Temporal Coding of Cortical Neural Signals and Camera Motion Estimation in Target Tracking

by

Chenhui Yang

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2012 by the
Graduate Supervisory Committee:

Jennie Si, Chair
Leonidas Jassemidis
Christopher Buneo
Glen Abousleman

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

This dissertation includes two parts. First it focuses on discussing robust signal processing algorithms, which lead to consistent performance under perturbation or uncertainty in video target tracking applications. Projective distortion plagues the quality of long sequence mosaicking which results in loosing important target information. Some correction techniques require prior information. A new algorithm is proposed in this dissertation to this very issue. Optimization and parameter tuning of a robust camera motion estimation as well as implementation details are discussed for a real-time application using an ordinary general-purpose computer. Performance evaluations on real-world unmanned air vehicle (UAV) videos demonstrate the robustness of the proposed algorithms. The second half of the dissertation addresses neural signal analysis and modeling. Neural waveforms were recorded from rats' motor cortical areas while rats performed a learning control task. Prior to analyzing and modeling based on the recorded neural signal, neural action potentials are processed to detect neural action potentials which are considered the basic computation unit in the brain. Most algorithms rely on simple thresholding, which can be subjective. This dissertation proposes a new detection algorithm, which is an automatic procedure based on signal-to-noise ratio (SNR) from the neural waveforms. For spike sorting, this dissertation proposes a classification algorithm based on spike features in the frequency domain and adaptive clustering method such as the self-organizing map (SOM). Another major contribution of the dissertation is the study of functional interconnectivity of neurons in an ensemble. These functional correlations among neurons reveal spatial and temporal statistical dependencies, which consequently contributes to the understanding of a neuronal substrate of meaningful behaviors. This dissertation proposes a new generalized yet simple method to study adaptation of neural ensemble activities of a rat's motor cortical areas during its cognitive learning process. Results reveal interesting temporal firing patterns underlying the behavioral learning process.

DEDICATION

Dedicated to my parents, Qingtian Yang and Yanping Liu; and to my wife, Xue Jiang.

ACKNOWLEDGEMENTS

I especially would like to acknowledge my advisor, Dr. Jennie Si, for her continued encouragement, support and guidance throughout my PhD program. Without her help, I could not have finished the Ph.D. studies and this dissertation.

I would like to acknowledge Dr. Glen Abousleman for giving me the opportunity to work on the exciting projects broadening my vision in image processing and software optimization based on which part of dissertation has been developed.

I would like to acknowledge my other committee members: Dr. Leon Iasemidis and Dr. Christopher A. Buneo.

I would like to acknowledge the fellow graduate students in my group. They are Yuan Yuan, Hongwei Mao, and Bing Cheng.

Finally, I would like to acknowledge that this work was supported by General DynamicsC4 Systems and t by the NSF under grant ECCS-0702057 and ECCS-1002391.

TABLE OF CONTENTS

Chapter Page

Chapter Page

Chapter 1

INTRODUCTION

Signal detection, classification, and modeling are fundamental and important topics in the field of signal processing. Most well known algorithms were developed based on some statistical distributions of the signal and the noise. By doing, one obtains assurances of some type of optimality. However, real world applications do not usually meet those conditions, and thus, robustness of the algorithms may not be retained. In this dissertation, robustness of an algorithm refers to consistent performance of the signal processing algorithm under consideration under a wide range of realistic signal perturbations or system uncertainty. This dissertation addresses robust signal detection, classification, and modeling with applications to image target tracking and neural spike analysis.

The first contribution of this dissertation is correction of projective distortion in long-image-sequence mosaics without prior information. Image mosaicking is the process of piecing together multiple video frames or still images from a moving camera to form a wide-area or panoramic view of the scene being imaged. Mosaics have widespread applications in many areas such as security surveillance, remote sensing, geographical exploration, agricultural field surveillance, virtual reality, digital video, and medical image analysis, among others. When mosaicking a large number of still images or video frames, the quality of the resulting mosaic is compromised by projective distortion. That is, during the mosaicking process, the image frames that are transformed and pasted to the mosaic become significantly scaled down and appear out of proportion with respect to the mosaic. As more frames continue to be transformed, important target information in the frames can be lost since the transformed frames become too small, which eventually leads to the inability to continue further. Some projective distortion correction techniques make use of prior information such as GPS information embedded within the image, or camera internal and external pa-

rameters. Alternatively, this study proposes a new algorithm to reduce the projective distortion without using any prior information whatsoever. Based on the analysis of the projective distortion, an affine model is used to approximate the projective matrix that describes the transformation between image frames. Using singular value decomposition, it is possible to deduce the affine model scaling factor that is usually very close to 1. By resetting the image scale of the affine model to 1, the transformed image size remains unchanged. Even though the proposed correction introduces some error in the image matching, this error is typically acceptable and more importantly, the final mosaic preserves the original image size after transformation. The evaluation of this new correction algorithm with two real-world unmanned air vehicle (UAV) sequences demonstrates that the proposed method is effective and suitable for real-time implementation.

The second contribution of this dissertation is software-based robust global motion estimation for real-time video target tracking. In video tracking systems using image subtraction for motion detection, the global motion is usually estimated to compensate for the camera motion. The accuracy and robustness of the global motion compensation critically affects the performance of the target tracking process. The global motion between video frames can be estimated by matching the features from the image background. However, the features from moving targets contain both camera and target motion and should not be used to calculate the global motion. Sparse optical flow is a classical image matching method. However, the image features selected by optical flow may come from moving targets, with some of the image features matched not being accurate, which leads to poor video tracking performance. Least Median of Square (LMedS) is a popular robust linear regression model and has been applied to real-time video tracking systems implemented in hardware to process up to 7.5 frames/second. In this study, a robust regression method is used to select features only from the image background for robust global motion estimation, and a real-time

(10 frames/second) software-based video tracking system is developed for the platform of an ordinary Windows-based general-purpose computer. The software optimization and parameter tuning for real-time execution are discussed in detail. With real-world Unmanned Air Vehicle (UAV) videos, the evaluations of tracking performance demonstrate the improvements in global motion estimation in terms of accuracy and robustness.

The third contribution of this dissertation is multi-scale correlation of wavelet coefficients approach to spike detection. Extracellular chronic recordings have been used as important evidence in neuroscientific studies to unveil the fundamental neural network mechanisms in the brain. Spike detection is the very first step in the analysis of the recorded neural waveforms to decipher useful information and to provide useful signals for brain machine interface applications. The process of spike detection is to extract action potentials from the recordings which are often compounded with a myriad of noise from different sources. This dissertation proposes a new detection algorithm, which leverages a technique from wavelet based image edge detection. It utilizes the correlation between wavelet coefficients at different sampling scales to create a robust spike detector. The algorithm has one tuning parameter, which potentially reduces subjectivity of detection results. Both artificial benchmark data sets and real neural recordings are used to evaluate the detection performance of the proposed algorithm. Compared with other detection algorithms, the proposed method has a comparable or better detection performance with the potential for real-time implementation.

The fourth contribution of this dissertation is spike classification using neural waveform frequency domain features and self-organizing map (SOM) clustering. It is well accepted that even some of the simplest act elicit a dynamic response of a large number of neurons. In recent years, multi-array electrode (MAE) recording systems have made real time simultaneous recording of a large number of neurons possible. Based on the raw neural waveforms, individual spike instances are first detected. It is

followed by a classification process to associate some of the detected waveforms with a single neuron. Most of the existing spike classification techniques involve extensive calculation plus assumptions about prior information on the signal or the noise, and some requires tedious parameter tuning. This dissertation proposes a new spike classification algorithm based on spike waveform features in the frequency domain and the self-organizing map (SOM). It requires less prior information and it operates with intuitive parameter specifications. The proposed algorithm is evaluated on artificial and real neural recordings. The results are comparable or better than the couple popular published algorithms and also some commercial algorithms.

The fifth contribution of this dissertation is the study Spike-timing-dependent plasticity (STDP) during a cognitive learning. Spike-timing-dependent plasticity (STDP) has been implicated in neural development and learning. *In vitro* recordings from paired neurons reveal that the STDP process adjusts the connection strengths based on the relative timing between the action potentials of a pair of pre- and post synaptic neurons. However, it is often difficult to do the same experiments *in vivo* due to challenges associated with intracellular recordings. This becomes even more challenging when using live and behaving subjects. On the other hand, as a spike-time coding scheme, neuronal interactions measured by synaptic efficacy has been studied extensively by computational neuroscientists. Several algorithms have been frequently used to estimate the interaction strength between paired neurons. Cross correlation, mutual information, and Granger causality are some examples. To simultaneously account for multiple pairwise interactions among an ensemble of neurons, network likelihood models have been proposed and demonstrated as efficient tools for analyzing spatiotemporal spike firing patterns. Accordingly, maximum likelihood methods have been developed to estimate the model parameters. This study is to derive a principled approach to estimating the functional synaptic interaction strength in the network likelihood model. It is based on direct analysis of spatiotemporal spike firing patterns. The computation required to

estimate the model parameter is therefore straightforward with few assumptions. In addition, this new approach sheds light on the neuronal interaction patterns in rats' motor cortical areas in relation to the rats' cognitive learning control behaviors. The changes in functional synaptic efficacy were studied based on *in vivo* extracellularly recorded spike trains from behaving rats.

Chapter 2

CORRECTION OF PROJECTIVE DISTORTION IN LONG-IMAGE-SEQUENCE MOSAICS WITHOUT PRIOR INFORMATION

2.1 Introduction

Image mosaicking is an image processing method that registers multiple frames of a video sequence with respect to a common reference coordinate system to compose a panorama consisting of all frames. Generally, the image registration is described by a projective transform. Image mosaics have wide-spread applications such as security surveillance [1, 2] , remote sensing [3], geographical exploration [4], agricultural field surveillance [5], virtual reality [6], digital video, and medical image analysis [7].

A main source of error when generating a mosaic is known as projective distortion. Due to the principles and assumptions of the projective model, objects that are far away from the optical axis of the camera image plane are smaller than those near the optical axis when they are projected onto the image plane. Consequently, this projective distortion accumulates as the mosaicking process continues. For short image sequences, the projective distortion is not readily apparent. But for long image sequences, the projective distortion becomes much more pronounced and eventually leads to the inability to continue. The classical method to reduce the projective distortion is to rectify the image to yield a fronto-parallel view [8, 9]. These methods require the knowledge of several camera parameters such as focal length and camera posing (pan, tilt, and row angles), or they need to extract image features. Recently, several methods have been investigated that use the crossed-slits projection model rather than the projective model to reduce the distortion [10, 11]. However, with most camera systems, correction of the projective distortion requires the synthesize of crossed-slits projection views from regular perspective images, which may be infeasible.

In this paper, we propose a new projective distortion correction method that is independent of camera parameters. Due to its simplicity, it can be used with a variety of mosaicking algorithms without significant changes to the baseline system.

2.2 Projective Distortion Correction

2.2.1 Image Mosaic Based on the Projective Model

To create a mosaic from a sequence of frames or still images, the first step is frame matching. In this process, the corresponding points between frames are identified. In this paper, the image matching was implemented with the scale-invariant feature transform (SIFT) method [12]. The next step is to estimate the transform matrix between the frames. The transform matrix transforms the frames with respect to a common reference coordinate system. Both projective (Equation (2.1)) and affine models (Equation (2.2)) can be used to align the multiple frames. As shown in Equations (2.1) and (2.2), the difference between the projective and affine models lies in the last row of the matrix.

$$P = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{vmatrix}. \tag{2.1}$$

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{vmatrix}. \tag{2.2}$$

For image sequence, $I_0, I_2, ..., I_n, ..., I_N, 0 \leq n \leq N$, let $T_i, 1 \leq i \leq N$, be the projective/affine transform matrix between frame pairs, $\{I_{i-1}, I_i\}$. For image frame, I_n, the corresponding transform matrix, T_{n0}, between I_0 and I_n, is defined as in Equation (2.3):

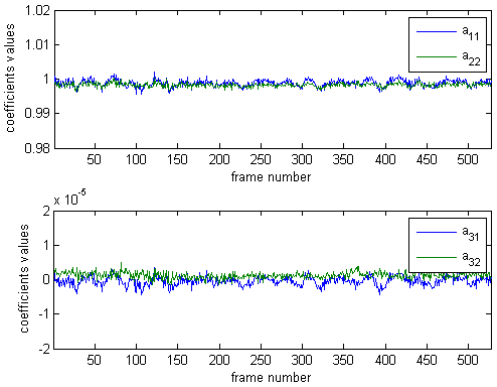
$$T_{n0} = T_n \times T_{n-1} \times \cdots T_2 \times T_1. \tag{2.3}$$

2.2.2 *Projective Distortion Correction by Approximating the Projective Model with a Translation Model*

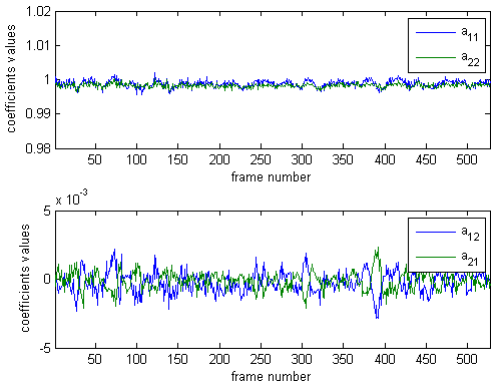
The projective model can capture "chirping" and "keystoning" effects [13]. The "chirping" effect refers to the change of spatial frequency with respect to spatial location, while "keystoning" refers to the effect of line convergence. The coefficients, a_{31} and a_{32}, in Equation (2.1) reflect these effects. However, since the frame rate for most video sequences is 15-30 frames per second, the chirpping and keystoning effects of two consecutive frames are relatively small as compared with the other coefficients.

For long image sequences, the multiplication of transforms, T_i's, scales the frame far away from the optical axis of the camera system to a much smaller size with respect to its original size. Usually, for the affine transform matrix, we can analyze its transform effect on images via singular value decomposition (SVD) [14]. The affine transform matrix can be decomposed into a rotation matrix and a scale matrix. The rotation matrix does not change the image size, while the scale matrix directly determines the transformed image size. However, there is no general way to decompose the projective model in the same way as the affine model [14]. Figure 2.1(a) shows that the coefficients a_{31} and a_{32} have much smaller magnitudes than a_{11} and a_{22} for a representative UAV video sequence. Accordingly, since they are much smaller than the other coefficients of the projective matrix, we approximate the projective model with an affine model by letting $a_{31} = 0$ and $a_{32} = 0$ in Equation (2.1). Similarly, Figure 2.1(b) shows that the coefficients a_{12} and a_{21} are much smaller than a_{11} and a_{22}. On the other hand, a_{21} and a_{22} are shown to be close to 1. Thus, we conclude that the relative motion between two consecutive frames is mainly pure translation for the UAV video sequence.

Equation (2.4) shows the synthesized transform matrix (pure translation) used to correct the projective distortion.



(a) The coefficients a_{31} and a_{32} represent "chirpping" and "keystoning" effects, and are much smaller in magnitude than a_{11} and a_{22}.



(b) The coefficients a_{12} and a_{21} are much smaller in magnitude than a_{11} and a_{22}. The relative motion between frames for this video sequence is mainly pure translational.

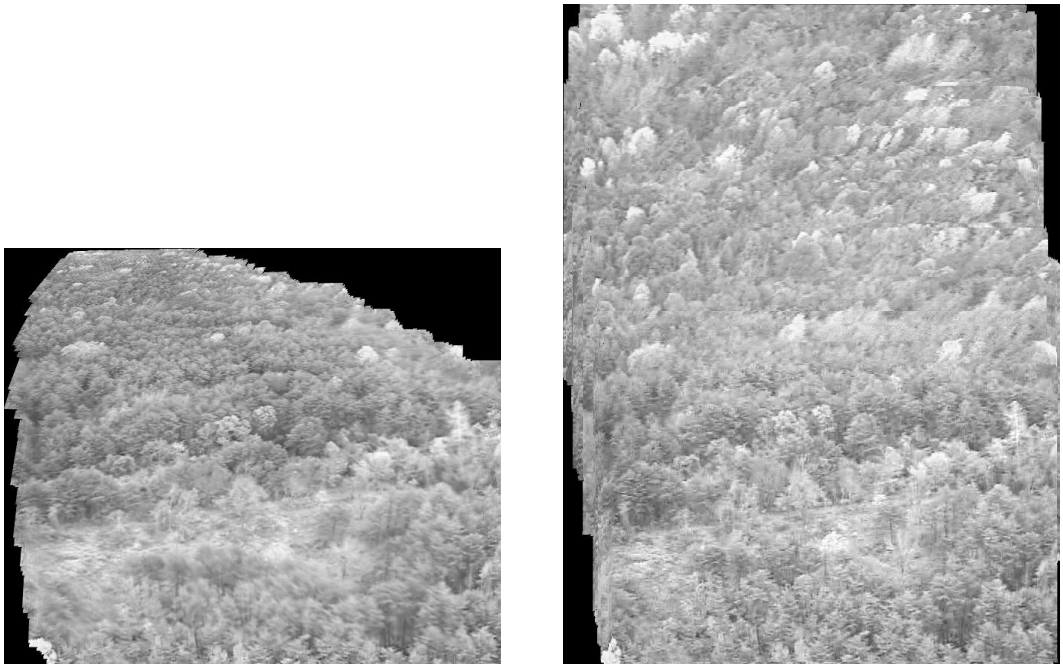
Figure 2.1: Projective matrix coefficients of a UAV video sequence. Approximately 500 frames were used for analysis.

$$P = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & a_{13} \\ 0 & 1 & a_{23} \\ 0 & 0 & 1 \end{vmatrix}. \tag{2.4}$$

To illustrate the performance of the proposed method, Figure 2.2 shows the mosaic resulting from the UAV video sequence with and without the projective distortion correction. Comparing Figure 2.1a and 2.1b, it is obvious that the projective distortion is reduced dramatically by approximating the projective transform matrix with a pure translation matrix as shown in Equation (2.4).

2.2.3 Projective Distortion Correction by Approximating the Projective Model with Affine Model

The video analyzed in Section 2.2.2 has very little rotation. That is why a_{12} and a_{21} is much smaller in magnitude than a_{11} and a_{22}. If the video exhibits rotation, we can still correct the projective distortion with an approximated affine model and SVD. First, we approximate the projective model with an affine model as shown in Equation (2.5). Secondly, we apply SVD to the matrix, A, as shown in Equation (2.6).



(a) Mosaic without projective distortion correction. Note that subsequently processed frames (shown at the top) become smaller as the mosaicking process continues.

(b) Mosaic with proposed projective distortion correction. Note that subsequently processed frames retain the correct scale.

Figure 2.2: Mosaic constructed from UAV video sequence with and without projective distortion correction. Number of mosaicked frames is identical in both cases.

$$P = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{vmatrix}. \tag{2.5}$$

$$A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = R(\theta)R(-\theta)DR(\phi). \tag{2.6}$$

Here, $R(\theta)$, $R(-\theta)$, and $R(\phi)$ are rotation matrices, and θ and ϕ are the rotation angles. $D = \begin{vmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{vmatrix}$ is the scale matrix, and λ_1 and λ_2 are scale factors. These scale factors are responsible for the changes in image size. In order to correct the projective distortion, we set λ_1 and λ_2 to 1, and therefore D becomes an identity matrix. The new transform, A_{new}, defined in Equation (2.7), will replace A used in the image mosaic,

where I is a 2×2 identity matrix:

$$A_{new} = R(\theta)R(-\theta)IR(\phi). \tag{2.7}$$

The corrected projective matrix is calculated in Equation (2.8), where $\mathbf{t} = [a_{13}\ \ a_{23}]^T$, $0^T = [0\ \ 0]$.

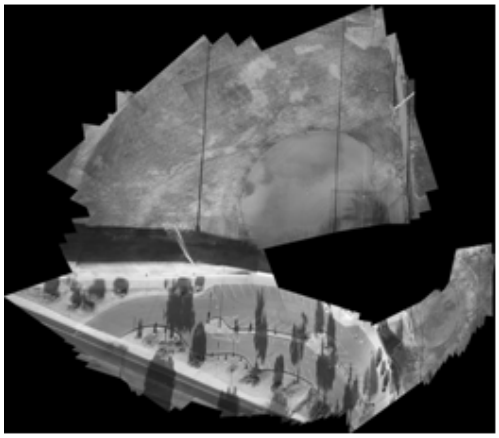
$$P_{new} = \begin{vmatrix} A_{new} & \mathbf{t} \\ 0^T & 1 \end{vmatrix}. \tag{2.8}$$

The proposed projective distortion correction algorithm thus has two steps: First, we replace the projective transform matrix with an affine model by letting $a_{31} = a_{32} = 0$ in the estimated projective transform matrix. Next, apply SVD to the approximated affine model and calculate the new transform matrix, P_{new}, by replacing the scale matrix, D, with an identity matrix as shown in Equation (2.8).

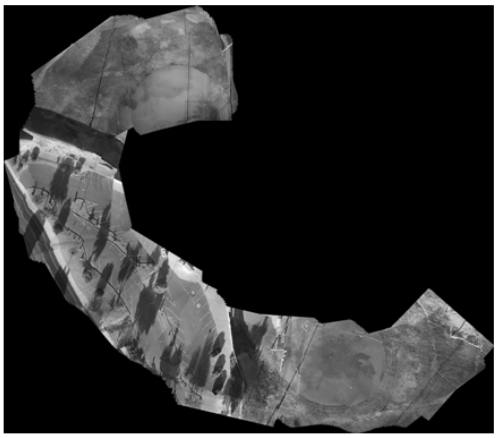
Figure 2.3 shows the mosaic created from a second UAV video sequence with P_{new}. Note that the video sequence exhibits large amounts of rotation. Comparing Figure 2.3a and 2.3b, we note that the projective distortion has been reduced dramatically. However, we also note some misalignments in the mosaic as shown by the red and blue circles in Figure 2.4. Compared with the projective distortion, however, such misalignments are deemed quite acceptable.

2.3 Conclusion

We have developed an algorithm for correcting the projective distortion inherent in the mosaicking process of long video sequences. The proposed algorithm reduces the projective distortion without requiring UAV telemetry information. The method is also computational tractable, which makes it suitable for real-time applications.

