph_{k|k-1} + Kh_k x_k^T Ph_{k|k-1} \]  \hspace{1cm} (5-6 c)

\[ H_k^T = H_{k-1}^T + Kh_k (\lambda_k - H_k x_{k|k-1}) \]  \hspace{1cm} (5-6 d)

Notice that carefully choosing the parameters in the noise estimation (covariance \( Q_{1k} \) in state dynamic model and covariance \( Q_{2k} \) in tuning dynamic model) could affect the algorithm performance. However, since we have no access to the desired kinematics in the test data set, the parameter estimations of both algorithms were obtained from the training data sets. For the Kalman filter, the noise in the kinematics model (Equation 5-1) is approximated by a Gaussian distribution with covariance \( Q_{1k} \). We set the initial state \( \tilde{x}_0 \) to be the zero vectors and the state variance \( P_{00} \) is estimated as the state variance from the training data.
The initial tuning parameter $\tilde{H}_0$ can be set as the one estimated from training by least square. It is somewhat different to set the variance parameters $Q_{2k}$ and $P_{h_{00}}$ in the tuning dynamic model. This is because we have the access to a series of the stochastic kinematics signals in the training set, but only the deterministic result to get the tuning parameters by least square solution. In order to get a series of the tuning parameter changing over time, we run the dual Kalman (Equation 5-6 a-d) to estimate the tuning parameters over time in the training set, where the kinematics state is set directly as the true value. Since in the testing set, the noise term is always contributed by 2 terms, the noisy kinematics state and the noisy tuning parameters, here we set covariance $Q_{2k}$ of the noise term $u_k$ in tuning dynamic is only 20% of the noise variance approximated by $\overline{(H_i - \bar{H}_{i-1})}$ from the time series of the tuning parameters. The variance $P_{h_{00}}$ is also set as 20% of the variance from the time series of the tuning parameters estimated from the training data.

Table 5-1 shows reconstruction results on 800 sample segment (time index from 213.5 m to 293.5 m) of a test segment of neural data by Kalman filter and by dual Kalman with tuning parameter modification on 10 most important neurons with the criterion Normalized Mean Square Error (MSE normalized by the power of the desired signal) between the desired signal and the estimations.

Table 5-1 shows that dual Kalman filter obtained less NMSE than Kalman filter with fixed tuning parameters for all the kinematics. Figure 5-1 shows the reconstruction performance by Kalman filter and dual Kalman filter on 10 most important neurons for 1000 test samples. The left and right column plots display the reconstructed kinematics for $x$-axis and $y$-axis. The 3 rows of plots illustrate from top to bottom the reconstructed position, the velocity and the acceleration.
In each subplot, the red line indicates the desired signal, the green line indicates the estimation by Kalman filter and the blue line indicates the estimation by Dual Kalman filter. We zoom in the position reconstruction in the plots. It is shown that dual Kalman filter provides better estimation at the peak of the desired signal than Kalman filter, because the tuning parameter is slowly tuned over the time. Figure 5-2 shows the tracking of the tuning parameters of the 10 neurons estimated by dual Kalman filter in test set. As we expected, we see the slow change of the parameters over the time. Neuron 72 and neuron 158 show diverge of the parameter change. It only appears when a pair or pairs of the parameter changes fast over time. We could infer that after the linear projection, the pair of the fast changing weight could results in a slow change of the linear output.

The preliminary results of the dual Kalman shows the possibility of the tracking the nonstationary tuning properties of the motor neurons. As we know from the experiment, the results are very sensitive to the parameter settings. The systematic way to decide the optimal parameters could be studied. Again the algorithm should be tested for longer data in the future.
Table 5-1. Results of the kinematics reconstructions by Kalman and dual Kalman for segment of test data

<table>
<thead>
<tr>
<th>NMSE</th>
<th>Position</th>
<th>Velocity</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
</tr>
<tr>
<td>Kalman</td>
<td>0.5706</td>
<td>0.5222</td>
<td>0.4747</td>
</tr>
<tr>
<td>Dual Kalman</td>
<td>0.5574</td>
<td>0.5170</td>
<td>0.4740</td>
</tr>
</tbody>
</table>

Figure 5-1. The reconstructed kinematics for 2-D reaching task by Kalman and dual Kalman filter
Figure 5-2. The tracking of the tuning parameters for the 10 most important neurons in dual Kalman filter
LIST OF REFERENCES


BIOGRAPHICAL SKETCH

Yiwen Wang received a B.S. in engineering science with a minor in automatic control from University of Science and Technology of China (USTC, Hefei, Anhui, China) in 2001. In 2004, she received a master’s degree in engineering science with a minor in pattern recognition and intelligent system from University of Science and Technology of China (USTC, Hefei, Anhui, China). Right then, she joined the Department of Electrical and Computer Engineering at the University of Florida-Gainesville, FL, USA, and received a Ph.D. in 2008. Under the guidance of Dr. Jose C. Principe in computational neuro-engineering lab, she has investigated the application of advanced signal processing and control methods to neural data for brain machine interfaces (BMIs). Her research interests are in brain machine interfaces, statistical modeling on biomedical signals, adaptive signal processing, pattern recognition, and information theoretic learning.