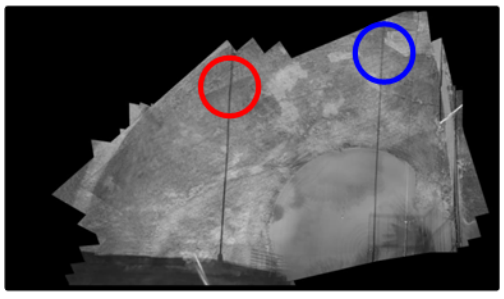


(a) Mosaic without projective distortion correction. Note that subsequently processed frames (shown at the bottom) become smaller as the mosaicking process continues.

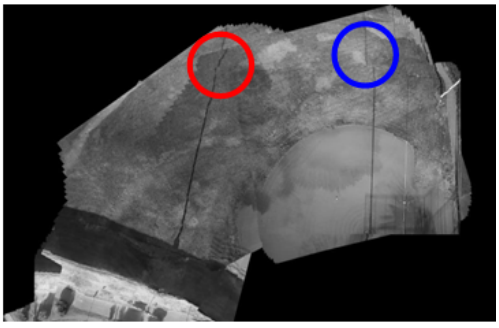


(b) Mosaic with proposed projective distortion correction using P_{new} in Equation 2.8. Note that subsequently processed frames retain the correct scale.

Figure 2.3: Mosaic constructed from UAV video sequence with and without projective distortion correction. Number of mosaicked frames is identical in both cases and the UAV flies counterclockwise.



(a) Segment of mosaic without projective distortion correction.



(b) Segment of mosaic with projective distortion correction.

Figure 2.4: Projective distortion correction results in slight misalignment.

Chapter 3

SOFTWARE-BASED ROBUST GLOBAL MOTION ESTIMATION FOR REAL-TIME VIDEO TARGET TRACKING

3.1 Introduction

Video target tracking is a process of identifying and associating moving objects across multiple video frames. It has wide applications in many areas such as video security surveillance[15], video compression[16], neuroscience study[17], medical diagnose[18], traffic monitoring and control[19], and many others. Motion detection and estimation are fundamental processes in any tracking system. The accuracy and robustness of the motion detection stage will determine the overall tracking performance. In a real-time tracker, the feasibility of real-time motion detection and estimation demands algorithm simplicity. Image background subtraction between consecutive frames is a simple and effective method of motion detection[20, 21, 22], especially for tracking systems where a static camera is assumed. For tracking systems with moving cameras, image background subtraction is still applicable after the global (camera) motion has been compensated for. The motion of the camera between frames can be described by a projective matrix, which is estimated through image registration of image background features between video frames.

Typical image registration methods match not only the image features of the background, but also the features of moving targets. However, to obtain an accurate registration, the camera motion should be estimated only with image background features. Thus, it is necessary to discriminate between features belonging to the background from those belonging to the moving targets to obtain an accurate estimation of camera motion. Least Median of Square (LMedS), a robust linear regression method, has been applied to image tracking[23] to identify image features from the background. It was also reported in a hardware implementation of a real-time (7.5 frames/second) image tracking system[24].

In this paper, we propose a camera motion estimation algorithm for a real-time (10 frames/second) software-based video tracking system on a general-purpose computer. The proposed algorithm includes an image registration procedure based on optical flow, and a feature selection procedure based on LMedS. Software optimization, parameter selection, and tunning are discussed in detail. Evaluation results obtained with real-world UAV video sequences demonstrates that the proposed method robustly identifies the background motion.

3.2 Global (Camera) Motion Estimation

For the system discussed herein, global (camera) motion estimation comprises two steps: First, the registered pixels between frames are identified by an optical flow method. Secondly, the registered pixels from the background are used to estimate the camera motion.

3.2.1 Image Registration Based on Sparse Optical Flow

Generally speaking, optical flow methods can be classified into two categories: dense optical flow[25] and sparse optical flow[26]. The dense optical flow method calculate the motion of each pixel while sparse optical flow only considers the motion of selected pixels. Dense optical flow provides comprehensive motion estimation but is computationally expensive, especially for software-based real-time tracking systems. On the contrary, sparse optical flow is much more computationally tractable, which makes it suitable for the real-time application considered in this paper.

Let I_i and I_{i+1} be i^{th} and $(i+1)^{th}$ video frames. Without loss of generality, we assume I_i and I_{i+1} are grayscale images. The gray values of the pixels at position (x,y) are represented by $I_i(x,y)$ and $I_{i+1}(x,y)$, respectively.

The underlying assumption in the optical flow method is that the pixel values are constant with camera motion as defined in Equation (3.1), where d_x and d_y are the

pixel displacements due to camera motion between I_i and I_{i+1}:

$$I_i(x,y) = I_{i+1}(x+d_x, y+d_y). \tag{3.1}$$

The differences in pixel values between frames is measured by $E(d_x,d_y)$, which is defined in Equation (3.2), where u_x and u_y are the radii of the search neighborhood:

$$E(d_x,d_y) = \sum_{d_x=-u_x}^{d_x=u_x} \sum_{d_y=-u_y}^{d_y=u_y} [I_i(x,y) - I_{i+1}(x+d_x, y+d_y)]^2 . \tag{3.2}$$

For simplicity, we define $\mathbf{d} = (d_x,d_y)$. The essence of image registration is to find a $\mathbf{d}$ which satisfies Equation (3.1), or equivalently, minimizes $E(d_x,d_y)$. In this paper, we use the Lucas-Kanade algorithm[27] to estimate $\mathbf{d}$ iteratively. Let $\mathbf{d}_{min}$ be the optimal solution that minimizes $E(d_x,d_y)$. The first derivative of $E(d_x,d_y)$ at $\mathbf{d}_{min}$ should be zero as defined in Equation (3.3):

$$\frac{\partial E(\mathbf{d}_{min})}{\partial \mathbf{d}_{min}} = 0. \tag{3.3}$$

Substituting Equation (3.2) into Equation (3.3) yields

$$\frac{\partial E(\mathbf{d}_{min})}{\partial \mathbf{d}_{min}} \approx -2 \sum_{d_x=-u_x}^{d_x=u_x} \sum_{d_y=-u_y}^{d_y=u_y} [I_i(x,y) - I_{i+1}(x+d_x, y+d_y)] \cdot \begin{bmatrix} \frac{\partial I_{i+1}(x+d_x,y+d_y)}{\partial d_x} \\ \frac{\partial I_{i+1}(x+d_x,y+d_y)}{\partial d_y} \end{bmatrix}^T . \tag{3.4}$$

where $I_{i+1}(x+d_x,y+d_y)$ is approximated by its first-order Taylor expansion,

$$I_{i+1}(x+d_x,y+d_y) \approx I_{i+1}(x,y) + \frac{\partial I_{i+1}(x,y)}{\partial d_x} d_x + \frac{\partial I_{i+1}(x,y)}{\partial d_y} d_y. \tag{3.5}$$

Let

$$\delta I \triangleq I_i(x,y) - I_{i+1}(x,y), \;\; I_x \triangleq \frac{\partial I_{i+1}(x,y)}{\partial d_x}, \;\; I_y \triangleq \frac{\partial I_{i+1}(x,y)}{\partial d_y}, \;\; \nabla I = [I_x I_y]^T . \tag{3.6}$$

Equation (3.4) then becomes

$$\frac{1}{2}\frac{\partial E(\mathbf{d}_{min})}{\partial \mathbf{d}_{min}} \approx \sum_{d_x=-u_x}^{d_x=u_x}\sum_{d_y=-u_y}^{d_y=u_y}\left[\nabla I^T \mathbf{d}_{min} - \delta I\right]\nabla I^T,$$

$$= \sum_{d_x=-u_x}^{d_x=u_x}\sum_{d_y=-u_y}^{d_y=u_y}\left(\begin{bmatrix} {I_x}^2 & I_xI_y \\ I_xI_y & {I_y}^2 \end{bmatrix}\mathbf{d}_{min} - \begin{bmatrix} \delta I \cdot I_x \\ \delta I \cdot I_y \end{bmatrix}\right). \tag{3.7}$$

Finally, let

$$A \triangleq \sum_{d_x=-u_x}^{d_x=u_x}\sum_{d_y=-u_y}^{d_y=u_y}\left(\begin{bmatrix} {I_x}^2 & I_xI_y \\ I_xI_y & {I_y}^2 \end{bmatrix}\right), \quad b \triangleq \sum_{d_x=-u_x}^{d_x=u_x}\sum_{d_y=-u_y}^{d_y=u_y}\left(\begin{bmatrix} \delta I \cdot I_x \\ \delta I \cdot I_y \end{bmatrix}\right). \tag{3.8}$$

Combining Equations (3.3), (3.7), and (3.8), $\mathbf{d}_{min}$ can be written as

$$\mathbf{d}_{min} = A^{-1}b. \tag{3.9}$$

In the process of deriving Equation (3.9), we assumed that the pixel displacement is small enough such that the first order Taylor expansion in Equation (3.5) is valid. This assumption indicates that the solution in Equation (3.9) may not be optimal if the pixel displacements are not close to zero. Furthermore, the derivation of Equation (3.9) does not consider presence of image noise. Therefore, in order to find an accurate motion estimation, it is necessary to iteratively estimate the motion using Equation (3.9). Let $\mathbf{d}^k_{min} = (d^k_x, d^k_y)$ be the motion estimation after the k^{th} iteration. For the $(k+1)^{th}$ iteration, the pixel displacement is estimated by replacing $I_{i+1}(x+d_x, y+d_y)$ in (3.2) with $I_{i+1}(x+d^k_x, y+d^k_y)$. Following the same procedures as shown in (3.2) - (3.9), we can derive the motion estimated for the $(k+1)^{th}$ iteration. Let η^{k+1} be the motion estimated in the $(k+1)^{th}$ iteration. The motion after $k+1$ iterations is defined as

$$\mathbf{d}^{k+1}_{min} = \mathbf{d}^k_{min} + \eta^{k+1}. \tag{3.10}$$

On the other hand, Equation (3.9) is valid only if A^{-1} exits. A is determined by image features I_x and I_y defined in the neighborhood of $I_{i+1}(x,y)$. Therefore, it is necessary to select neighborhoods where I_x and I_y are not close to zero. The study by Shi and Tomasi[28] indicated that eigenvalues of A could be used as a criterion to determine whether A^{-1} can be was solved reliably. In this paper, we use a threshold to select pixels based upon their eigenvalues. Let r $(0 < r < 1)$ and λ_{max} be the threshold and the maximum eigenvalue, respectively, of the image. Any pixels whose eigenvalues are in the range of $[r\lambda_{max}, \lambda_{max}]$ are selected for further analysis.

3.2.2 Image Pyramid Representation

The pixel motion, $\mathbf{d}_{min}$, is constrained in the range of $-u_x \leq d_x \leq u_x$ and $-u_y \leq d_y \leq u_y$, according to the discussion in Section 3.2.1. In order to estimate the motion correctly, the range of u_x and u_y must cover the range of possible motion between frames. However, it is impossible to predict the reasonable range of motion for arbitrary image frames. Intuitively, a larger search neighborhood is always better than a smaller neighborhood in terms of covering the possible motion. However, a larger search neighborhood requires more computations to estimate $\mathbf{d}_{min}$. On the other hand, the approximation of the Taylor expansion in Equation (3.5) holds when the motions, d_x and d_y $(d_x \in [-u_x, u_x], d_y \in [-u_y, u_y])$, are small enough. Therefore, it is necessary to estimate pixel motions in a multiple-resolution fashion.

Let I^L represent the L^{th} level resolution representation of image I defined in Equation (3.11). The original image is represented as $zero^{th}$ level, i.e., $I^0 = I$.

$$
\begin{aligned}
I^L(x,y) = {} & \frac{1}{4} I^{L-1}(2x,2y) + \\
& \frac{1}{8}\left[I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)\right] + \\
& \frac{1}{16}\left[I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1)\right] + \\
& \frac{1}{16}\left[I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y-1)\right].
\end{aligned} \tag{3.11}
$$

Let $\mathbf{d}^L$ be the pixel motion at the $L^{th}-$level resolution. According to Equation (3.11), the motion at the $(L-1)^{th}-$level resolution is defined as

$$\mathbf{d}^{L-1} = 2\mathbf{d}^L. \tag{3.12}$$

Multi-resolution motion estimation begins at the coarsest level. The motion at the coarsest level is estimated according to the discussion in Section 3.2.1. This motion is then mapped to the second coarsest level as the initial value of the iterative motion estimation given by Equation (3.12). With the initial value, motion of the second coarsest level is estimated again according to discussion in Section 3.2.1. The same procedures are applied to the next level and so on through level zero. The total motion across multiple resolutions is defined in Equation (3.13), where L_m is the number of levels:

$$d_{min} = \sum_{L=0}^{L_m} 2^L \mathbf{d}^L. \tag{3.13}$$

In summary, the sparse optical flow procedure discussed in this paper estimates the pixel displacement iteratively at multiple resolutions of the image. Additional details regarding sparse optical flow can be found in Bouguet[26].

3.2.3 Robust Camera Motion Estimation Based on Least Median of Squares (LMedS)

Camera motion can be estimated according to the registered pixel pairs across frames. The registered pixel pairs contain not only camera motion, but also motion from moving targets. Given an image transform matrix, $\mathbf{T}$, for a registered pixel pair, its accuracy can be evaluated by the matching error. Let $P_i = (x_i, y_i, 1)^T$ and $P_i' = (x_i', y_i', 1)^T$ be the i^{th} registered pixel pair of two frames, where $1 \leq i \leq N$, and N is the number of registered pairs. For an image transform matrix, $\mathbf{T}$, the matching error between P_i and P_i' is denoted by $e(i)$, as defined in Equation (3.14):

$$\mathbf{T} \triangleq \begin{pmatrix} t_{1,1} & t_{1,2} & t_{1,3} \\ t_{2,1} & t_{2,2} & t_{2,3} \\ t_{3,1} & t_{3,2} & t_{3,3} \end{pmatrix}, \quad e(i) = \|\mathbf{T}P_i - P_i'\|. \tag{3.14}$$

As mentioned previously, only registered pixel pairs containing camera motion should be used to estimate the camera motion. Under this assumption, registered pixel pairs of camera motion are defined as inliers, while registered pixel pairs of moving targets are considered outliers. Among several algorithms proposed to identify outliers to improve estimation accuracy of camera motion, the method identifying the outliers by a threshold[29] is suitable for real-time applications. Given a threshold, the method by Lin[29] identified outliers in two steps. First, all registered pixels are used to estimate the transform matrix with the matching errors of all registered pairs being calculated. All registered pixels are classified into two categories: a pair whose error is beyond a threshold is classified into an outliers cluster; otherwise it is classified into an inlier cluster. Secondly, all registered pairs belonging to the inlier cluster are used to estimate the camera motion. The threshold for classification is critical to the accuracy of the camera motion estimation, and should be selected according to the motion properties of the camera and of the moving targets.

Least Median of Squares (LMedS) is a robust linear regression method proposed by Rousseeuw and Leroy[30]. It does not utilize a threshold to identify the outliers. Furthermore, LMedS is more robust to outliers than linear regression. Accordingly, in the present work, we use LMedS to estimate the camera motion.

3.2.3.1 Affine and Projective model

The motion of the camera between frames can be described by a projective matrix. For videos captured in real time, Yang et at [31] showed that the projective model between two consecutive frames can be approximated by an affine model. Let $(x, y, 1)$

and $(x', y', 1)$ be the matched pixels from the background motion in two consecutive frames. Their relationship is described by an affine matrix, **A**, which is defined as

$$\mathbf{A} \triangleq \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}. \tag{3.15}$$

From (3.15), we can see that

$$x = a_{1,1}x' + a_{1,2}y' + a_{1,3}, \;\; y = a_{2,1}x' + a_{2,2}y' + a_{2,3}. \tag{3.16}$$

Equation (3.16) indicates that the camera motion between frames is linear if it is modeled by an affine matrix. Therefore, linear regression is a natural solution to estimate the camera motion. However, registered pixel pairs of moving targets are different from those resulting from camera motion, and therefore they do not satisfy the linear mapping defined in Equation (3.16). Thus, registered pixel pairs of moving targets are the outliers for estimating camera motion. The camera motion estimated by linear regression should only use inliers, i.e., registered pixel pairs of camera motion.

3.2.3.2 Linear Regression Based on Least Median of Squares (LMedS)

The input data of linear regression based on LMedS are the registered pixel pairs identified by optical flow, and the output is the estimated affine matrix. The algorithm is summarized as follows:

1. Randomly select a subset, S, from the input data. Let J be the number of possible combinations of three matched pixels pairs in the subset S;

2. For the $j^{th}(1 \leq j \leq J)$ combination of three matched pairs, an affine matrix, $\mathbf{A}_j$, is estimated accordingly;

3. For each affine matrix, $\mathbf{A}_j (1 \leq j \leq J)$, the matching errors of all S matched pairs are calculated according to Equation (3.14) by replacing $\mathbf{T}$ with $\mathbf{A}_j$. Let med_j be the median of matching errors. It is used as the evaluation quantity of background motion estimation by $\mathbf{A}_j$;

4. The motion of camera, $\mathbf{A}_{camera}$, is found by Equation (3.17):

$$\mathbf{A}_{camera} = \underset{\mathbf{A}_j}{\operatorname{argmin}} \{med_j\}. \tag{3.17}$$

Since the median is used as the evaluation quantity of background motion, the underlying assumption for LMedS is that the percentage of registered pixels from moving targets is less than 50%. This assumption should be considered in determing the paramters for motion detection.

3.2.4 Parameter Selection

Three pairs of registered pixels are needed to determine an affine matrix, while four pairs of registered pixels are needed to estimate a projective matrix. For example, suppose the number of registered pixel pairs for the input LMedS is S, the numbers of all possible combination of four and three matched pixels pairs are $C(S,4)$ and $C(S,3)$, respectively. The ratio of these is $\frac{S-3}{4}$. It means that the number of possible combination for projective matrix is greater than that for affine model if $S > 7$. Usually, pixels corresponding to moving targets have large eigenvalues since target motion results in large gradients. Thus, these pixels are highly likely to be selected for image registration. However, the assumption for LMedS is that registered pixel pairs from moving targets is in the minority. Therefore, it is necessary to select as many pixels corresponding to background as possible for image registration. In the present work, the video resolution is 320×240 pixels. We select 400 registered pixel pairs, i.e., $J = 400$. Among these registered pixel pairs, we randomly select 41 pairs used in linear regression based on

LMedS. For $S = 41$, the ratio, $\frac{J-3}{4}$, is almost 10. It indicates that the the computational complexity for the affine matrix is only 10% of that for the projectvie matrix. Therefore, we used the affine model as an approximation of the projective model for real-time applications.

The optical flow algorithm was implemented with OpenCV[32]. The parameters were selected so that the minority of registered pixel pairs are from the moving targets. Therefore, the threshold, r, for selecting the eigenvalue is small so that there are many pixels selected for image registration. Due to the continuity of the eigenvalue, once a pixel is selected, the other pixels around it will be selected as well. Consequently, most of the pixels near the moving targets are selected. In order to meet the minority assumption of LMedS, it is necessary to select pixel uniformly by specifying minimum distances between selected pixels. Table 3.1 shows the parameter lists for optical flow. Iteration of the motion estimation at each resolution level stops when the maximum number of iterations is reached or $\eta < 0.01$ (Equation (3.10)).

Table 3.1: Parameter lists for optical flow.

minimum distance (pixels) between selected pixels	10 pixels
Maximum number of iterations	40
Maximum number of multi-resolution	5
η	0.01
u_x	3 pixels
u_y	3 pixels

Median filtering used in LMedS is a non-linear process. In this paper, we use the numerical implementation[33] of median filtering whose algorithm complexity is $\mathrm{O}\left(n\right)$, where n is the number of matching errors.

3.3 Camera Motion Estimation with Real UAV Video

The video used for evaluation was captured by a moving camera mounted on a UAV. Note that if the camera motion is estimated accurately, detection of the moving targets based on image subtraction is also accurate. Therefore, the performance of estimating camera motion can be evaluated by the performance of detecting moving target based on image subtraction. In this paper, image morphological processing has been applied to the subtracted image to remove noise[34].

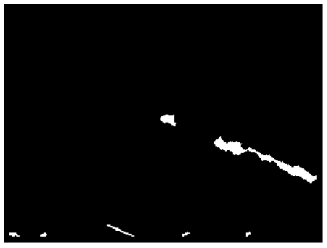
We developed a real-time (10 frames/second), software-based video tracking system that runs on an ordinary Windows-based general-purpose computer. The proposed method was implemented as the motion detection stage of the video tracking system. The host computer was equipped with an Intel Xeon Quad-core (2.83Ghz) CPU and 4GB of RAM.

Figure 3.1 shows the motion detection results for linear regression with a threshold for inliers [29] and LMedS. Figure 3.1a shows image frames containing moving targets highlighted by red circles. Figure 3.1b and Figure 3.1c are the binary representation of motion detections by two different algorithms, where white represents detected moving targets, and black represents the background. Comparison between Figure 3.1b and Figure 3.1c shows that motion detection based on LMeds has better performance than linear regression. In case that two consecutive video frames are the same, as shown in Figure 3.2a, no motion is detected. Figure 3.2b shows a false alarm by linear regression. However, as shown in Figure 3.2c, there is no false alsrm using LMeds.

3.4 Conclusion

In a video tracking system using image subtraction for motion detection, the accuracy and robustness of camera motion compensation critically affects the target tracking performance. The camera motion between frames should be estimated by matching

(a) Image frame used for motion detection. Moving targets are highlighted by red circles.



(b) Binary representation of motion detection by linear regression with a threshold for inliers (threshold= 1.4 pixel)[29].

(c) Binary representation of motion detection by robust linear regression based on LMedS.

Figure 3.1: Image frame and binary representations of motion detections by linear regression and LMedS.

(a) Image frame used for motion detection. UAV captures two identical frames; no motion should be detected.

(b) Binary representation of motion detection by linear regression with a threshold for inliers (threshold= 1.4 pixel)[29]. The detected motion is a false alarm.

(c) Binary representation of motion detection by robust linear regression based on LMedS. There is no false alarm since no moving targets are present.

Figure 3.2: Image frame without moving targets and its binary representation of motion detections by linear regression and LMedS.

the features from the image background and not from the moving targets. A software-based, real-time camera motion estimation procedure based on sparse optical flow and robust linear regression has been proposed in this paper. The tuning of parameters are discussed in detail. Evaluation with real UAV video sequences shows that the proposed algorithm is more accurate and robust than the linear regression method.

Chapter 4

A MULTI-SCALE CORRELATION OF WAVELET COEFFICIENTS APPROACH TO SPIKE DETECTION

4.1 Introduction

Neural action potentials, also known as nerve impulses or spikes, play an important role in understanding the central nervous system. In chronic multichannel recordings from behaving animals, action potentials are obtained by multichannel electrodes implanted in brain areas of interest. As such, noise from brain tissues, muscle movement, and other biological and instrumental interferences are inevitable [35]. As the first step of neuroscientific studies and engineering applications such as brain machine interfaces, identifying real neural spikes from noisy recordings is essential.

Several spike detection techniques have been developed and some are used in commercial neural recording packages. The simplest to implement, and the most intuitive method is by thresholding [36]. It identifies a spike by comparing the neural waveform with a pre-set threshold. This is obviously suitable for real-time application. However the manual selection of the threshold by users is subjective. The study in [37] showed high variability in manual spike detection and sorting results by different users. Since thresholding method mainly relies on the waveform amplitude, not considering other characteristics such as spike waveform shape, spike rising and falling slopes and spike width, it is thus sensitive to amplitude changes caused by either spike waveforms or noise. Consequently, false alarm rate may be high especially if the thresholding level is set low.

In [38], a statistical spike detection method was proposed based on a non-parametric Bayesian framework. With the prior probability provided by the user, and the posterior probability estimated by a nonparametric quantization algorithm, the method tests a Bayesian hypothesis based on the ratio between the probability of the recorded waveform being a neural spike and that of being background noise, given the wave-

form's shape and repetitiveness. This algorithm requires using a high threshold to gather spikes as ground truth. But such a high threshold may leave out real neural spikes with low magnitudes and thus results in missed detection.

The authors of [39] proposed an echo state network (ESN) and minimum average correlation energy (MACE) method for spike detection with a better detection performance than thresholding and matched filter detection. A set of "representative" spike signals were used for training the ESN. In order to catch all possible spike patterns including those embedded in background noise, a highly variable set of "representative" spikes are needed for training, which is a challenging manual task since no systematic procedure was introduced. The same problem was encountered and discussed in [37]. In addition, results of the MACE filter were compared with the method of thresholding for spike detection. However, neither the selection of "representative" spike signals nor the threshold was addressed in detail in the paper.

In [40], it presents an automatic and real-time detection method based on first extracting background noise and then performing a template match. This algorithm is based on an assumption that the noise has a colored Gaussian distribution. The restrictive nature of such assumption was discussed in [39] and [41]. It was pointed out that the high variances in neural spike appearances would sacrifice detection accuracy during template matching. Another template matching method was proposed in [41] where the variability of extracellular spike waveforms was modeled by a linear superposition of a long-interspike interval (ISI) neural spike waveform and a short-ISI neural spike waveform with different weights. Thus the spike waveform width varies. This actually indicates that wavelet transform is a natural solution for this problem. By varying the wavelet scales, one generates a series of wavelets ranging from long-ISI waveform to short-ISI waveform. Therefore, different wavelet transforms have been considered in neural spike detection applications.

The wavelet transform is a technique for representing a time domain signal by a set of functions that are scaled and time-translated from a mother wavelet. Its characteristics make it a natural candidate for transient signal representation and thus spike detection applications. Several spike detection algorithms based on wavelet transforms have been proposed.

Yang [42] proposed a spike detection algorithm based on discrete Harr transform (DHT). The wavelet function was selected based on its resemblance to the neural spike waveforms. Once the wavelet coefficients are computed by the Harr wavelet, a threshold was applied to these DHT coefficients for spike detection. The threshold was estimated under the assumption that noise complies with a Gaussian distribution, which is not realistic in real neural recordings.

Kim [43] proposed a detection method using thresholding on the discrete wavelet transform (DWT) coefficients. The DWT was performed by a bank of discrete time filters. It enhanced the detection performance by combining (multiplying) the wavelet coefficients. However, since it was a discrete dyadic wavelet, the multiple time translations scale down exponentially and therefore, it did not fully utilize the power of multiplying the wavelet coefficients at different levels. Sometimes, they may introduce adverse effects of missed detection.

The algorithms in [44] and [45] were based on wavelet packet and mutual information. Detection takes place in two steps. Firstly, hard thresholding is applied to recorded neural waveforms to select the potential neural spikes. Secondly, wavelet packet and mutual information are used to sort those potential pool of spikes from noise. As indicated by [46], the detection method in [44] and [45] can not be implemented in an unsupervised fashion. Or in other words, this algorithm is a user dependent approach to spike detection and therefore, its outcome varies according to the user's subjective selection of the parameters in the algorithm. In addition, estimating the mutual information is not straightforward.

Oweiss et al. [47, 48] proposed the multi-resolution generalized likelihood ratio test (MRGLRT) based on a Gaussian noise assumption. Two thresholds are needed in MRGLRT. First, a threshold matrix was used to suppress the noise term from the wavelet transform of neural waveforms. Second, another threshold was used for spike detection with generalized likelihood ratio test. It is necessary to optimize both threshold values to have an optimal detection performance under a Gaussian noise assumption. Optimization introduces additional computation.

Nenadic [49] and [50] proposed the wavelet detection method (WDM) based on continuous wavelet transform. First, by analyzing the neural recording waveform profile at each sampling scale via wavelet coefficients, a median filter based threshold is used to separate spikes from noise. At each scale, based on the spike waveform and noise waveform from the first step of thresholding, the variance of the wavelet coefficients corresponding to noise and the sample mean of the absolute value of wavelet coefficients corresponding to spikes are used to form a threshold value in a Bayesian hypothesis test to minimize a proposed detection cost. Finally, detection results at different scales are combined to form the final detection. Nenadic indicated in [49] that WDM outperformed the single amplitude thresholding method (SATM), the double amplitude thresholding method (DATM), and the power detection method (PDM). The authors of [51] also demonstrated WDM's robust performance when subject to a wide range of noise as signal-to-noise ratios (SNR) reduce. Furthermore, even though not explicitly discussed, the WDM has the potential for near real-time applications [49].

To summarize the discussions, it is noticed that simple threshold based detection methods are intuitive in principle and easy to implement. This is echoed by its popularity including commercial realizations of the algorithms. However as pointed out earlier, the detection results are variable and subjective to users, in addition to high false alarm rates. Other than these direct thresholding of the recorded neural waveforms as a function of time, the idea of thresholding was also an important part of

several other approaches discussed above, such as the threshold applied to wavelet coefficients in [42] and [43], a higher than usual threshold to gather spikes as the ground truth in [38], the threshold applied to the output of MACE filter in [39], the threshold for selecting potential neural spikes in [45] and [44], two thresholds used in MRGLRT in [47] and [48], and the threshold used for separating neural spikes from noise in [49]. It is worth pointing out that several algorithms rely on a Gaussian noise assumption to make an optimal detection statement. On one hand, it gives users some assurance of optimality, but unfortunately, noise profile is rarely Gaussian in recorded neural waveforms.

In this paper, we focus on developing a new spike detection algorithm aiming at providing robust detection performance with high detection rate and low false alarm. The goal is to alleviate subjectivity and variability in detection results, and make the algorithm near real-time. In doing so, we made use of the observation that a sharp rise of neural waveform signifying the onset of a neural spike in a 1-D neural signal is similar in characteristic to an edge in a 2-D image. Therefore, our proposed algorithm is a wavelet based approach, inspired by image edge detection. In [52], an edge detection algorithm makes use of a property in wavelet transform coefficients that the wavelet transform coefficients of image edges usually have higher magnitudes than the coefficients from noise. As shown later in this paper, the wavelet coefficient magnitudes of neural recordings preserve similar properties with a properly selected wavelet function: coefficients of neural spikes have higher magnitudes than those coefficients of noise.

Our proposed algorithm utilizes continuous wavelet transform as that in WDM [49] and [50], however with different wavelet functions in the respective implementations. Another major difference between the two algorithms is that while WDM performs detection at individual wavelet scales prior to fusing the results from multiple levels for a final spike detection, our approach fuses wavelet transforms from multiple scales first at each scale level and then perform a single detection by hypothesis

testing. By doing so, we have taken advantage of continuous wavelet transform over discrete wavelet transforms to avoid possible adverse effects due to coarse wavelet scale sampling, and we only introduce one free parameter, which in turn helps reduce the subjectivity of the algorithm.

4.2 Background on Wavelet Transform Based Spike Detection

A wavelet, $\psi(t) \in L^2(R)$, is a function of *zero average* [53], i.e.,

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0, \tag{4.1}$$

and *finite energy* defined by

$$\int_{-\infty}^{+\infty} \psi(t)^2 dt = 1. \tag{4.2}$$

The wavelet transform of a signal $x(t) \in L^2(R)$ is defined by

$$Tx(a,b) = \int_{-\infty}^{+\infty} x(t) \frac{1}{\sqrt{a}} \psi(\frac{t-b}{a})dt, \tag{4.3}$$

where $Tx(a,b)$ denotes the wavelet transform of $x(t)$, a is the scale factor and b represents time translation. From a different perspective, the correlation coefficient between a wavelet $\psi(t)$ and a signal $x(t)$ is defined by

$$R(\tau) = \int_{-\infty}^{+\infty} \psi^*(t+\tau)x(t)dt, \tag{4.4}$$

where $\psi^*(t)$ is the complex conjugate of $\psi(t)$.

It is well known that the correlation coefficient identifies the strength of relevancy between $\psi(t)$ and $x(t)$. Inspecting (4.3) and (4.4), it is easy to note that a wavelet transform also reflects on the same property between $\psi(t)$ and the signal under consideration.

Given the wide range of wavelet families and their unique features, it is important to select a suitable wavelet function for spike detection. It is noted in [54] that the waveforms of extracellular neural action potentials typically appear mono-phasic, bi-phasic and even tri-phasic. The research by [55] proves that the action potential waveforms of single units in human peripheral nerves also consists of such three kinds of waveforms. Since mono-phasic can be viewed as a building block of bi/tri-phasic waveforms, and the latter can be represented approximately by a superposition of mono-phasic waveforms, in this paper we focus on detection using an approximation of a mono-phasic wavelet function. The wavelet function"coiflets" was selected and used for neural spike detection in [44], [45], and [43]. It is also chosen in this paper. Figure 4.1 depicts the "bi-orthogonal 1.5" used in [49] and [50], and the "coiflets" wavelet function used in this paper and other papers mentioned above. We chose "coiflets" based on the following considerations. When the time support of the wavelet function matches the duration of one phase of a neural waveform, the corresponding wavelet transform coefficients become high. As such, the mono-phasic wavelet function is also able to generate high wavelet transform coefficients at one phase of the bi/tri-phasic neural spikes waveform. But the waveforms of noise usually do not resemble the wavelet function. Therefore the coefficients from noise have small or close to zero magnitudes. By inspecting waveforms corresponding to high wavelet transform coefficients, we can detect neural spikes even though they may have different phases.

Given the characteristics of the "coiflets" wavelet, and also according to (4.3) and (4.4), the following observations are utilized in the development of the proposed spike detection algorithm in this paper. Since real neural spike waveforms resemble that of the chosen wavelet albeit a difference in sign, it is expected that the absolute value of the wavelet coefficients for these waveforms are nontrivial. On the other hand, typically background noise signals have zero mean, and thus, wavelet coefficients corresponding

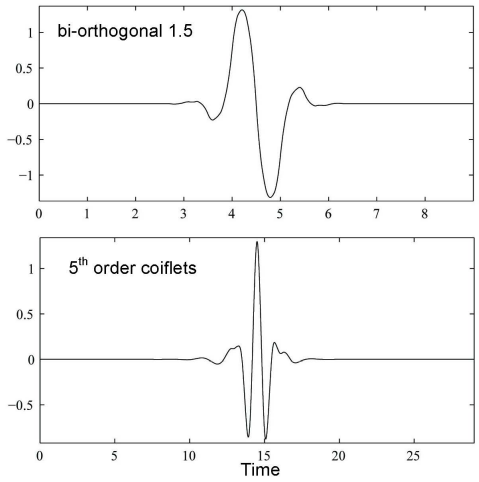


Figure 4.1: The Wavelet functions: "bi-orthogonal 1.5" (top) used in WDM and 5^{th} order "coiflets" (bottom) used in this paper and in [44], [45], and [43].

to noise are expected to fluctuate around zero. With such notion in mind, the guiding principle of spike detection is to distinguish the transform coefficients of neural spike waveforms from background noise. Additional considerations are also given to the development of a robust and accurate spike detection procedure in the next section.

For computing the wavelet transform using (4.3), it is customary to perform the integration within a finite time window, instead of the entire recording time horizon. In the present paper, we use N to denote the total number of waveform samples within such a finite time window. The length of the window is denoted by J as illustrated in Figure 4.2, where J is 100 ms, N is 2400 for a sampling rate of 24kHz in this study unless otherwise stated.

There are two control parameters in the chosen wavelet transform: the time translation factor b and the scale factor a. When applying the transform to a neural waveform in a given observation window containing N neural waveform samples, the time translation, b, is selected from

$$b \in B = \{0, 1, \cdots, N-1\}. \tag{4.5}$$

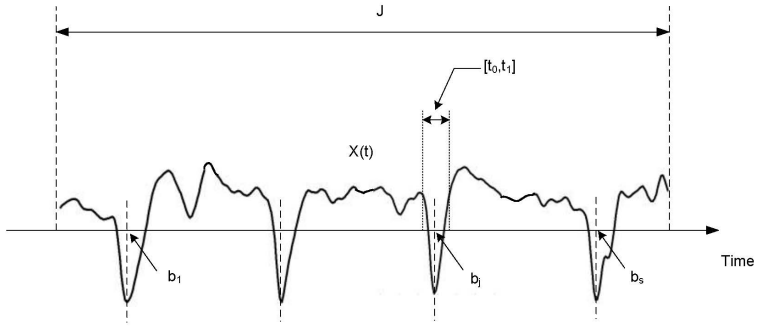


Figure 4.2: The integration window for computing wavelet coefficients and illustration of finding spike instants: b_j, $j = 1, 2, \cdots, s$, represent the instants of neural spike peaks.

The scale factor a, on the other hand, determines the support of the wavelet function. When applied to spike detection in neural recordings, it is chosen to represent the width of a neural spike, which is chosen between 0.5ms and 1.5ms in this paper as in [49] and [50], namely,

$$a \in A = \{0.5, 0.6, \cdots, 1.5\}. \tag{4.6}$$

The wavelet transform discussed in the above implies that the proposed new spike detection algorithm is based on a continuous wavelet transform. For the discrete wavelet transform, the i^{th} scale, α_i, and the $(i+1)^{th}$ scale, α_{i+1}, are related by $\alpha_{i+1} = \alpha_i * 2$. In this paper, the continuous wavelet transform is implemented as follows: the i^{th} scale, a_i, and the $(i+1)^{th}$ scale, a_{i+1}, are related by $a_{i+1} = a_i + 0.1ms$ as shown in (4.6). Both discrete and continuous wavelet transforms measure the similarity of a signal and the wavelet function. However discrete wavelet functions dilate exponentially as a function of the sampling scale level, much faster than their continuous counter parts. And thus, continuous wavelet transform is used in the development of the proposed detection algorithm in this paper.

4.3 Spike Detection Based on Multiscale Correlation of Wavelet Coefficients

As discussed above, this paper proposes a new spike detection algorithm based on continuous wavelet transform. It takes a multiscale approach by first calculating the wavelet coefficients at each scale, and correlates (by multiplication) wavelet coefficients from multiple scales, and then perform a hypothesis test for spike detection. This approach to detection is referred to as the multiscale correlation of wavelet coefficients (MCWC) method in this paper. It is motivated by a robust image edge detection algorithm [52] where in this case, a sharp spike is considered a 1-D edge. What follows is a step by step development of the proposed MCWC algorithm.

4.3.1 Computing Normalized Correlation of Wavelet Coefficients

Consider a neural waveform $x(t)$. Let J be the width of the observation window of the waveform under consideration which is used as the integration interval in the calculation of wavelet coefficients (refer to Figure 4.2). And let N be the number of samples in the observation window J. It is to be used in the calculation of the correlation of the wavelet coefficients. Apply "5^{th} order coiflets" wavelet transform to the neural waveform, $x(t)$, over the window of width J that contains N samples. With parameter sets $\{a_i\}$ and $\{b_j\}$ chosen as discussed above to reflect characteristics of neural spikes, i.e., $\{a_i\} = \{0.5, 0.6, \cdots, 1.5\}$, and $\{b_j\} = \{0, 1, 2, \cdots, N-1\}$, we obtain

$$Tx(a_i, b_j) = \int_J x(t) \frac{1}{\sqrt{a_i}} \psi(\frac{t - b_j}{a_i}) dt. \tag{4.7}$$

Let S be the number of sampling scales in a continuous wavelet transform, and let $r_S(a_i, b_j)$ be the correlation of wavelet coefficients among S sampling scales.

For each $\{a_i\} = \{0.5, 0.6, \cdots, 1.5\}$ and $\{b_j\} = \{0, 1, 2, \cdots, N-1\}$, $r_S(a_i, b_j)$ is obtained from the product of the wavelet transform of $x(t)$ across all S levels, i.e.,

$$r_S(a_i, b_j) = \prod_{k=0}^{S-1} Tx(a_{i+k}, b_j). \tag{4.8}$$

As can be seen from (4.8), $r_S(a_i, b_j)$ is a measure of the strength of resemblance between the wavelet function $\psi(t)$ and the neural signal $x(t)$ at each scale level a_i and location b_j. It does so by first computing the wavelet coefficient $Tx(a_{i+k}, b_j)$ at one scale, and then the strength of resemblance is enhanced by a product among all sampling scales S. As such, $r_S(a_i, b_j)$ defined by (4.8) can potentially produce a more pronounced separation of the coefficients corresponding to neural spikes from those corresponding to noise. Or in other words, this product can potentially reinforce the presence of neural spikes, while it is reduced if $x(t)$ contains mostly noise. The strengthened separation of signal and noise by a product of wavelet coefficients across multiple levels was also reported in [56] and [57] for signal and image processing applications. Additional example of making use of multiplication of multiple wavelet coefficients to enhance image edge detection can be found from [52]. Even though utilized differently, the same principle of making use of multiple scale wavelet coefficients was also identified and implemented by [49] for spike detection.

Once $r_S(a_i, b_j)$ is obtained, it should be normalized so that the correlation of coefficient measure is still based on the original neural signal scale level, not on different sampling scale levels. This makes the correlation of coefficient measure comparable with the wavelet coefficient.

Let the power normalized correlation of wavelet coefficients be denoted by $r'_S(a_i, b_j)$. It is obtained based on the power of $r_S(a_i, b_j)$ defined in (4.9), and the power of $Tx(a_i, b_j)$ defined in (4.10). The former involves all S sampling levels, while the latter only involves a single base level measure of similarity. And therefore, the normalization denoted by (4.11) is necessary.

$$P_{r_S}(a_i) = \sum_{j \in J} r_S(a_i, b_j)^2, \tag{4.9}$$

$$P_{Tx}(a_i) = \sum_{j \in J} Tx(a_i, b_j)^2, \tag{4.10}$$

$$r'_S(a_i, b_j) = r_S(a_i, b_j) \times \sqrt{\frac{P_{Tx}(a_i)}{P_{r_S}(a_i)}}. \tag{4.11}$$

4.3.2 Spike Detection using Hypothesis Testing

With the well defined power normalized correlation of coefficients measure $r'_S(a_i, b_j)$ in (4.11), neural spikes embedded in a noisy neural recording are now considered for detection using a binary hypothesis testing. Actually this entails both the declaration of the existence of a spike in a noisy recording and the specification of spike timing.

Let H_0 be the null hypothesis that within a small window $[t_0, t_1]$ belonging to J (refer to Figure 4.2), $x(t)$ does not contain any neural spikes, and let H_1 be the alternative hypothesis that within the the small window $[t_0, t_1]$, $x(t)$ contains a spike at b_j. Or in other words:

H_0: $x(t)$ contains no spikes in the small window $[t_0, t_1]$ belonging to J under consideration (Figure 4.2) ,

H_1: $x(t)$ contains a spike at b_j in the small window $[t_0, t_1]$ belonging to J under consideration (Figure 4.2) .

Specifically, H_0 holds, or no spike is detected if

$$\left| \frac{r'_S(a_i, b_j)}{Tx(a_i, b_j)} \right| \leq 1 \tag{4.12}$$

and H_1 holds, or a spike is detected if (4.13) is satisfied,

$$\left| \frac{r'_S(a_i, b_j)}{Tx(a_i, b_j)} \right| > 1 \tag{4.13}$$

Figure 4.3 illustrates the principle of spike detection proposed in this paper.

The next question is naturally where the spike instances are. In our spike detection problem formulation, the spike instances are denoted by those b_j's that result in a positive hypothesis test of H_1. Refer to Figure 4.3, notice that multiple b_j's in the vicinity of a neural spike can possibly pass H_1 test while corresponding to the same potential spike. In this paper, we choose the one and only b_j that gives rise to the most positive test result of H_1.

Let H_1 hold inside the small interval $[t_0, t_1]$ at specific points of b_j where there can possibly be more than one b_j's. Let t_d be the instant of a spike within $[t_0, t_1]$ (refer to Figure 4.3). Then a spike is detected at t_d within $[t_0, t_1]$ from the following

$$t_d = \arg \max_{\substack{b_j \in [t_0, t_1] \\ a_i \in \{0.5, \cdots, 1.5\}}} \left| Tx(a_i, b_j) \right|. \tag{4.14}$$

The width of the spike detected at t_d, τ, is estimated by

$$\tau = \arg\max_{a_i \in \{0.5, \cdots, 1.5\}} \left| Tx(a_i, t_d) \right|. \tag{4.15}$$

This effectively implies that no other spikes exist within a distance of τ from t_d.

4.3.3 Detection Principle: Adaptive Thresholding

We are now ready to demonstrate that the proposed MCWC detection algorithm actually is an adaptive thresholding method. The threshold level changes as the signal-to-noise ratio or the noise covariance varies.

First, consider the case of $S = 2$. Re-write (4.13) into the following by assuming that $Tx(a_i, b_j)$ is non-zero, which is commonly true.

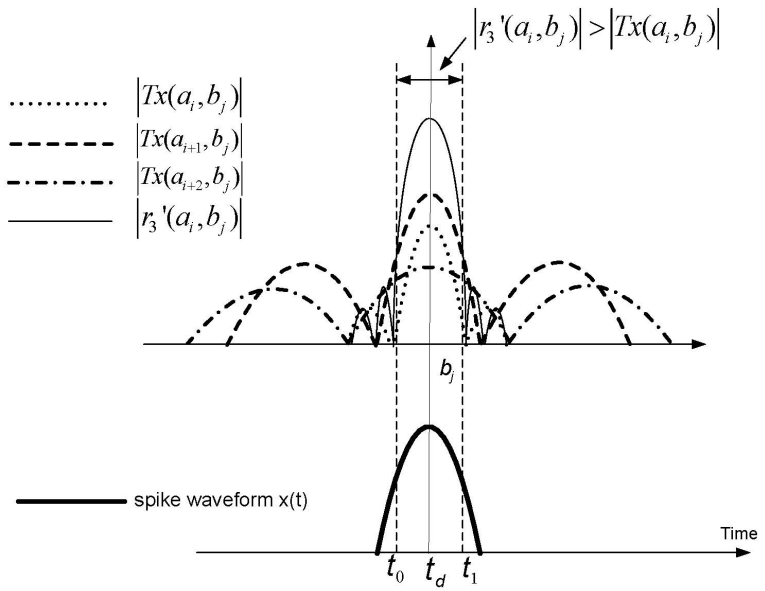


Figure 4.3: Demonstration of MCWC detection principle: The multiplication of multi-scale wavelet coefficients enhances the detection of a neural spike. Inequality $|r'_S(a_i,b_j)| > |Tx(a_i,b_j)|_{S=3}$ for $t \in [t_0,t_1]$ indicates that hypothesis H_1 passes the test in this interval. The detection of a neural spike at time instant t_d is declared.

$$\left| Tx(a_i,b_j)Tx(a_{i+1},b_j)\sqrt{\frac{\sum_{j\in J} Tx(a_i,b_j)^2}{\sum_{j\in J} Tx(a_i,b_j)^2 Tx(a_{i+1},b_j)^2}} \right| > |Tx(a_i,b_j)|. \tag{4.16}$$

As discussed in the previous section, once we have identified the b_j's that satisfy criterion (4.13) or equivalently (4.16), we can then apply (4.14) and (4.15) to determine spike time and spike width. However, to gain additional insight, further derivations are performed below. Define $\overline{T}x(a_i,S)|_{S=2}$ as in (4.17),

$$\overline{T}x(a_i,S)|_{S=2} \triangleq \frac{\sum_{j\in J} Tx(a_i,b_j)^2 Tx(a_{i+1},b_j)^2}{\sum_{j\in J} Tx(a_i,b_j)^2}. \tag{4.17}$$

Re-arranging (4.16) by substituting the newly defined term $\overline{T}x(a_i,S)|_{S=2}$, we obtain the following new form of spike detection criterion,

$$Tx(a_{i+1},b_j)^2 > \overline{T}x(a_i,S)|_{S=2}. \tag{4.18}$$

Under a similar assumption to that in [49] at the i^{th} scale level, $\{Tx(a_i,b_j)\}$ are independent Gaussian random variables and comply with the following distributions,

$Tx(a_i,b_j) \sim N(0,\sigma^2)$ given H_0 holds, which implies that given H_0, $Tx(a_i,b_j)$ complies with a Gaussian distribution with zero mean and σ^2 as its variance.

$Tx(a_i,b_j) \sim N(\mu,\sigma^2)$ given H_1 holds, which implies that given H_1, $Tx(a_i,b_j)$ complies with a Gaussian distribution with μ as its mean and σ^2 as its variance.

Define a weighting coefficient w_i as shown below,

$$w_i \triangleq \frac{Tx(a_i,b_j)^2}{\sum_{j\in J} Tx(a_i,b_j)^2} = \frac{Tx(a_i,b_j)^2/\sigma^2}{\sum_{j\in J}\left[Tx(a_i,b_j)^2/\sigma^2\right]}. \tag{4.19}$$

Let $P(H_0)$ be the prior probability associated with hypothesis H_0 and $P(H_1)$ be that with hypothesis H_1. Then for real neural recordings, it is reasonable to assume that $P(H_0) \gg P(H_1)$ since majority of the time course of a neural recording corresponds with noise [49]. Given that H_0 holds, then $Tx(a_i,b_j)$ complies with $N(0,\sigma^2)$. Consequently the new variable $Tx(a_i,b_j)^2/\sigma^2$ complies with the chi-square distribution with 1 degree of freedom, i.e., $Tx(a_i,b_j)^2/\sigma^2 \sim \chi_1^2(1)$. Therefore $E\left[Tx(a_i,b_j)^2/\sigma^2\right] = 1$ remains valid most of the time. It also is approximately true if H_1 holds but the mean μ in $Tx(a_i,b_j)$ is relatively low. Since $E\left[Tx(a_i,b_j)^2/\sigma^2\right] \approx 1$, then $\sum_{j\in J}\left[Tx(a_i,b_j)^2/\sigma^2\right] \approx M$, where M is the cardinalities of J. Thus the weight $w_i \approx 1/M$.

Since $Tx(a_i,b_j)$ for a given scale may be viewed as an independent Gaussian variable with zero mean most of the time especially when it corresponds with noise, the maximum likelihood estimate of the variance of the noise sequence $\{Tx(a_{i+1},b_j)\}$ is $\frac{1}{M}\sum_J Tx(a_{i+1},b_j)^2$. To see that, refer to (4.19) and that $w_i \approx 1/M$. Therefore the threshold $\overline{T}x(a_i,S)|_{S=2}$ defined in (4.18) can be viewed as an approximation of the

maximum likelihood estimation of the noise variance since $P(H_0) \gg P(H_1)$. When $Tx(a_{i+1},b_j)^2 > \overline{T}x(a_i,S)|_{S=2}$, it implies that the correlation between the neural waveform and the wavelet is greater than the noise variance, and therefore, a neural spike is likely to be present, and that H_1 is true.

For $S \geq 3$, (4.13) is re-written as below

$$\begin{aligned}
&\left|Tx(a_i,b_j)\cdots Tx(a_{i+S-1},b_j)\right| \sqrt{\frac{\sum_{j\in J} Tx(a_i,b_j)^2}{\sum_{j\in J}\left[Tx(a_i,b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2}} \\
&> \left|Tx(a_i,b_j)\right| \\
&\Rightarrow \\
&\left[Tx(a_{i+1},b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2 > \frac{\sum_{j\in J}\left[Tx(a_i,b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2}{\sum_{j\in J} Tx(a_i,b_j)^2}. \quad (4.20)
\end{aligned}$$

Let the right hand side of inequality (4.20) be

$$\begin{aligned}
\overline{T}x(a_i,S) &\triangleq \frac{\sum_{j\in J}\left[Tx(a_i,b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2}{\sum_{j\in J} Tx(a_i,b_j)^2} \\
&= \sum_{j\in J}\left[w_i Tx(a_i+1,b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2. \quad (4.21)
\end{aligned}$$

Then (4.20) becomes

$$\left[Tx(a_{i+1},b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2 > \overline{T}x(a_i,S). \quad (4.22)$$

Similar to the analysis conducted for the case of S=2, the weight factor w_i in (4.19) appears in the newly defined threshold $\overline{T}x(a_i,S)$ in (4.21), and it approaches $1/M$. This indicates that $\overline{T}x(a_i,S)$ can be viewed as a sample mean of $[Tx(a_{i+1},b_j)\cdots Tx(a_{i+S-1},b_j)]^2$, which is a uniformly minimum-variance unbiased estimate of the expectation of $\left[Tx(a_{i+1},b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2$. Again, using similar arguments as in the case of $S=2$ as well as that in [49], we have $P(H_0) \gg P(H_1)$. Therefore, $\overline{T}x(a_i,S)$ may be viewed as average noise of $\left[Tx(a_{i+1},b_j)\cdots Tx(a_{i+S-1},b_j)\right]^2$.

Next, let $Tx_s(a_i,b_j)$ and $Tx_n(a_i,b_j)$ denote the wavelet coefficients corresponding to neural spikes and noise, respectively. For real neural recordings of reasonable

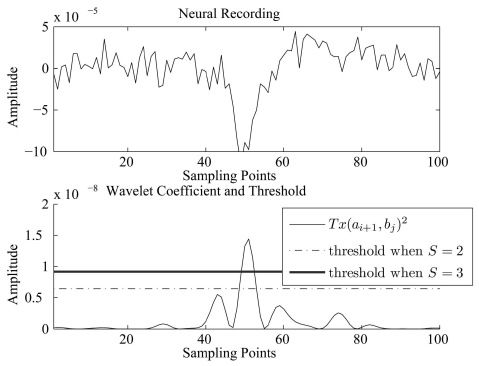
quality, it is generally true that $Tx_s(a_i,b_j) > Tx_n(a_i,b_j)$ for any one of the multiple sampling scales. It is therefore easy to obtain that $\prod_i^{i+S-1} Tx_s(a_i,b_j) / \prod_i^{i+S-1} Tx_n(a_i,b_j) > Tx_s(ai,b_j)/Tx_n(a_i,b_j)$. It means that the product of multi-scale wavelet coefficients enhances the difference between spikes and noise. In turn, this shows that if inequality (4.20) holds, then a spike is most likely to be present. Therefore it is reasonable to declare a spike if (4.20) holds.

Note that S is the only free parameter to be determined by the user as discussed throughout this paper. Unlike simple waveform magnitude thresholding methods, which is usually set by the user via visual inspection, the parameter S can actually be chosen in consideration of the signal-noise-ratio (SNR) or noise co-variance as well as neural firing rates and other physical properties of neural spikes. When the SNR is low or the noise covariance is high, it is difficult to distinguish the wavelet coefficients between noise corrupted neural spike waveforms and noise waveforms. Therefore increasing S will enhance the difference in wavelet coefficients corresponding to spike and noise, and help differentiate spikes from noise. On the other hand, increasing S usually reduces the overall detected spike rates. Therefore, neural spike characteristics such as neuron firing rate, physical features of the rising and falling edges of a neural spike, etc. should be taken into consideration, which can assist the process of choosing an appropriate multiple scale sampling parameter S as will be discussed and demonstrated later in our simulation studies.

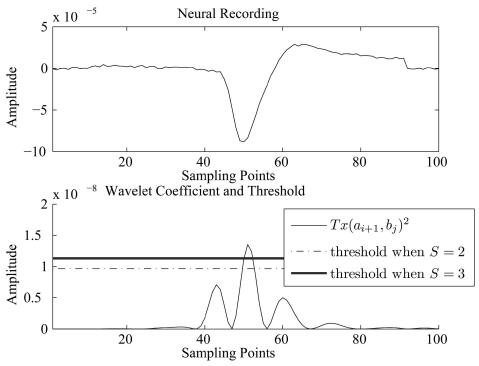
Figure 4.4a and Figure 4.4b illustrate how the adaptive threshold values $\overline{T}x(a_i,S)$ in (4.21) vary as a function of the SNR or the noise co-variance and the scale level S.

In addition to the above discussion on the nature of adaptive thresholding for the MCWC algorithm, note also that in case of low frequency background noise, according to the *zero average* property (4.1) and the compactness of the wavelet function "coiflets", the low frequency noise has little contribution to the wavelet coefficients when MCWC is used for detection. This demonstrates the robustness of the MCWC

algorithm to low frequency noise. However, for the strictly thresholding based detection, slowly fluctuating noise may change the magnitude of the neural waveforms and alter the presence of a real neural spike, which may be overshadowed by noise or may only be detected reliably if a high threshold value is placed. Therefore, for strictly thresholding based detections, manual adjustment of threshold values is needed in order to obtain robust detection results. This is in contrast to the nature of automatically adjusting threshold values in MCWC to maintain robustness of the algorithm.



(a) Detection thresholds at two S levels when neural signal has a low SNR or high noise covariance.



(b) Detection thresholds at two S levels when neural signal has a high SNR or low noise covariance.

Figure 4.4: Illustration of adaptive threshold in MCWC, which varies with SNRs or noise covariances and S values. From (a) and (b), at a given SNR, when S is high, the threshold levels are high and vice versa. The SNR in (a) is lower than that in (b), therefore at a given S, the threshold level is higher when SNR is higher.

4.4 Detection Performance Evaluation

In this section, we provide detailed performance evaluation on the proposed multiscale correlation of wavelet coefficient (MCWC) algorithm. While comparisons are conducted for a few algorithms including direct thresholding, our focus is on comparing MCWC and WDM since both are wavelet based, and WDM has been shown outperforming several other approaches [49] and [51]. It is noted that WDM detects neural spikes using Bayesian hypothesis test at each wavelet transform scale and then fuse all the detection results from all scales [49] to form the final detection. On the other hand, MCWC first correlates the wavelet transform coefficients among multiple scales and

then run the spike detection hypothesis on the correlated coefficients once to form the final detection results.

Performance evaluations were carried out using 23 data sets in this study, which include both artificially generated neural spikes and real neural spikes recorded from rat's motor cortex.

All 18 artificially generated neural waveforms span 50 seconds, and they are sampled at 20KHz. Half of the 18 artificial data sets (A1-1 to A1-9) were obtained with 1dB signal-to-noise ratio (SNR), while the other half (A2-1 to A2-9) with 10dB SNR.

Each artificial neural data set was generated the same way as in [58]. The simulated neural data sets were obtained by activating three types of neurons in the program: target neuron (used as truth), correlated neuron (its waveforms are regarded as correlated interference), and uncorrelated neuron (its waveforms are regarded as uncorrelated interference). The spike generating program was used to simulate neural waveforms of a single channel from multiple channel recording arrays. In the simulation program, the refractory period was set to 1ms.

For the 5 real neural data sets, the data length was 5 seconds for 2 data sets (R1-1 and R2-1) and 50 seconds for the other 2 sets (R1-2 and R2-2). The real neural data set R2-L is 10 minutes long. The real neural waveform were sampled at 24KHz. For obtaining real neural spikes, the intended task for the rat was for him to press levers after light and sound cues for food reward. But the data used in this study were recorded while the rat was freely moving about in a Skinner box.

In our implementation, the SNR was calculated as follows,

$$SNR(dB) = 20\log 10\left(\frac{A_{signal}}{A_{noise}}\right), \tag{4.23}$$

where A_{signal} and A_{noise} are signal and noise root mean square (RMS) amplitudes, respectively. For the artificial data set, SNR is calculated before mixing spike data with noise. For real neural recording data, the SNR is estimated based on detected spikes (as signal) and noise (recorded waveform minus signal). For example, the SNR of data set R1-1 is estimated with signal and noise root mean square (RMS) amplitudes as in (4.23): 1) Use the detected spike waveforms to estimate the signal RMS amplitude, 2) Deduct spike waveforms from the neural recording to obtain noise RMS amplitude. The estimated SNR for data R1-1 is 4.3dB.

When applying the WDM algorithm, one is required to select a parameter L that determines the cost ratio between the false alarms and missed detections. Once L is selected, different thresholds used in WDM at different scales are determined, respectively and the detection results at all scales are obtained. The final detention result is the combination of the detection results at each scale. Therefore the selection of L affects detection performance [49]. In all our later simulations, the WDM algorithm was implemented using the code provided online by [59].

As will be shown in the next section when we discuss the issue of real-time implementations, the observation window length J in (4.7) is a parameter that can be chosen from a broad range of values, without major degradation in performance. Therefore, we consider S, the number of down sampling scale levels, as the only tuning parameter in the MCWC algorithm that may be optimized to gain detection performance. In principle, the larger the S parameter, the more likely that the MCWC picks up less neural spikes but only those with higher SNRs, and vice versa. Thus, S should be chosen at an appropriate level to reflect the physical nature of neural firing rates and features of neural spikes to ensure the robustness of the algorithm.

4.4.1 Comparison of Detection Performance among Thresholding, MCWC, and WDM using Artificial Neural Data Sets

In this section, we compare detection performances among MCWC, WDM, and thresholding provided in Plexon's Offline Sorter. Eighteen artificial data sets, A1-1 to A1-9, and A2-1 to A2-9 are used. The thresholding method used was the "Signal Energy" in Offline Sorter as described below,

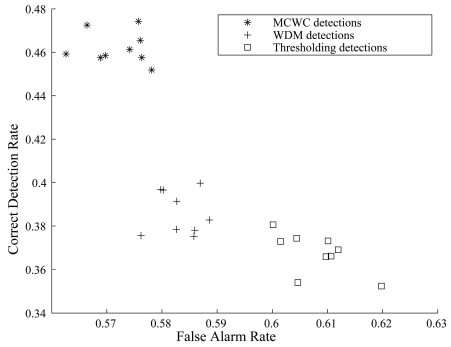
$$energy(i) = \frac{1}{W} \sum_{j=i-W/2}^{i+W/2} v^2(j), \tag{4.24}$$

where $v(j)$ is the raw neural recording at time j. W is the window width used in averaging. In this paper, $W = 7$ is used.

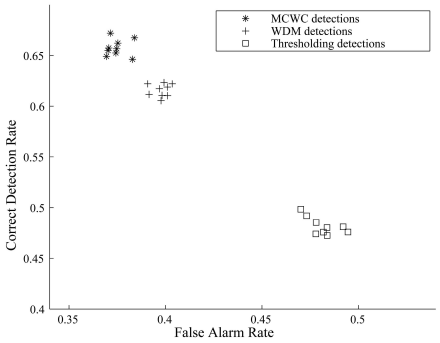
To make results comparable, we manually selected the threshold value in Offline Sorter such that the total number of detected neural spikes was close to that of the ground truth. The L and S parameters in WDM and MCWC, respectively, were chosen similarly such that the total number of detected spikes by each algorithm was close to the ground truth. To remove low frequency noise, the artificial data sets used in thresholding detection were filtered with a band-pass butterworth filter, which usually enhances its performance. The pass band is [100, 6000]Hz. However, the data used in WDM and MCWC were not filtered. The receiver operating characteristics (ROC) as a measure of detection performance are shown in Figure 4.5a and Figure 4.5b. Based on the ROCs, the MCWC outperformed WDM and thresholding at the two tested SNR levels.

4.4.2 Evaluation of Algorithm Parameter Setting using Artificial Neural Data Sets

In this section, we compare MCWC and WDM detection performances using artificial neural data sets from A1-1 to A2-9. Our goal is to examine the robustness and impact of the L and S parameters used in the WDM and MCWC, respectively. As before, we used the observation window $J = 100$ms for both MCWC and WDM. Figure 4.6a and



(a) Detection performance with data sets A1-1 to A1-9 (SNR=1dB).



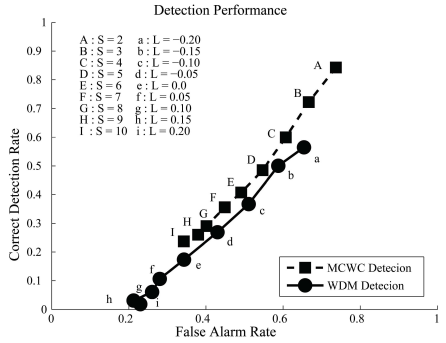
(b) Detection performance with data sets A2-1 to A2-9 (SNR=10dB).

Figure 4.5: ROC performance for MCWC, WDM and Plexon thresholding: MCWC has a better performance than WDM and Plexon thresholding in low and high SNR scenarios.

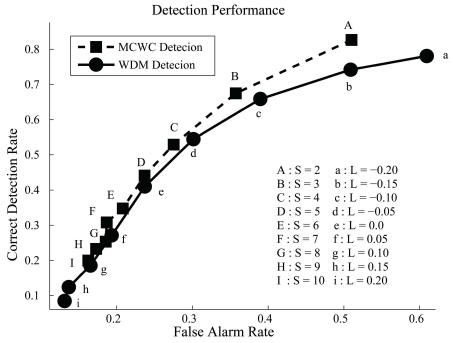
Figure 4.6b are ROC performances for MCWC and WDM at different SNRs and different parameter value settings. These figures show that MCWC generally outperforms WDM. As noted from the ROCs generated for WDM and MCWC, the scale levels L and S do affect the detection results in a manner noted in the discussions in earlier sections of this paper. Specifically, As discussed in [49], increasing L implies an increase of the cost of false alarms. Since the detection principle of WDM is based on setting a proper threshold to minimize the total detection cost, with an increase in the cost of false alarms, the false alarm rate should decrease. As for MCWC, when S increases, the adaptive threshold used in MCWC increases so that the detected neural spikes may be considered present in neural recording with high SNRs, consequently the false alarm rate decreases.

4.4.3 Evaluation using Real Neural Cortical Waveforms without Spike Waveform Verification by Human

In this test and what follows next (section 4.4.4), we used real microarray recordings from rat's motor cortex digitized at 24kHz using TDT RX5 Pentusa Base Station (TDT, Inc). The data acquisition procedure was similar to [60]. Four sets of neural recordings (R1-1, R1-2, R2-1, and R2-2) from 2 rats were examined. In this section, we compare



(a) Detection performance with data sets A1-1 to A1-9 (SNR=1dB).



(b) Detection performance with data sets A2-1 to A2-9 (SNR=10dB).

Figure 4.6: Detection results for MCWC and WDM as a function of *S* and *L*. MCWC has a slightly better detection performance over the entire ranges of *S* and *L* for the artificial data sets.

detection performance for thresholding, WDM and MCWC. The scale parameter *a* for WDM and MCWC was set between 0.5 and 1.5 ms.

The major difference between this section and section 4.4 is that we did not provide "ground truth" when evaluating algorithm performances in this section while we did manual inspection on some real neural waveforms and used those as ground truth to conduct performance comparisons.

Without "ground truth" provided, the ROC curve as performance measure no longer is valid. Therefore we have developed other means of spike detection performance evaluation. Specifically, we examined the shape features of the spike waveforms, the spike duration, and the rising/falling rate of spike charge/discharge, as discussed in [61] and [62], and used them as alternative measures of detection performances. Spike sorting was also performed after detection using respective algorithms. As such, it allowed us to analyze neural waveform characteristics by grouping them together according to waveform similarity.

4.3.1 Selection of Detection Parameters

For comparing detection performances by WDM and MCWC, we need to properly choose the parameters L and S, respectively for each of the algorithms. To provide a ground of comparison with reasonable values of L and S, we created Tables 4.1 and 4.2 using the real data set R1-1. Data set R1-1 was also manually inspected for spikes and it was determined that the neural firing rate was about 40 Hz for the recorded neuron. The tables summarize detected spike firing rates as a function of different parameter levels by using each of the detection algorithms. A small segment of the waveforms from data set R1-1 is shown in Figure 4.7.

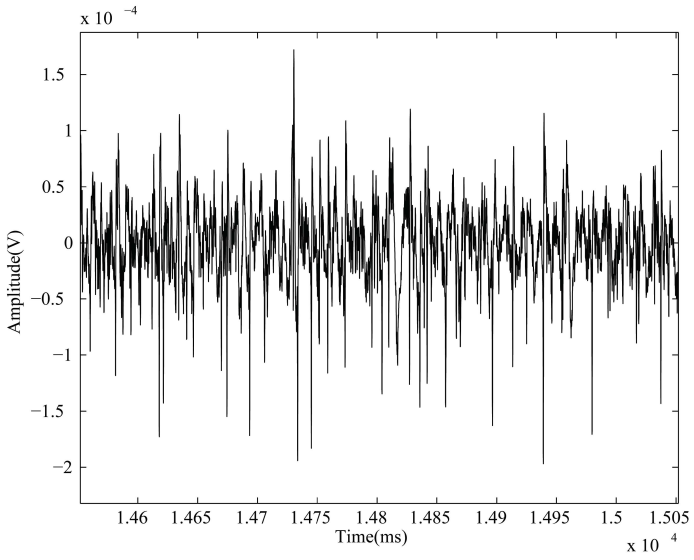


Figure 4.7: A small segment of waveforms from data set R1-1.

The same procedure was performed on R2-1 for selecting parameters L and S before analyzing respective performances of each algorithm using data set R2-1.

Table 4.1: Detected spike firing rates with different L values for WDM.

L	-0.2	-0.1	0	0.1	0.2	0.3	0.4
Firing rate (Hz)	240	160	115	82	68	52	37

Table 4.2: Detected spike firing rates with different S values for MCWC.

S	2	3	4	5	6	7	8	9	10
Firing rate (Hz)	157	110	81	63	54	45	40	34	30

By inspecting Table 4.1 and Table 4.2, S=8 and L=0.4 were chosen for MCWC and WDM, respectively in our later comparison of results. At those parameter settings, both detection algorithms resulted in neural firing rates at around 40Hz. When thresholding method was used in comparison, we selected the threshold value such that spike firing rate is about 40 Hz as well. By doing so, we have made all algorithm comparable.

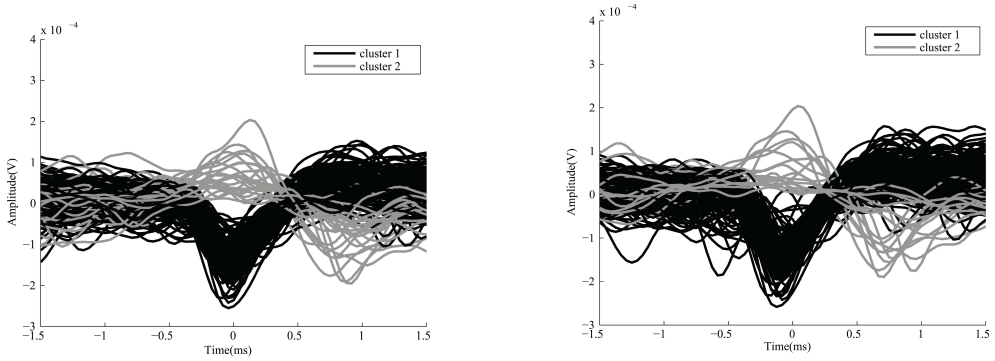
4.3.2 Detection Results using R1-1 and R2-1

To make the detection algorithms comparable for the next results using recorded neural waveforms, it is also necessary to align the detected spike waveforms with a common reference point. As in [63], in the following results, each detected spike waveform by thresholding, WDM, and MCWC was aligned by maximizing the correlations between detected spike waveforms. Once the alignment point is determined for each detected spike waveform, a segment of 3ms spike waveform with 1.5ms to the left and 1.5ms to the right from the alignment point is extracted and used in the following analysis.

Figure 4.8a and Figure 4.8b depict detected spikes using MCWC and WDM, respectively using R1-1. With the help of K-Means clustering, the MCWC detection results are classified into 2 clusters. The firing rate of cluster 1 is 34.6Hz, and 4.8Hz for cluster 2. By visual inspection of the waveforms and the duration about 2ms of those detected potential spikes in cluster 2, cluster 2 is classified into noise. The same analysis with K-Means clustering applied to WDM detections: the resulted WDM cluster 1 fired at 33.4Hz, while cluster 2 fired at 4Hz, which is classified into noise as well.

As can be seen from Figure 4.8a and Figure 4.8b, MCWC has a comparable detection performance with WDM for data set R1-1.

Detections by "signal energy" in Offline sorter with K-Means clustering are shown in Figure 4.9. By visual inspection, only the waveforms in cluster 1 and part of clusters 2 and 3 appear neural spike like but with a low rate. This indicates that thresholding method does not detect as many real spikes as WDM and MCWC did given a total number of detections, which in this case is a firing rate of 40Hz. Also from its detection principle, thresholding only used waveform amplitude, not other features as did in MCWC or WDM. Therefore it may be more sensitive to noise with a more pronounced noise cluster than MCWC or WDM. Therefore, from here on, we will focus on comparisons between WDM and MCWC.

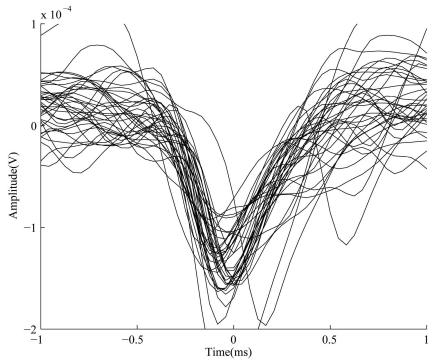


(a) Spikes detected by MCWC from data set R1-1. The firing rates of cluster 1 and 2 are 34.6Hz and 4.8Hz, respectively.

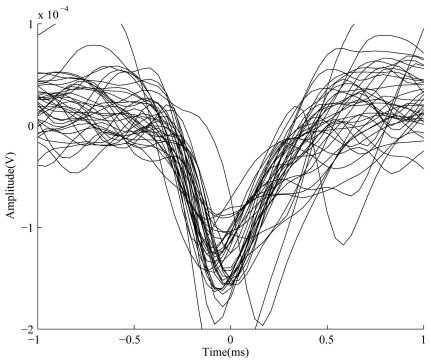
(b) Spikes detected by WDM from data set R1-1. The firing rates of cluster 1 and 2 are 33.4Hz and 4Hz, respectively.

Figure 4.8: Detection results for MCWC and WDM with data set R1-1. The detection results by MCWC and WDM are similar.

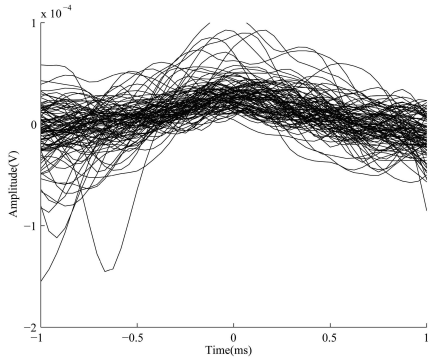
For the data set R2-1, we selected parameters for MCWC and WDM the same way as when we analysed data set R1-1. With the same detection performance measure, we find that both MCWC and WDM resulted in 3 clusters: 2 neural spike clusters and 1 noise cluster. The noise cluster rates are 3Hz and 7Hz for MCWC and WDM, respectively. Again, the results by MCWC and WDM are comparable for data set R2-1.



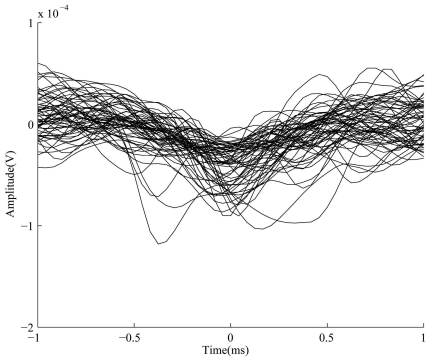
(a) Waveforms of detected spikes from R1-1 using Plexon thresholding with peak alignment.



(b) Plexon thresholding detected spike waveforms from R1-1: cluster 1 with firing rate 6.8Hz.



(c) Plexon thresholding detected spike waveforms from R1-1: cluster 2 with firing rate 21Hz.



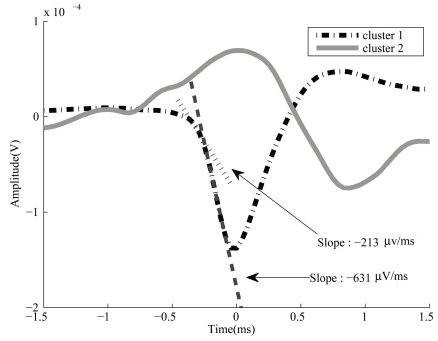
(d) Plexon thresholding detected spike waveforms from R1-1: cluster 3 with firing rate 13.4Hz.

Figure 4.9: Plexon thresholding detected spikes from R1-1. By visual inspection, the detected waveforms in cluster 1 and part of clusters 2 and 3 are neural spikes with high confidence while most waveforms in clusters 2 and 3 are possible false alarms with high rate.

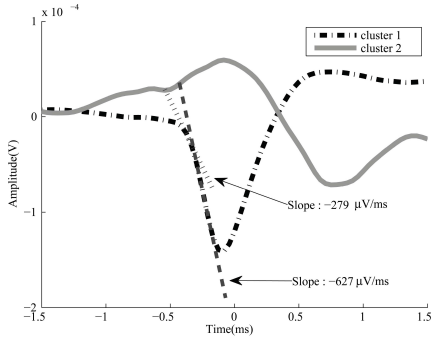
Next, based on the two separated clusters as shown in Figure 4.8a and Figure 4.8b for data set R1-1, we computed the average spike waveform for each cluster, and they were used in our analysis for inspecting the detected spike characteristics.

With the chosen parameter values of $S = 8$ and $L = 0.4$ for MCWC and WDM, respectively, and with the waveform alignment performed as discussed above, we are now in a position to evaluate spike detection performances of the algorithms using R1-1. Since detection rate and false alarm rates are no longer valid measures without ground truth, we examined the detected spikes by their physiological characteristics.

In this case, they include the duration of a neural spike (typically around 1ms) and the change rate of cortical neural action potentials during charging and discharging. According to [61], an action potential can be characterized by a very abrupt onset and a rapid change in membrane potential, and the change rate of cortical neural action potential is usually at 10mv/ms. It also indicated [62] that the spike waveform can be fitted by a straight line after the onset initiation. Based on these principles, we tried to qualitatively evaluate the characteristics of the detected spikes.



(a) The average waveform of MCWC detected spikes and its spike onset slopes with data set R1-1.



(b) The average waveform of WDM detected spikes and its spike onset slopes with data set R1-1.

Figure 4.10: The average waveforms of cluster 1 in MCWC and WDM are analyzed for their respective spike onset properties: the onset of a neural spike is a rapid change in membrane potential [61], and the rapid change portion of the spike waveform can be modeled by a straight line after the onset initiation [62]. The detection results of cluster 1 in MCWC and WDM are neural spikes with high confidence.

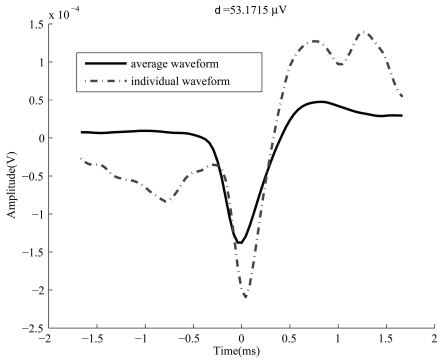
Figure 4.10a shows the averaged waveforms for the two spike clusters detected by MCWC for R1-1. We computed and illustrated the two straight lines to fit the main portion of the spike charging/discharging waveforms. From these figures, it is apparent that the spikes detected by MCWC have met all major characteristics of a neural spike - the spike duration is about 1ms, rapid change rate at the onset of a spike, and good straight line fits during charging/discharging of an action potential. Similar results were obtained and illustrated in Figure 4.10b for the WDM algorithm. These results show that MCWC has provided comparable detection results as WDM. By visual inspection, the detected neural spike cluster 2 in Figure 4.10a and Figure 4.10b do not preserve the

neural characteristics of a spike: 1ms duration and rapid slopes. Therefore the detected cluster 2 by MCWC and WDM was classified as noise.

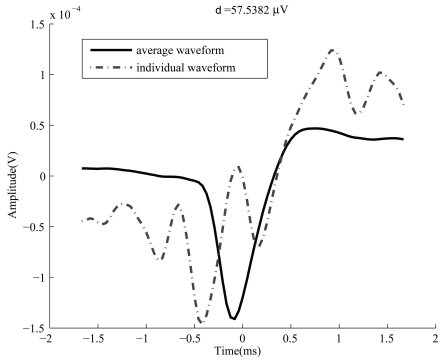
Next we look into another measure to evaluate variations in the detected spike waveforms to obtain a quantitative evaluation and comparison. The average waveform of cluster 1 in Figure 4.10a and Figure 4.10b were used to form the neural spike template waveforms. The distance between the template waveform and each detected waveform is defined as

$$d = \frac{1}{K}\sum_{i=1}^{K} |s(i) - t(i)|, \tag{4.25}$$

where d denotes the distance between the detected waveform and the template, K is the sample length of the template, $s(i)$ and $t(i)$ are the i^{th} element of the detected waveform and the template, respectively, and $|\cdot|$ denotes 1-norm.



(a) Averaged spike waveform and an individual spike waveform that has the largest distance measured by d in (4.25). Detection was performed by MCWC using data set R1-1.



(b) Averaged spike waveform and an individual spike waveform that has the largest distance measured by d in (4.25). Detection was performed by WDM using data set R1-1.

Figure 4.11: Comparison between averaged spike waveforms and an individual spike waveform which is the farthest from the average with the distance measured by (4.25). The spike waveform that is the farthest from the average using MCWC (in part (a)) appears more spike like than that detected by WDM (in part (b)).

For data set R1-1, the distances for individual spike waveforms were computed, and the waveforms with the largest distances are displayed in Figure 4.11a for MCWC and Figure 4.11b for WDM. Based on the distance measure (4.25), statistics of the

computed waveform to template distances were generated for both MCWC and WDM. Table 4.3a and Table 4.3b are the measured means and standard deviations (std) of the computed distances for the first cluster from MCWC and WDM, respectively. These results show that the mean and the standard deviation of the distance measure are slightly smaller for MCWC than the WDM.

	neural spike cluster 1	noise
rate (Hz)	34.6	4.8
mean of histogram distance (μV)	19.9	N.A.
std of histogram distance (μV)	11.4	N.A.

(a) MCWC detection with data set R1-1.

	neural spike cluster 1	noise
rate (Hz)	33.4	4
mean of histogram distance (μV)	22	N.A.
std of histogram distance (μV)	12	N.A.

(b) WDM detection with data set R1-1.

	neural spike cluster 1	neural spike cluster 2	noise
rate (Hz)	12.4	25.6	3
mean of histogram distance (μV)	9.4108	8.0419	N.A.
std of histogram distance (μV)	3.2693	3.0673	N.A.

(c) MCWC detection with data set R2-1.

	neural spike cluster 1	neural spike cluster 2	noise
rate (Hz)	8.6	25.2	7
mean of histogram distance (μV)	9.3171	7.7697	N.A.
std of histogram distance (μV)	3.4044	2.8188	N.A.

(d) WDM detection with data set R2-1.

Table 4.3: Statistics of the distance measurements using (4.25).

For data set R2-1, we analyzed MCWC and WDM detection results the same way as we analyzed data set R1-1. For each detected neural spike cluster, the average waveforms of WDM and MCWC detections preserve the neural spike characteristics. The statistics from the distances calculated from (4.25) for MCWC and WDM detection results are shown in Table 4.3c and Table 4.3d. Although for data set R2-1, the mean and the std for MCWC are a little higher than those of WDM. Note however that the

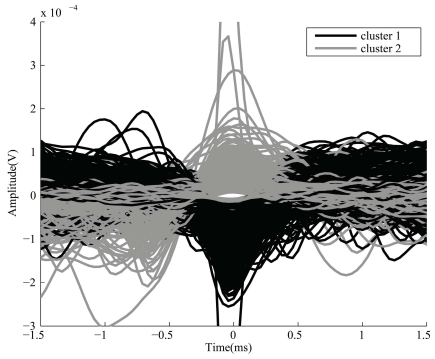
WDM has put more waveforms into the "noise" category and has a little lower detection rate than the MCWC. As a result, this may have improved the mean and std measures for WDM. With both test results using R1-1 and R2-1, since there is no significant difference measured by the mean and std, we consider the two algorithms performed similarly on these two data sets.

4.3.3 Detection Performance with Neural Data Sets R1-2 and R2-2

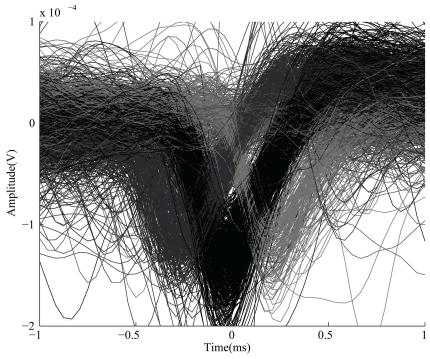
To evaluate the robustness of the detection algorithms, we used R1-2 and R2-2 (50 seconds of neural recordings each) to examine detection performances by MCWC and WDM.

Figure 4.12a illustrates the MCWC detection results using detected spike clusters with alignment for R1-2. As shown in the figure, cluster 2 does not preserve the neural spike characteristics and therefore it is classified into noise. As shown, spike sorting in this case is quite straightforward. Figure 4.12b shows the detection result by WDM without alignment when plotting each and individual spike waveform. Actually, in our experiment, when alignment was performed by using the same correlation approach as did in previous tests for R1-1 and R2-1, the spike waveforms detected by WDM did not appear aligned due to significant noise present in the real neural recordings. In order to provide a fair comparison between WDM and MCWC, we used our subjective judgment to manually classify and align the WDM detection results. By visually inspecting the resulted spikes by WDM algorithm, we noticed time shifted groups of spike waveforms in Figure 4.12b. With the help of K-means, we clustered the detected spike waveforms by WDM using the natural time shift as cluster differentiators. Figure 4.13a, Figure 4.13b, Figure 4.13c and Figure 4.13d are detection results shown by manually aligned spike waveform clusters by WDM. By a visual inspection, the detected waveforms in Figure 4.13a, Figure 4.13c and Figure 4.13d preserve the neural spike characteristics, while waveforms in Figure 4.13b appear to be noise. Com-

pared to Figure 4.12a, it takes more effort to sort spikes in Figure 4.12b to obtain the results shown in Figure 4.13.



(a) MCWC detection results using data set R1-2 with peak alignment (Cluster 1 Rate: 35Hz, Cluster 2 Rate: 7.6Hz. S=5.). Detection results are consistent with that of R1-1.

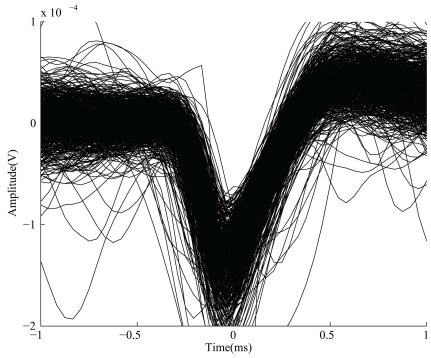


(b) WDM detection results using data set R1-2 without peak alignment, which was attempted but encountered difficulty due to significant noise presence. Manual inspection and classification is thus applied to the detection in this case.

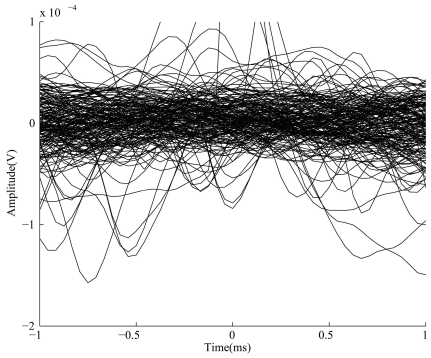
Figure 4.12: Waveforms of detected spikes using data set R1-2. MCWC has resulted in consistent detection outcomes for data sets R1-1 and R1-2.

The MCWC and WDM detection results using real neural recordings of data set R2-2 are shown in Figure 4.14 and Figure 4.15. Unlike data set R1-2, there are 2 clusters of neural spikes in R2-2. The detected waveforms in cluster 2 by MCWC (Figure 4.14c) appear to be noise or these waveforms cannot be reasonably aligned. Like the detection results by WDM using data set R1-2, the WDM detected waveforms from R2-2 still exhibit difficulty as shown in Figure 4.15a when alignment was performed using the principle of maximizing correlation between waveforms. By applying the same manual alignment method as for R1-2, the WDM sorted out 4 separate clusters, one of which is considered noise. This result still is similar to that obtained from data set R1-2.

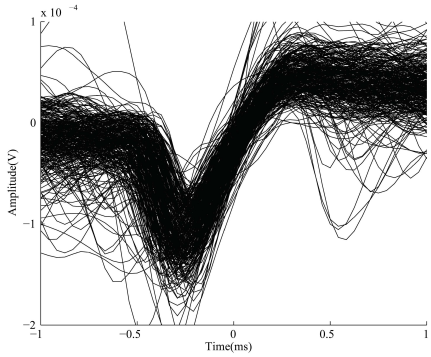
Through the analysis of detection results in this section using sorting and manual inspection, it shows that MCWC has demonstrated a comparable or better detection performance than WDM.



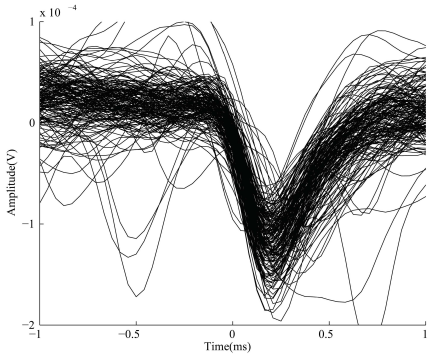
(a) Cluster 1 of WDM detected spikes using R1-2 (Rate: 21.2Hz).



(b) Cluster 2 of WDM detected spikes using R1-2 (Rate: 5.7Hz).



(c) Cluster 3 of WDM detected spikes using R1-2 (Rate: 8.4Hz).



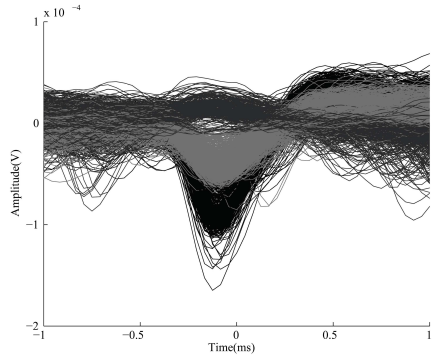
(d) Cluster 4 of WDM detected spikes using R1-2 (Rate: 4.7Hz).

Figure 4.13: Manually peak aligned WDM detected spikes using data set R1-2.

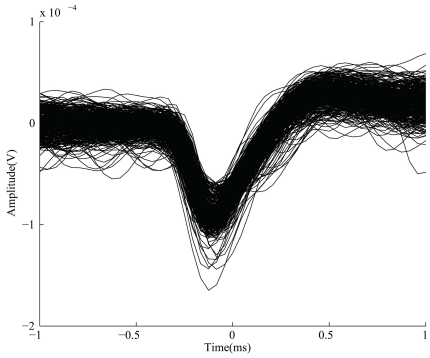
4.4.4 Evaluation using Real Neural Cortical Waveforms with Spike Waveform Verification by Human

In this section we provide detection performance comparison for MCWC and WDM using ROC performance curves. To do so, we manually examined two data sets to provide the ground truth for 1) data set R1-1 that was used in section 4.3 earlier and 2) a 10 minute long neural waveform recorded from rat 2 (R2-L).

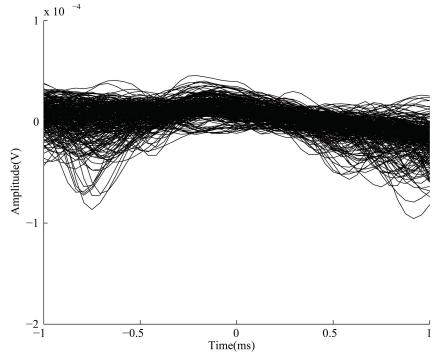
To provide ground truth for R1-1, we manually inspected the neural waveforms and extracted those identified spikes. The firing rate is about 40Hz. Figure 4.16a, Figure 4.16b and Figure 4.16c illustrate 3 sets of hand selected spike examples. Using these manually generated spike truth, we were able to conduct a detection performance



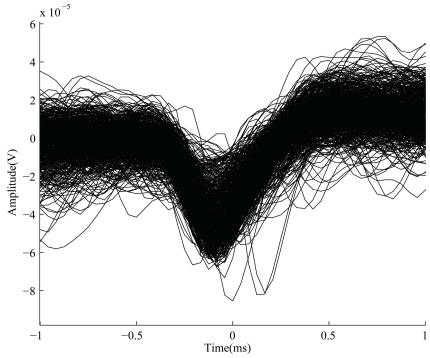
(a) Waveforms of detected spikes with R2-2 using MCWC with peak alignment.



(b) MCWC detected spike waveforms of cluster 1 with R2-2 (Rate: 13Hz).



(c) MCWC detected spike waveforms of cluster 2 with R2-2 (Rate: 5.6Hz).

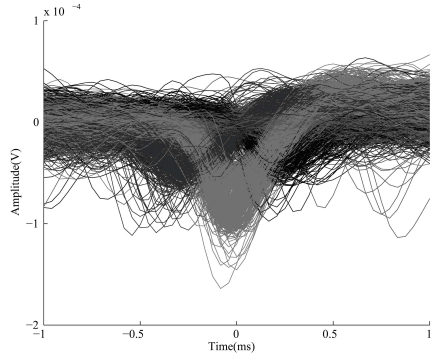


(d) MCWC detected spike waveforms of cluster 3 with R2-2 (Rate: 26Hz).

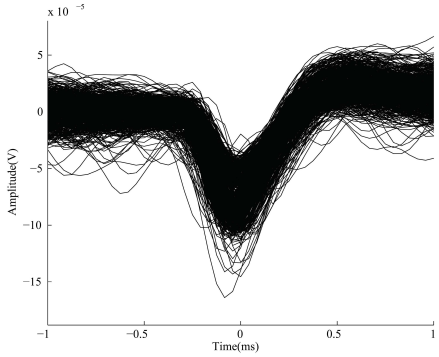
Figure 4.14: MCWC detected spikes using R2-2.

comparison using ROC performance curves, the results of which are shown in Figure 4.16d. As shown in the figure, MCWC generally outperforms WDM in terms of ROC.

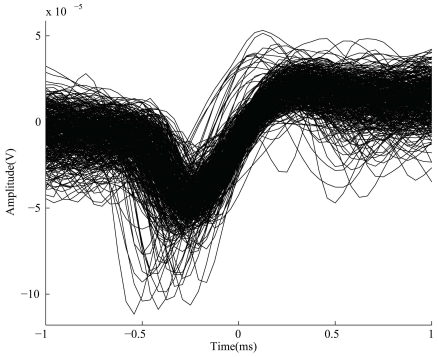
In addition, we performed another test using a 10 minute long neural recording from rat 2 (R2-L). The data set was obtained under the same condition as that used previously, specifically R2-1 and R2-2. Both algorithms were applied to this 10 minute long recording. Detection results are verified manually using the process similar to that used in obtaining Figure 4.16. Varying different detection parameters, we have the results shown in Figure 4.17. As shown in [49], the recommended range for L in WDM is $[-0.2, 0.2]$ with $L = 0$ as the default selection. Also as noted in [49], higher L means lower false alarm rate for WDM, which is verified by Figure 4.17 as well. Figure 4.17



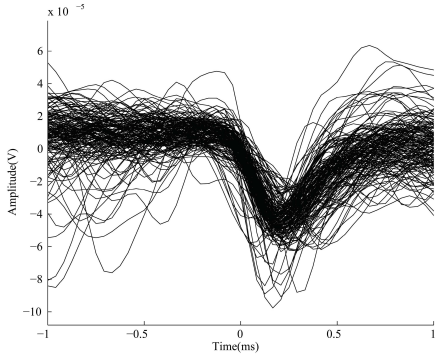
(a) Waveforms of WDM detected spikes using R2-2 without peak alignment.



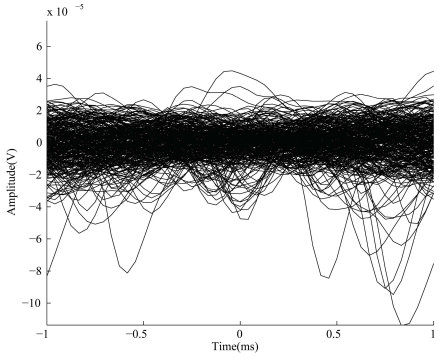
(b) WDM detected spike cluster 1 after manual peak alignment using R2-2 (Rate: 18.2Hz).



(c) WDM detected spike cluster 2 after manual peak alignment using R2-2 (Rate: 11.3Hz).

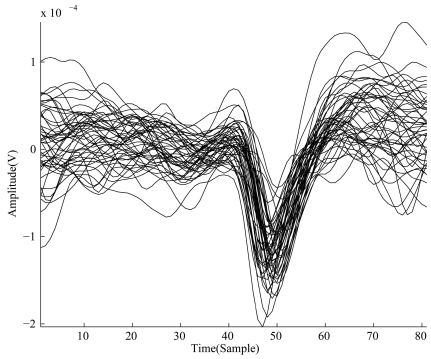


(d) WDM detected spike cluster 3 after manual peak alignment using R2-2 (Rate: 4.8Hz).

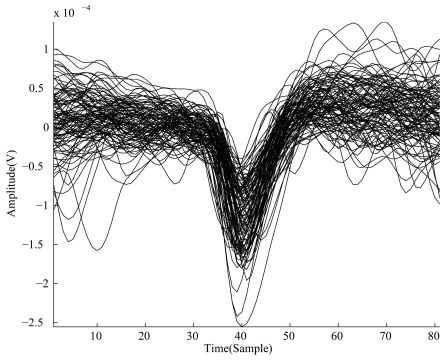


(e) WDM detected spike cluster 4 after manual peak alignment using R2-2 (Rate: 7.8Hz).

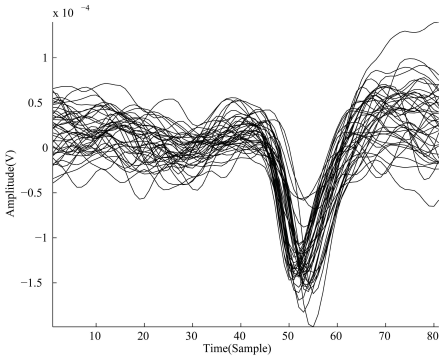
Figure 4.15: WDM detection results using R2-2 with peak manual alignment performed to overcome noise.



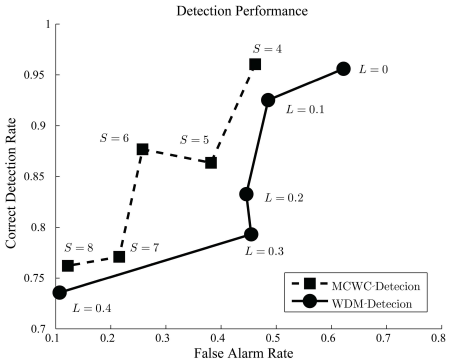
(a) Examples of the 1^{st} set of hand selected spike Waveforms from R1-1 using manual alignment.



(b) Examples of the 2^{nd} set of hand selected spike Waveforms from R1-1 using manual alignment.



(c) Examples of the 3^{rd} set of hand selected spike Waveforms from R1-1 using manual alignment.



(d) Comparison of ROC performance curves for MCWC and WDM based on manually provided ground truth of spikes for data set R1-1.

Figure 4.16: Spike detection for R1-1 with manual verification. In order to show the spike waveform clearly, three clusters were identified and plotted. The ROCs for WDM and MCWC are verified by the same manual spike detection.

also shows that for a given false alarm rate, MCWC's detection rate is generally higher than that of the WDM, and for a given detection rate, MCWC generates less false alarms than WDM.

4.4.5 The Effect of Window Length and Real Time Implementation Issue

The results demonstrated above were obtained by implementing the MCWC algorithm using Visual C++ (Microsoft Corporation) and Intel Integrated Performance Primitives (Intel Corporation). The computer platform is equipped with an Intel Pentium4 2.8GHz Hyper-Thread CPU with 1G memory. To assess real-time implementation potential of

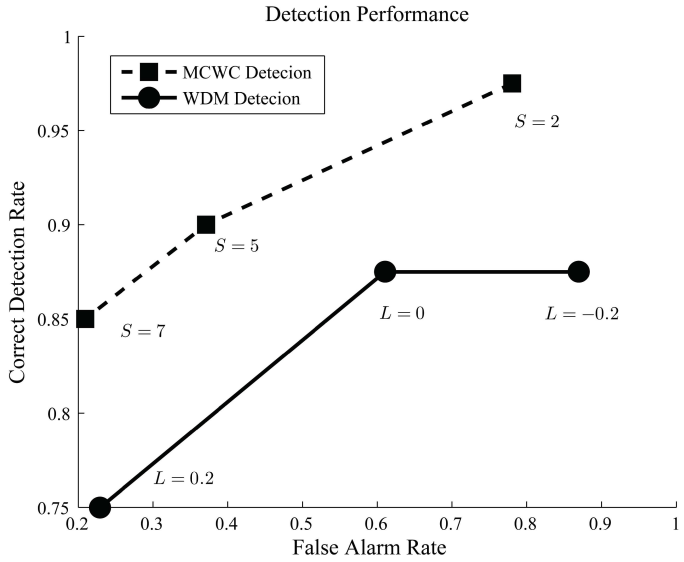


Figure 4.17: Comparison of ROC performance curves for a 10 minute long neural recording from the rat 2 (R2-L) using manually provided ground truth of spikes under different parameter settings for MCWC and WDM.

the proposed MCWC algorithm, we took record of computational time of the MCWC detection algorithm, and the results are summarized and reported here.

The proposed MCWC detection algorithm entails the following steps: wavelet transform, wavelet coefficient correlation (multiplication), and comparison. For wavelet coefficient correlation (multiplication) and comparison, the complexity is $O(n)$, where n is the number of data samples. For wavelet transform, using n samples for both the wavelet function and the neural waveform, the complexity is $O(n^2)$ or $O(nlogn)$ if optimization is performed in implementation. However, due to the compactness of a wavelet function, its effective length is much less than n. Therefore the wavelet transform complexity reduces to approximately $O(n)$. It is also possible to implement the wavelet transform (integration) using hardware. For software implementation in this paper, we take advantage of Intel Single Instruction Multiple Data (SIMD) technology to accelerate the transform.

In our implementation, the Application Programming Interface (API) functions, namely QueryPerformanceFrequency and QueryPerformanceCounter, were used to retrieve the frequency and the value of the high-resolution performance counter of the computer platform respectively (Microsoft Developer Network, Microsoft Corporation) in order to measure the elapsed time for detection. In consideration of real-time application, if the window length is chosen at $J = 100$ms in computing wavelet coefficients, then the waveform data from the window of 100ms will be stored in computer memory before computation. During real-time application, it can process one block of data with the length J at a time.

Table 4.4 shows the computation time of MCWC implemented at different window length J. The multiple-scale parameter S was set to a large value, 10, to make the time estimate more conservative, since from our previous simulation results, $S < 10$ was usually used which requires less computation time than using $S = 10$. With multiple thread programming technique on the multiple-core CPU architecture, we used two program threads in our implementation. One thread is to collect the neural data while the other one to run detection on the collected data. The window length was chosen at $J = 100$ms. While the data collecting thread collects data for current 100ms period, the detection thread runs the detection algorithm on the data recorded in the previous 100ms period. The time needed for running detection on 100ms data is much less than 100ms as shown in the Table 4.4. This implies that within the time frame of the current 100ms data collection, the detection procedure has completed detection on the previous block of the 100ms data. Since the computation time is much below the 100ms time window, any latency and jitter in the processor that may hinder the computation is not of concern.

Additionally, in the current study, we measure the computation time with the notion that the data of length $J = 100$ms have been collected. In our recording, the waveform sampling rate is 24kHz, and each data point is saved with 16 bits. For 100ms

data, the resulted data size is 4.8KB. This is not a problem for any current computer system that is able to collect data at the rate of 48KB/sec. For example, many camera systems running in the Windows platform can show 640x480 (pixel) images at a frame rate of 60 frame/second. If each pixel is 2 Bytes, the data rate is $640 \times 480 \times 2 \times 60$=36MB/sec.

Under these considerations, we refer to MCWC as a "real-time" detection algorithm. As shown throughout this study that, a window length J of 100ms is a reasonable choice.

Table 4.4: Computation time of MCWC software implementation for different window length J

window length J (ms)	50	100	250	500	1000	5000
Computation time (ms)	1.1	1.2	2.3	3.8	6.5	33.5

4.5 Conclusion

This paper introduces a new spike detection algorithm, MCWC, which makes use of correlation and comparison among continuous wavelet transform coefficients at multiple scales. The method avoids using hard thresholding as was used in many popular algorithms which usually results in high false alarm rates. But MCWC requires the selection of parameter S which is the number of sampling scales used in computing the multiple level wavelet coefficients. Similarly the WDM requires parameter L to be selected a priori. Note that the two parameters differ in nature and have different implications in implementations. We provided guiding principles on how to select S and also illustrated using artificial and real data on the effect of different S parameter and its impact on the detection performance. Our simulation results using real data sets show that MCWC is comparable to WDM in terms of identifying waveforms that preserve good physiological features of a neural spike. However, we clearly demonstrated that as a result of the design principle of MCWC, it is more robust than WDM in the sense

that it produces less false alarms as shown using the artificial data set as well as real neural recordings verified by manual inspection. In addition, the detected spike waveforms by MCWC are more consistent than those by WDM for real data set. Finally, we demonstrate the potential of the MCWC algorithm for real-time applications in brain machine interface or general neuronscientific studies.

Chapter 5

ROBUST SPIKE CLASSIFICATION BASED ON FREQUENCY DOMAIN FEATURES AND SELF-ORGANIZED MAPS

5.1 Introduction

Neural spike sorting refers to the process of neural spike detection and classification. Neural spike detection is to extract neural waveforms containing neural spikes from noisy recordings. Neural spike classification is necessary after spike detection since current extracellular multi-channel simultaneous recordings may pick up neural waveforms from more than one neuron on a single channel. Classification is then the process to categorize the detected neural spikes with similar features into one cluster within which each spike is considered to be from an isolated neuron. Spike sorting is thus the very first step in neuroscience for spike train based studies such as spike rate and spike time based neural coding [64], brain-machine-interface applications [37], and neural correlation and synchrony studies [65]. The accuracies of spike sorting and the robustness of spike sorting algorithms have significant impact on the performance of spike train based analyses.

5.1.1 Introduction to Spike Classification

Spike sorting has attracted many research interests over the past two decades. Various algorithms have been proposed [66]. Usually, neural spikes are clustered into disjoint spike groups according to spike waveform features such as waveform shape, ratio between spike peak and duration, and statistical characteristics of the waveform. Each of the clusters is considered a representation of neural spikes from one isolated neuron. Ideas used in spike sorting are diverse. In the following, we review a couple representative approaches to spike classification.

Pouzat et al.[67] proposed a spike sorting method based on the analysis of noise signature. In the development of the classification algorithm, noise is considered a multivariate Gaussian process. This method first extracted noise by subtracting the detected

spikes from the recorded neural waveforms. Then, analyzing the noise profile by computing its second-order statistics, the authors modeled neural spikes by a general data (spike) generation model. Each unit of the generation model represents one neuron. The number of unit (neuron) was estimated by maximizing Bayesian information criterion (BIC). Given the number of units (neurons), unit waveforms and firing rate, the data generation model was estimated by expectation-maximization of the posteriori probability of observed neural recordings. Once the general spike model is established, spikes are classified by minimizing the square difference between the spike and the model output.

The authors in [68] proposed a real-time spike classification method using templates generated from a "learning set" of neural spike waveforms. The candidate spikes were detected by manual thresholding method. The candidate spike waveforms of the learning set were selected by maximizing their discriminations among all unclassified waveforms in terms of variance and distance. Then K-means with different numbers of clusters was applied to all potential spike waveforms. The final chosen number of clusters was determined by finding two consecutive cluster numbers, K and $K+1$, such that the ratio in partition errors between the K^{th} and the $(K+1)^{th}$ clusters was less than 10. Once the number of clusters was estimated, the mean of each cluster was used as the spike template. In real-time spike sorting application, the Euclidean distance between the detected waveform and template was used for as a sorting criterion.

Independent component analysis (ICA) has been applied to spike sorting in [69]. A limitation of the ICA based spike classification method is that the number of recording channels needs to be greater than the number of neurons, which is not quite the case in extracellular cortical recording practices. Takahashi et al. then proposed a new method integrating ICA with K-means to overcome such limitation [70] and [71]. Before using ICA, K-Means was employed to sort out the detected spikes into many clusters. Usually, the cluster number is twice as many as the possible number

of neurons. This fix seems to be able to address the first limitation. But however, another limitation is that the algorithm requires statistical independence among the source (unmixed) signals. That implies that the single unit activities are independent. This is a rather un-realistic condition as shown in [72] where it shows that neuronal correlations clearly exist.

One common concern for spike sorting algorithms is that they usually require manual manipulation or parameter tuning. For multiple electrode arrays, such manual manipulation has to be applied to each channel [73]. The parameters set by different users may result in significant variations in the final clustering result [37].

To address these concerns, Letelier and Weber [74] classified neural spikes making use of discrete wavelet (Daubechies-8) coefficients. First, the mean and the standard deviation (SD) of the wavelet coefficients at each wavelet scale and time translation of all detected spikes were calculated. The wavelet coefficient with the largest mean and SD at some specific scales and time translations were visually selected to discriminate different spikes. Second, for all selected coefficients, all possible combinations of 2-D projections of the coefficients were plotted. With visual inspection, the projections showing clear clustering boundaries were used to classify the spikes. However, the neural spikes classified with respect to wavelet time translations may correspond to the same spike cluster but with a time delay between them. Therefore this spike classification requires an accurate neural waveform alignment procedure so that the spikes are aligned with respect to a common reference. Also the number of all possible combinations of 2-D projections of coefficients may be too high to make the algorithm practical since it involves manually visual inspections by the user.

Hulata et al. [44] and [75] proposed a spike detection and sorting algorithm based on a wavelet package. The optimal basis which concentrates most energy of the neural signals was selected by maximizing the information entropy. The spikes were sorted by local discriminant basis based on mutual entropy. As indicated in [76],

such approach is a supervised classification. Also the author in [77] showed that the estimation of mutual information used in spike sorting was not an easy task.

Various neural network methods have been applied to spike sorting. For example, Vogelstein et al. [78] uses support vector machines (SVM) for spike detection and classification. However, it is necessary to train the SVM with manually detected and classified spikes in advance. The accuracy of manual detected and classified spikes may play an important role in the sorting performance based on SVM. The author in [79] compared three classification methods: template matching, principle component analysis (PCA) and artificial neural networks (ANN). The conclusion was that ANN was the best classifier of all three methods though the ANN performance relied on a large training data set.

Frequency domain features of neural spike waveforms along with PCA method has been reported in [80] for spike sorting. The amplitude component (not phase component) of detected spikes were analyzed by PCA and sorted by Fuzzy C-Means (FCM). Evaluation with the artificial spikes showed that the proposed method improves classification performance if the detected spikes were mis-aligned. However the periodic nature of the discrete Fourier transform with a period π in the magnitude and phase component had complicated their classification procedure. i.e., the spike waveforms with phase difference at π share the same frequency magnitude components but with different waveforms in time domain.

By analyzing the physical features of a neural spike waveform, the authors in [81] discovered that the differences in spike waveforms resulted in high frequency components while low frequency components were results of noise. Based on the assumptions in the spike geometry model [82], the author used the first derivatives, acting as a frequency shaping filter, as the feature to enhance spike sorting. Spike clustering was implemented using evolving mean shift clustering algorithm. Sometimes post-merging process was necessary since evolving mean shift clustering might result in too many

clusters [83]. The authors demonstrated their spike sorting results by using an artificial data set.

WaveClus is a software package proposed by Quiroga et al. [76]. It is an unsupervised spike detection and classification method without any explicit assumption on the spike distribution. It describes spike features by wavelet transform coefficients and classifies these features by super paramagnetic clustering (SPC) [84]. One of the free parameters in the algorithm was the annealing temperature, which was used to determine the number of clusters. The SPC clustering resembled a spin glass model in physics, which changed its state in different ways at different temperatures. At low temperature, all spins changed their states simultaneously and are classified into one cluster. On the contrary, at high temperature all spins change states indecently and are classified into different clusters. At the proper medium temperature, only some of the spins change their states simultaneously and are classified into one cluster. WaveClus automatically selects the proper temperature in the SPC classification algorithm [76].

The Offline Sorter (Plexon Inc., Dallas, TX) is a popular commercial spike sorting package that has been widely used in the neuroscience community. It detects neural spikes with a manual threshold. After that, the user can choose one of the three types of spike classification algorithms: manual, semi-automatic and automatic methods [85]. Since manual and semi-automatic methods rely on a user's experience for parameter selection, the results are therefore subjective. But Offline Sorter also offers other automatic sorting algorithms including valley seeking and T-Distribution EM. Results in [86] demonstrated that the T-distribution EM outperformed the valley seeking.

It is understandable that a good spike sorting algorithm should be accurate with low false identification, robust to signal-to-noise ratio, and as less dependent on user subjective choice as possible. With the advances in brain machine interface (BMI), real time spike sorting also becomes a desired feature. Given the limitations of most of the

spike sorting algorithms, manual spike sorting still is very common [66]. This paper is among the many that attempts to develop a useful automatic spike sorting algorithm.

5.2 Classification Based on Frequency Domain Features (CFDF)

Figure 5.1 is a schematic of the proposed spike detection and classification algorithm. The system includes two important components: spike detection and spike classification. Once spikes are detected, a Fourier transform is performed on the spike waveforms, the frequency domain features of which are used for classification. First, the magnitude components are analyzed and placed into clusters. Second, the clusters obtained from the first step will be examined further by making use of the phase components.

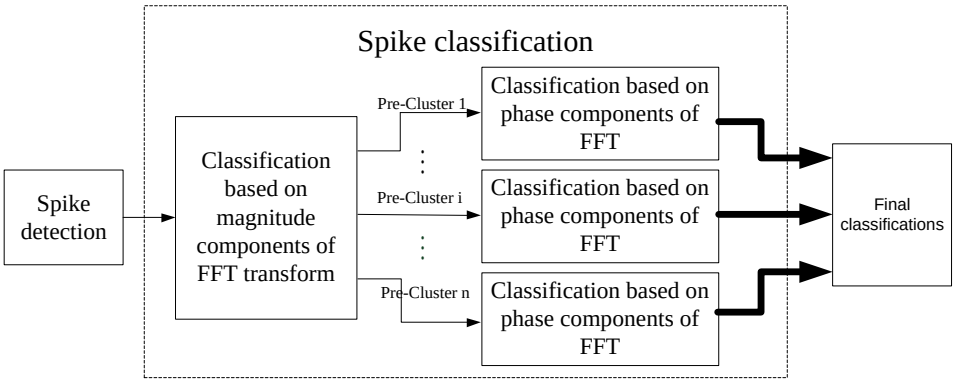


Figure 5.1: Schematic of the proposed spike detection and classification system named CFDF. Classification uses frequency domain features: it first clusters spikes based on the magnitude component. Then each cluster will be further classified based on the phase component to form the final spike classification.

5.2.1 Spike Classification Based on Frequency Domain Features

Once neural spikes are detected using MCWC, the segment of waveform containing a spike is extracted for further analysis. Aligning detected spikes according to a common reference is an important step before spike classification especially when the shape of a spike waveform is used as one of the spike sorting features. For example, a spike peak or valley or zero crossing of the waveform are usually used as alignment references. Various noise during recording, however, can interfere and result in mis-alignment,

which consequently degrades spike sorting performance. Our method examines the mis-alignment by employing phase component of frequency domain features of the spike waveforms. How the method works is introduced below.

Frequency Domain Features

A mis-aligned spike waveform introduces a time delay in the detected spike in relation to others. Notice that a time delay results in a phase shift in the frequency domain, but does not alter the frequency domain magnitude components of the spike waveform. Taking advantage of this property, the proposed spike classification includes two steps. First, spikes are classified according to frequency domain magnitude components. The magnitude components of a waveform correspond with spike waveform features such as spike duration and spike amplitude in the time domain. Therefore, this step results in clusters, each of which shares similar waveform features except spike mis-alignment in the time domain and possibly other subtle waveform differences. The second step makes use of the phase components in the frequency domain of the spike waveforms in each cluster for further spike classification. Mis-alignment of spike waveforms will be reflected in the phase components. As shown next, subtle waveform differences also result in phase deviations across all frequencies. This properties is helpful to classify the waveforms with subtle differences.

Enhancing Spike Features by FFT

Let $x(n)$ $(n \in [0, N-1])$ be a spike waveform, and $X(k)$ is its fast Fourier transform (FFT). Then,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{i2\pi kn}{N}} \tag{5.1}$$

As an example, let $x'(n)$ represent a spike waveform deviating from $x(n)$ by $\Delta(m)$ only at $n = m$ as defined in(5.2).

$$x'(n) = \begin{cases} x(n) & n \neq m \\ x(m) + \Delta(m) & n = m \end{cases} \tag{5.2}$$

Let $X'(k)$ be the FFT of $x'(n)$, we will observe the following relationship between $X(k)$ and $X'(k)$.

$$\begin{aligned} X'(k) &= \sum_{n=0}^{N-1} x'(n) e^{-\frac{i2\pi kn}{N}} \\ &= \sum_{n=0}^{m-1} x(n) e^{-\frac{i2\pi kn}{N}} + [x(m) + \Delta(m)] e^{-\frac{i2\pi km}{N}} + \sum_{n=m+1}^{N-1} x(n) e^{-\frac{i2\pi kn}{N}} \\ &= X(k) + \Delta(m)^{-\frac{i2\pi km}{N}}. \end{aligned} \tag{5.3}$$

According to (5.3), for each digital frequency k, there are always phase changes defined by $\Delta(m)^{-\frac{i2\pi km}{N}}$ which results from waveform change at $n = m$. The term $\Delta(m)$ acts as a short impulse in the time domain. It affects $X(k)$ in all k's. Taking this into account during implementation of the phase values, unwrapped phase components are considered so that the estimated phase components are continuous functions of time. Phase unwrapping is helpful to accumulate any phase changes due to continuity and smoothness constraints. Such property is helpful to capture subtle changes represented by frequency phase components.

After these two steps, waveforms in each cluster share similar waveform features with little or no time delay and little subtle waveform differences. A clustering algorithm such as the Self-Organized Map (SOM) can then be used for classification of waveforms into isolated neurons. Note that, if a detection algorithm has high detection accuracy and low false alarm rate, then any other clustering algorithms will result in very similar classification of neurons. Throughout this paper, we refer the proposed method as classification with frequency domain features (CFDF).

5.3 Sorting Performance Evaluation for Spike Classification using CFDF

In this section, we will be using two artificial data sets to bench mark the proposed spike classification algorithm CFDF. And then we will illustrate how the CFDF procedure can be used to classify real neural spike waveforms effectively.

5.3.1 Artificial Data I

Each artificial neural data set was generated the same way as in [87]. The simulated neural data sets were obtained by activating three types of neurons in the program: target neuron (used as truth), correlated neuron (its waveforms are regarded as correlated interference), and uncorrelated neuron (its waveforms are regarded as uncorrelated interference). The spike generating program was used to simulate neural waveforms of a single channel from multiple channel recording arrays. In the simulation program, the refractory period was set to 1ms.

In this study, two different artificial spikes were generated. Figure 5.2 shows their averaged waveforms based on the ground truth of spike waveforms.

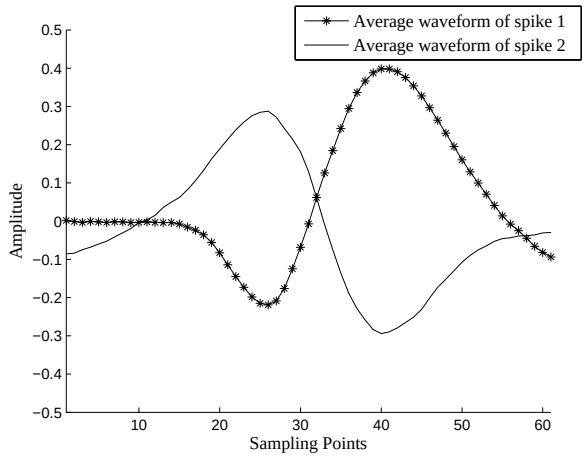


Figure 5.2: The average waveforms of artificial spikes generated according to the ground truth. There are two different spikes used for evaluating spike classification algorithms.

When Plexon Offline Sorter was applied, spike detection was done by thresholding on the "Energy of Signal". Spikes were classified by automatic T-Distribution E-M algorithm.

For thresholding detection, lower threshold guarantees most of spikes are detected while noise may also be detected as false positives. Usually, spike detection is supposed to sort out all these noise in order to reduce the false alarm, i.e., noise is classified into one "noise" cluster. For the artificial data set I, there are 112 spikes fired by two different neurons. The threshold was selected so that 206 potential spikes were detected. Offline Sorter classified three clusters by T-Distribution E-M algorithm.

Table 5.1 shows Offline Sorter performance for artificial data I. With the true spike timing provided by the artificial data I, the corresponding Offline Sorter correct detection rate is 53.57% and false alarm rate is 70.87%, respectively. The false alarms is not classified into one cluster as expected. Instead they distribute in each cluster.

Table 5.1: Offline Sorter performance for artificial data I.

Cluster No.	Number of correct detection	Number of false alarm
Cluster 1	7	10
Cluster 2	35	83
Cluster 3	18	53
Total	60	146

WaveClus Sorting Performance

WaveClus also uses thresholding for spike detection. Unlike Offline Sorter, WaveClus uses both low and high thresholds. Any waveforms within these thresholds are detected as spikes. The thresholds are defined in (5.4) and (5.5). The constant k in (5.5) defines the threshold. The pair, $(k_1, k_2), k_1 < k_2$, defines low and high thresholds. For example, $k_1 = 4$ and $k_2 = 20$ defines low and high thresholds as $4\sigma_n$ and $20\sigma_n$, respectively.

$$\sigma_n = median\left\{\frac{|x|}{0.6745}\right\}. \tag{5.4}$$

$$Thr = k\sigma_n. \tag{5.5}$$

where *median* is the median filter, Thr is the detection threshold.

Table 5.2 shows WaveClus detection performance in terms of correct detection and false positive rate by varying both low and high thresholds. Comparing the detection performance by MCWC later, WaveClus has a lower detection performance than MCWC. In order to compare different spike classifications fairly, we use the same detection result by MCWC as the input to both WaveClus and CFDF classifications.

Table 5.2: WaveClus detection performance with artificial data set I.

Low threshold (k_1)	High threshold (k_2)	Correct detection/False positive
$k_1 = 2$	$k_2 = 25$	49.11%/68.39%
$k_1 = 2$	$k_2 = 20$	49.11%/68.39%
$k_1 = 3$	$k_2 = 20$	17.86%/57.45%
$k_1 = 1$	$k_2 = 20$	76.79%/82.41%
$k_1 = 1$	$k_2 = 10$	76.79%/82.41%
$k_1 = 1$	$k_2 = 3$	74.11%/78.61%

Figure 5.3 shows the automatic classification result by WaveClus using MCWC detection as the input. The automatic classification only shows one cluster while the ground truth actually has two clusters. Manually varying the classification parameter, WaveClus is able to classify two clusters as shown in Figure 5.4.

CFDF Classification Performance

For the artificial data I, the correct detection rate and false alarm rate for MCWC were 95.54% and 10.83%, respectively. On the contrary, WaveClus only detects 8 spikes with the defalut settings for $k_1 = 4, k_2 = 25$.

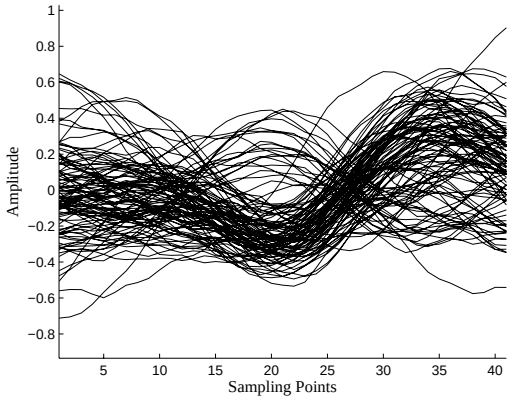
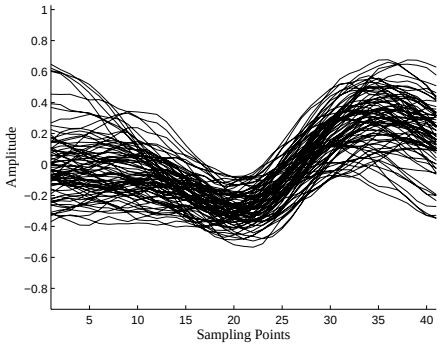
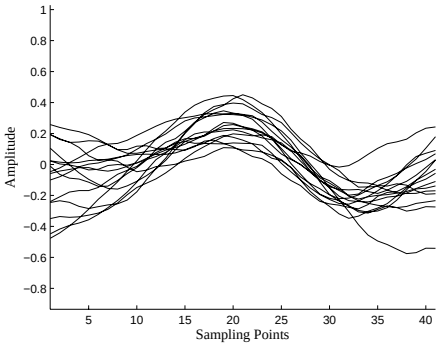


Figure 5.3: The unsupervised clustering results by WaveClus. Only one cluster was classified while the truth was two neuron clusters.



(a) The first cluster identified by WaveClus with manually selected parameters.



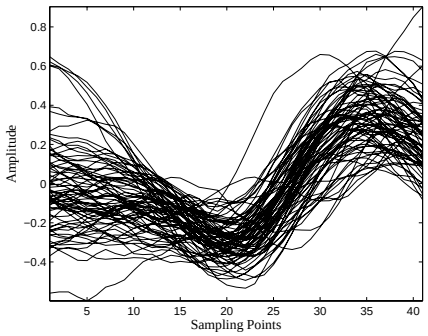
(b) The second cluster identified by WaveClus with manually selected parameters.

Figure 5.4: WaveClus classification parameters were manually tuned to obtain a result of two clusters.

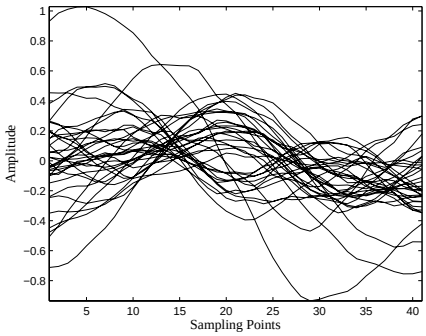
Figure 5.5 shows CFDF classification results. Clustering based on magnitude components in Figure 5.5a shows only one cluster. Classification based on phase features was applied to the clustering results by magnitude components, which resulted in two clusters as shown in Figure 5.5b shows two clusters. Figure 5.5c and Figure 5.5d illustrated final classification results that are consisitent with the ground turth.

(a) SOM clustering result based on the magnitude componets of FFT. Only one cluster was identified.

(b) Further classification based on phase features. This result was obtained by applying the SOM phase classification to the clustering result in (a).



(c) The first cluster of artificial data set I by CFDF.



(d) The second cluster of artificial data set I by CFDF.

Figure 5.5: The propose spike sorting structure in Figure 5.1 applied to artificial data set I.

5.3.2 *Artificial Data II*

The artificial data set II was provided in the WaveClus package [76]. We chose the three most difficult subsets from the package for sorting performance comparison. The truth to the data set includes three clusters in each of the three artificial data sets.

Offline Sorter Sorting Performance

As indicated in Offline Sorter manual [85], T-Distribution E-M algorithm has a setting for its parameter, Degree Of Freedom (DOF). The theoretically calculated DOF usually results in a poor classification. It is necessary to adjust DOF manually. Offline Sorter introduces a parameter, D.O.F Mult., used to adjust the theoretical DOF by multiplying theoretically calculated DOF with D.O.F Mult.. T-Distribution E-M algorithm usually classifies larger number of cluster with smaller DOF and less number of clusters with larger DOF. Plexon Offline Sorter also provides several statistical measurements used to evaluate spike sorting quality. The sorting quality is helpful to select the best DOF setting. Five types of measurements were used: J3, Pseudo-F, Davies-Bouldin validity metric, Dunn cluster validity metrics and p-value.

The selection criteria of the proper D.O.F Mult. is based on the following considerations: 1) The D.O.F Mult. is selected in a way such that at least three clusters are classified, since the ground truth is three. 2) if multiple D.O.F Mult.s result in at least three clusters, we will select the cluster number and D.O.F Mult. in terms of statistical measurements. In this paper, we used two different ranges and range resolutions for D.O.F Mult.. One range was $[10, 100]$ with a resolution of 10. The second range was $[1, 10]$ with a resolution of 1. The reason we used the second range was that the cluster number by T-Distribution E-M algorithm usually varies less in the range $[10, 100]$ than the range $[1, 10]$. So we used a finer resolution for the range $[1, 10]$.

Table 5.3: Plexon Offline sorting results
for artificial data set II-1.

D.O.F Mult.	Numbrt of Cluster	D.O.F Mult.	Numbrt of Cluster
1	8	2	6
3	4	4	4
5	4	6	3
7	3	8	3
9	2		
10	2	20	2
30	2	40	2
50	2	60	2
70	2	80	2
90	1	100	1

Table 5.4: Plexon Offline Sorter performance evaluation
for artificial data set II-1.

Best classification measured by	D.O.F Mult.	Numbrt of Cluster
J3	2	6
Pseudo-F	6	3
Davies-Bouldin validity metric	8	3
Dunn cluster validity	6	3
p-value	1	8

Table 5.3 shows how the number of clusters varies with D.O.F Mult.s. The data set used in this table was artificial data set II-1 (noise level 0.1). When the range of D.O.F Mult.s was $[10, 100]$, the number of clusters was two or one. However, the number of clusters was no less than three for the range $[1, 10]$. Table 5.4 shows the best classification and the corresponding cluster number in terms of statistical measurement. Since there is no universally acceptable sorting evaluation method, we selected the number of clusters which appeared more times than others. According to this criterion, the best cluster number in Table 5.4 is 3.

Similarly, we analyzed artificial data set II-2 (noise level 0.15) and II-3 (noise level 0.2) in the same way as for artificial data set II-1 (noise level 0.15). Table 5.5,

Table 5.5: Plexon Offline Sorter sorting results
for artificial data set II-2.

D.O.F Mult.	Numbrt of Cluster	D.O.F Mult.	Numbrt of Cluster
1	7	2	5
3	3	4	3
5	3	6	3
7	3	8	3
9	3		
10	3	20	3
30	2	40	2
50	2	60	2
70	2	80	2
90	3	100	3

Table 5.6: Plexon Offline Sorter performance evaluation
for artificial data set II-2.

Best classification measured by	D.O.F Mult.	Numbrt of Cluster
J3	1	7
Pseudo-F	3	3
Davies-Bouldin validity metric	3	3
Dunn cluster validity	3	3
p-value	1	7

Table 5.6, Table 5.7 and Table 5.8 show the corresponding Offline Sorter classification results. The best number of clusters was selected as 3 for both data sets. To compare sorting performance, we need to provide a fair comparing ground. In this case, we let all algorithms result in 3 clusters since that is the number in the truth. For example, for data set II-2, referring to Table 5.6 and Table 5.8, the Offline Sorter indicates that 3 is the most likely number of neuron clusters. We therefore compared the correct classification rates in each cluster with the truth in each cluster to come up with the final performance comparison Table 5.9.

WaveClus and CFDF Sorting Performance

Table 5.7: Plexon Offline Sorter sorting results
for artificial data set II-3.

D.O.F Mult.	Numbrt of Cluster	D.O.F Mult.	Numbrt of Cluster
1	7	2	6
3	3	4	3
5	3	6	3
7	3	8	3
9	3		
10	3	20	3
30	2	40	2
50	2	60	2
70	2	80	2
90	2	100	2

Table 5.8: Plexon Offline Sorter performance evaluation
for artificial data set II-3.

Best classification measured by	D.O.F Mult.	Numbrt of Cluster
J3	1	7
Pseudo-F	3	3
Davies-Bouldin validity metric	8	3
Dunn cluster validity	8	3
p-value	1	7

The WaveClus package provides some artificial data sets for evaluation purpose. In order to compare different algorithms fairly, WaveClus and the proposed CFDF method used the same detected spikes which was provided with WaveClus package. Figure 5.6 and Figure 5.7 show that the SOM classification results for the artificial data set II. The proposed CFDF method classified the spikes into four clusters. In Table 5.9, the correct classification rates for the artificial data set II are listed referenced by ground truth. The comparison shows that the proposed CFDF method has a better classification performance than WaveClus. For the third data set, the classification presented in [81] was 92%, higher than the proposed CFDF method by 9%. However, the algorithm in [81] sometimes needs "post-merging" since such clustering might result in too many

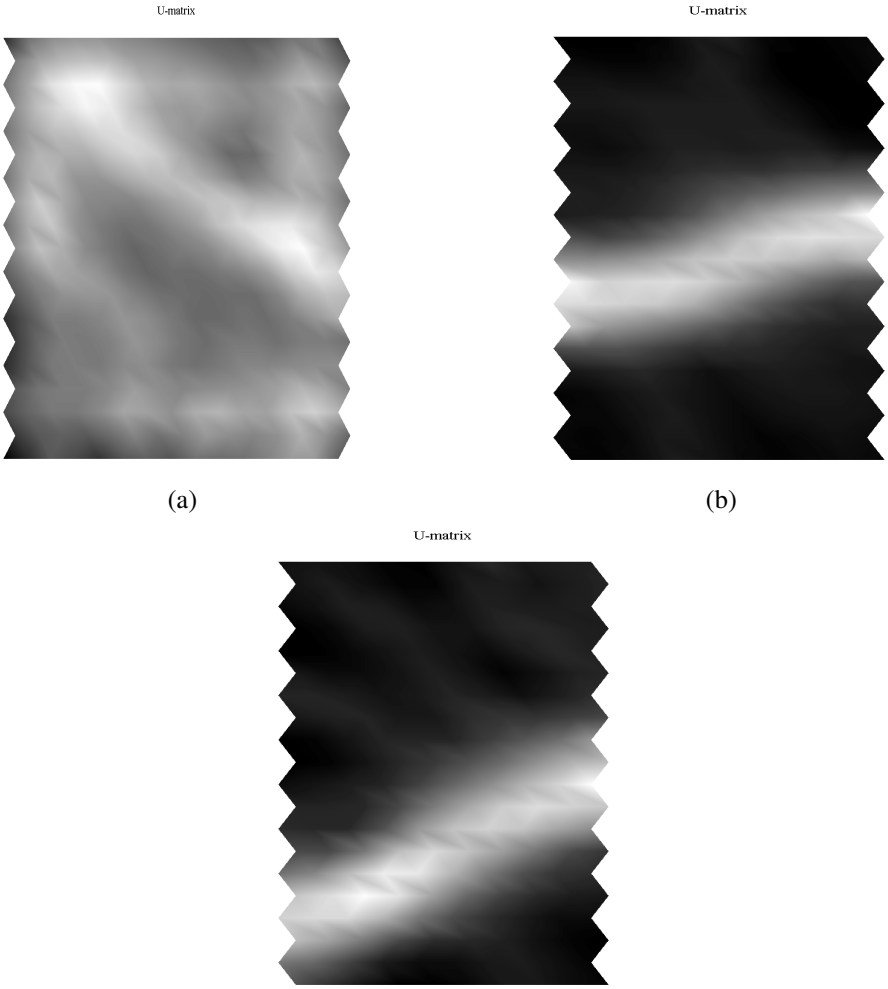
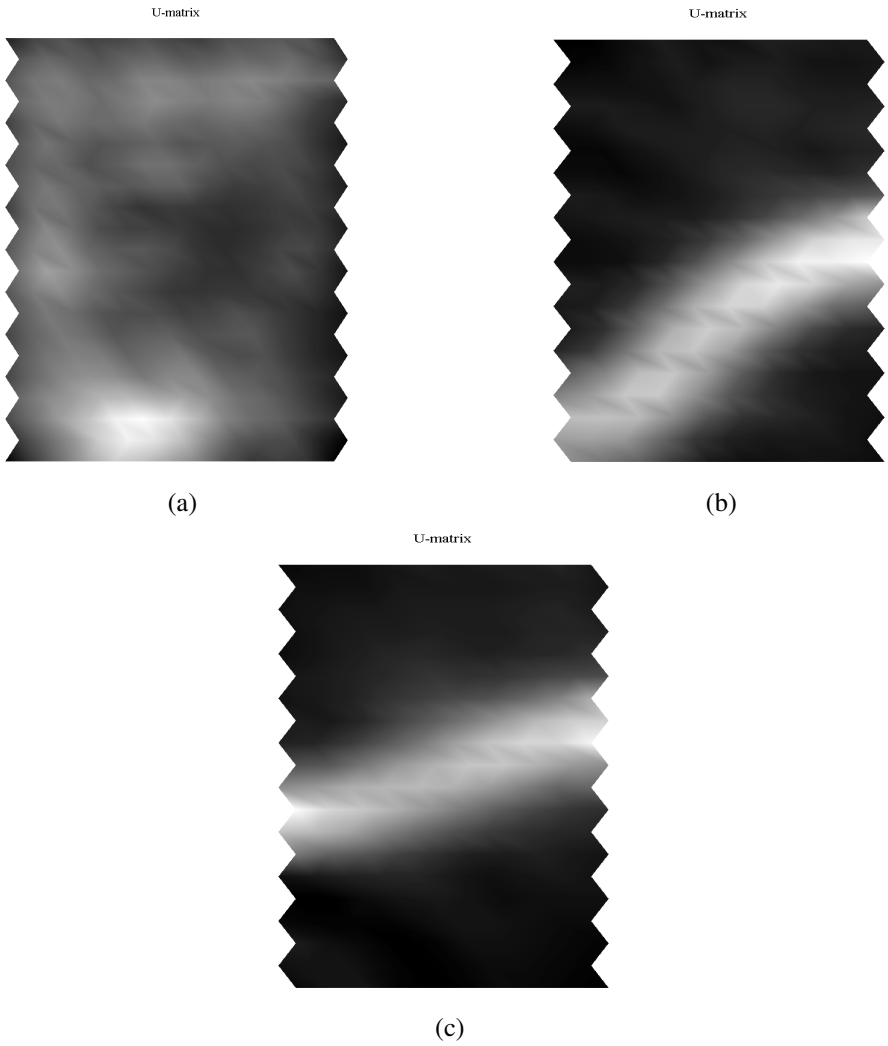


Figure 5.6: The SOM classification of artificial data set II-1. (a)The SOM classification of frequency amplitudes. Visual classification shows two clusters. Each of them will be further classified with phase information respectively as shown in (b) and (c). (b) The SOM classification of frequency phase of one cluster, which shows two clusters. (c) The SOM classification of frequency phase of the second cluster, which shows two clusters.

clusters [83]. Since no details were available on how to select the classificaton parameters and merge different clusters in [81], we just focused on comparing our proposed CFDF method with WaveClus.

5.3.3 Real Neural Recording

We used real microarray recordings from rat's motor cortical digitized at 24kHz using TDT RX5 Pentusa Base Station (TDT, Inc). Four sets of neural recordings (RM-1,



(a) (b)

(c)

Figure 5.7: Classification results by CFDF for artificial data set II-3.(a)The SOM classification of frequency amplitudes. Visual classification shows two clusters. Each of them will be further classified with phase information respectively as shown in (b) and (c). (b) The SOM classification of frequency phase of one cluster, which shows two clusters. (c) The SOM classification of frequency phase of the second cluster, which shows two clusters.

Table 5.9: Correct classification rate for the artificial data set II.

Data set	Offline sorter	WaveClus	CFDF
II-1 (noise level 0.1)	$\leq$68.6%	99.71%	99.65%
II-2 (noise level 0.15)	$\leq$59.47%	83.12%	95%
II-3 (noise level 0.2)	$\leq$50.0%	46.17%	83%

RM-2, RS-1, and RS-2) from 2 rats were examined. Neural waveforms were recorded while the rats freely moved about inside a Skinner box.

Figure 5.8 shows the detected waveforms of RS-1.

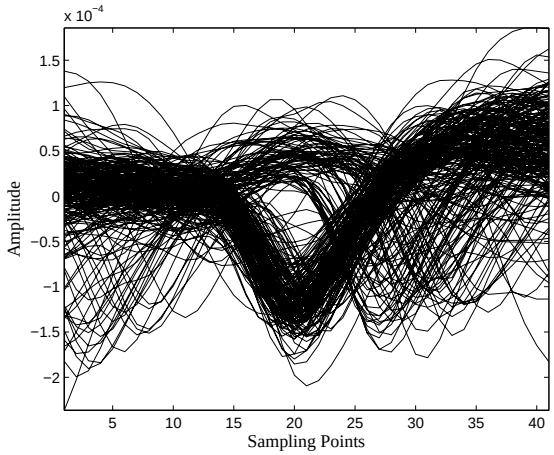
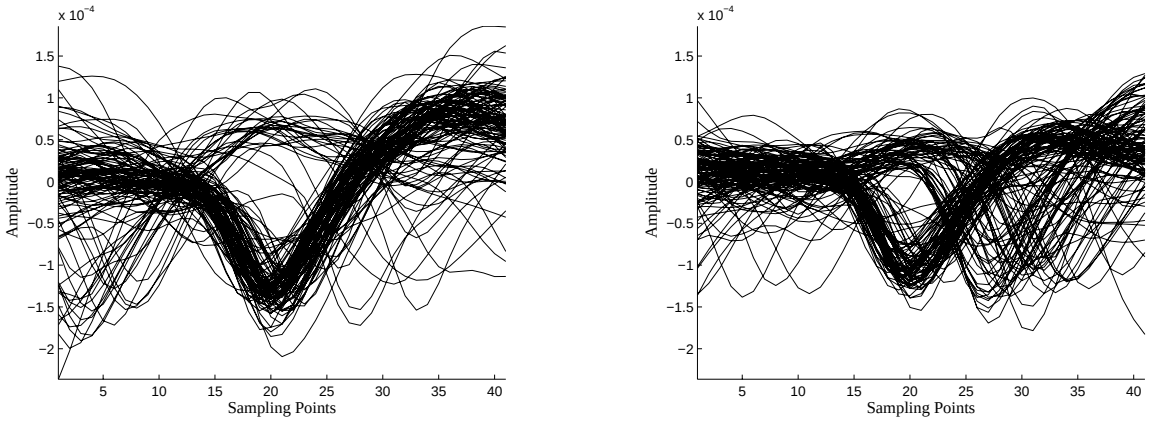


Figure 5.8: Detected spike waveforms from RS-1.

Figure 5.9 shows the waveforms of two clusters classified by the amplitude feature of RS-1.

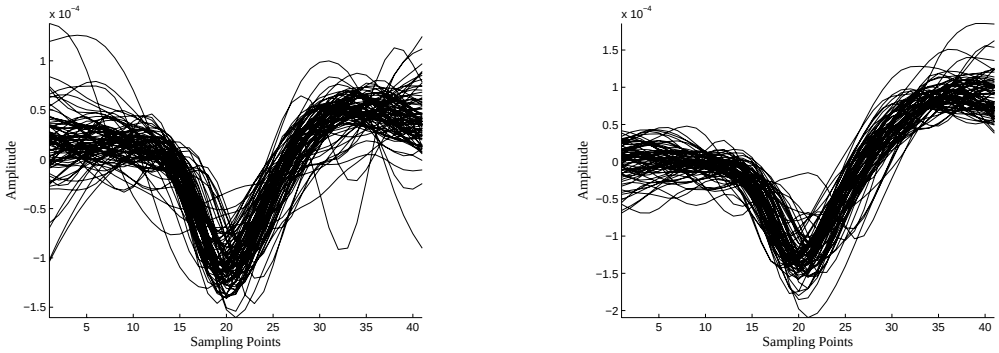
After the first step of clustering based on the amplitude components, we proceeded to the second step: classifying each cluster from the first step using phase features. Finally, we classified the RS-1 into four clusters as shown in Figure 5.10. We applied the same classification procedure to RM-1 and find similar outcome.

In order to compare the method CFDF proposed in this paper with other methods fairly, we used the same detected waveforms as used in CFDF for WaveClus classi-

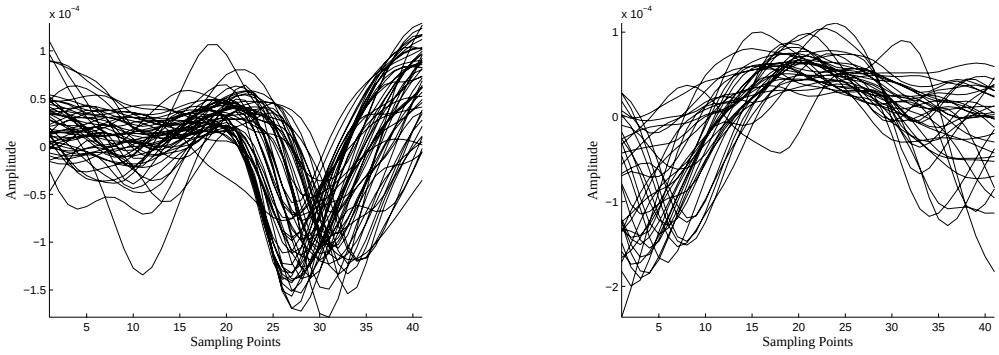


(a) (b)

Figure 5.9: (a)The first cluster identified by the first SOM using magnitude components for data set RS-1. (b) The second cluster identified by the same SOM in (a).

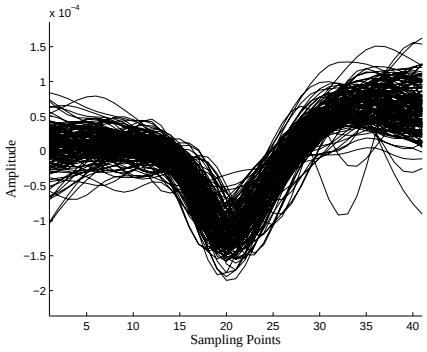


(a) The first cluster of data RS-1 (b) The second cluster of data RS-1

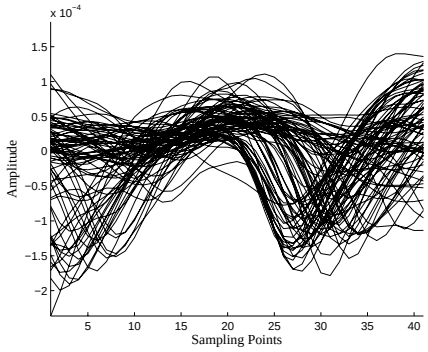


(c) The third cluster of data RS-1 (d) The fourth cluster of data RS-1

Figure 5.10: The four clusters identified by frequency domain features using the proposed algorithm in Figure 5.1. For the first cluster in (a), the waveform amplitudes around sample 0 to 15 are pretty close to those around over-shoot area (from sample 30 to sample 40). For the second cluster in (b), the waveform amplitudes around the on-set (from sample 0 to sample 15) are lower than those around over-shoot area (from sample 30 to sample 40).



(a) The first cluster identified by WaveClus for data RS-1



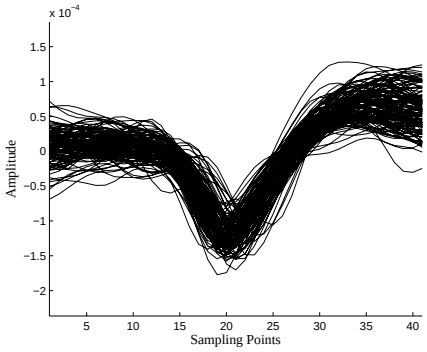
(b) The second cluster identified by WaveClus for data RS-1

Figure 5.11: The WaveClus parameter setting scenario 1 : the unsupervised clustering result (Temperature is selected automatically). Only two clusters are classified.

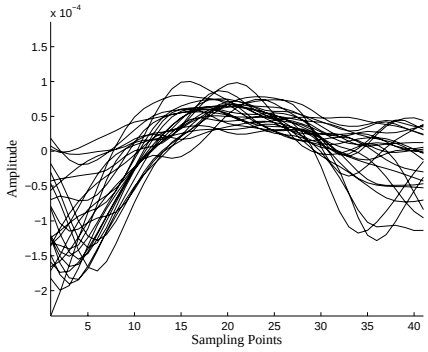
fication. Figure 5.11 and Figure 5.12 show that WaveClus classification results of RS-1 varied with the parameters, which require careful selection. None of WaveClus classification results shows clearly as what the CFDF method did. It is noted that Waveclus can't classify the waveforms in Figure 5.11a in the same way as CFDF did as shown in Figure 5.10a and Figure 5.10b. Similar observations apply to the data RM-1 for CFDF and WaveClus.

5.4 Conclusion

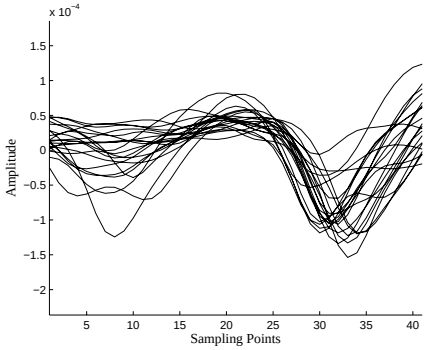
This paper proposes a spike classification algorithm, namely the CFDF. The algorithm relies on both magnitude and phase features of the potential spike waveforms in the frequency domain. Classification based on magnitude features alone results in a rough distributions of clusters. Further classification based on phase features can capture fine and subtle differences within each cluster generated from the previous step. The performance comparison shows that the CFDF algorithm is more robust than the sophisticated Super Paramagnetic Clustering (SPC), and it is comparable to popular spike sorter such as Offline Sorter. Even though it requires supervision when deciding on the final cluster numbers, using the SOM clustering algorithm makes the process straightforward since the U-matrix intuitively and clearly shows different clusters. Existing automatic



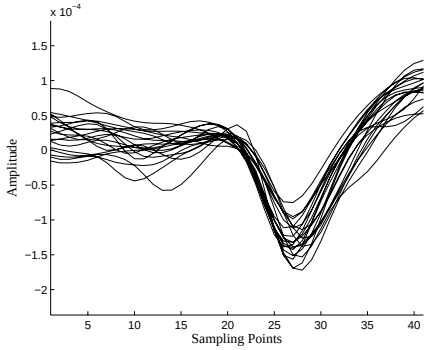
(a) The first cluster of data RS-1



(b) The second cluster of data RS-1



(c) The third cluster of data RS-1



(d) The fourth cluster of data RS-1

Figure 5.12: The WaveClus parameter setting scenario 2 : Manual classifying four clusters of RS-1.

methods can also be applied to determine the number of clusters [88] in this this regard.

Therefore the proposed method is suitable for neuroscience applications.

Chapter 6

SPIKE-TIMING-DEPENDENT PLASTICITY *IN VIVO*: MODIFICATION OF FUNCTIONAL EFFICACY IN RAT'S MOTOR CORTICAL AREAS INDUCED BY COGNITIVE LEARNING

6.1 Introduction

Spike-timing-dependent plasticity (STDP) refers to the biological process that modifies neural synaptic efficacy based on relative spike timings of a pair of pre- and post-synaptic neurons. It provides a potential hypothesis on the biological basis and functional mechanism for explaining cognitive learning and memory in the brain [89]. Central to STDP is the idea that any significant and consistent synaptic connections among neurons are expected to result in significant neural spiking dependency. Functional neural interactions discussed in this paper refers to the strength conjoining a pair of neurons in a network likelihood model which takes into account inputs from a neural ensemble toward the firing probability of a target neuron. Under this consideration, this paper aims at studying functional STDP via neuronal interactions embedded in a spiking neuron model, or specifically the network likelihood model where the functional interactions between neurons are estimated from extracelluarly recorded spike trains.

This paper is built on the network likelihood model proposed in [90], and the focus is to develop means to estimate neuronal interaction parameters embedded in the model. The proposed estimation algorithm is based on direct observations of the spatiotemporal firing patterns and as a result, it is much simplified from the (iterative) maximum likelihood methods [90] and requires few assumptions. This paper also attempts to identify neuronal firing patterns in a rat's motor cortical neural ensemble as the rat learned to perform a directional control task. Such spatiotemporal patterns are examined via the functional interconnection strengths embedded in the network likelihood model.

The cellular basis of neuronal interactions or specifically STDP is an intense area of investigation in neuroscience. It involves the examination of the synaptic efficacy between a pre- and a post-synaptic neuron [91]. From a computation perspective, the simple correlation measures of neural activities reveal dependencies of spike timings between two spike trains. There have been several popular and different approaches to computing temporal dependencies among neurons. The correlation coefficient is probably the most common [92]. The strength of such correlation coefficients reveals coincidences of spike timings in a neural ensemble. Furthermore, statistics based on inter-spike intervals (ISI) from spike timings of multiple neurons are also important descriptions of neuronal interactions. Examined in a frequency domain using Fourier transform, ISIs correspond to phase lags in the frequency domain. Thus, a phase lag index (PLI) was proposed to quantify synchronization and functional connectivity among multiple channel EEGs and MEGs in [93]. Using probability and information theory, neuronal interactions can be measured by mutual information of spike timings in a neural ensemble [94]. More generally, Granger causality [95], a powerful method revealing the causal relationship between two time series, has been applied to estimating neuronal interactions [96]. However, all these measures discussed above are only good mathematical indicators of pair-wise interactions between two neurons. As discussed in [90], the pair-wise measurement is incapable of taking into account the dependencies on other neurons in the ensemble.

A class of maximum likelihood methods was first proposed in [97] for the simultaneous analysis of multiple pair-wise interactions of neurons in an ensemble. In [90], the authors proposed a discrete-time version of one of those maximum likelihood methods. In the meantime, the authors created a new approach to computing the starting values and the stopping criterion in the iterative maximum likelihood method. In this study, we aim at using the same discrete-time network likelihood model as in [90]

to study neuronal interactions based on spike trains of a neural ensemble to go beyond pair-wise neural interactions.

Specifically in this paper, we introduce a new and easy to implement computational approach to estimate the interconnection strength parameters in the network likelihood model. Actually, our results can be extended to spiking neuron models more general than the network likelihood model. As will be shown, the method relies on identifying repeated firing patterns in a statistically significant sense and therefore, leads to accurate parameter estimation as long as a sufficiently long recording is available. We will derive a clear bound on the data length for accurate parameter estimation. This new method is easy to implement, and it accounts for simultaneous firing of multiple neurons within a history window. This consideration is more consistent with neurophysiology since restricting multiple neurons from spiking in a history window, which spans at least 10 to 100 msec, is not always realistic. Therefore, the assumption of only one spike fires in a history window as needed in [90] can be removed in this study. In addition, this new method of parameter estimation provides us an opportunity to study functional neural plasticity in an ensemble to relate the mechanics of brain activity to measurable learning control behavior of rats.

6.2 Neuronal Interaction Represented in a Spike Train

Here in this study, we broadly refer to spiking neuron models as the general class of models aiming at increasing the level of realism in a neural simulation of spiking neurons. In the following, we introduce some model constructs that are closely related to the network likelihood mode that is considered in our study. Generally speaking, these spiking neuron models consider firing an action potential when the neuron's membrane potential reaches a specific value. When a neuron fires, its action potential travels to other neurons and via synaptic connections between neurons in the ensemble, in turn, it increases or decreases other neurons' potentials. The Hodgkin-Huxley model is widely regarded as one of the most important, neurophysiological description on how neural

action potentials are initiated and propagated. It is a set of nonlinear ordinary differential equations that approximates the electrical characteristics of excitable neuron cells. Several simplified neuronal models have also been developed, which also facilitate mathematical insight into dynamics of realistic action potential generation.

The spike response model (SRM) [98] is reduced from the Hodgkin-Huxley equations. As a result, it is a single-variable threshold model. The model is approximated by a response kernel expansion in terms of a single variable, the membrane voltage. It was shown a good approximation to the Hudgkin-Huxley model [98]. In the SRM, the post-synaptic membrane potential of a neuron, denoted by $u(t)$, is described by (6.1), where t is time, $\hat{t}$ is the time of the last neuronal spiking time, η represents the action potential depolarization and hyperpolarization, w_j represents the j^{th} synaptic efficacy, t_j^f denotes the arrival time of the f^{th} spike at synapse j, and finally, ε is the time course of post synaptic action potentials. In SRM, a neuron fires an action potential or spike when $u(t)$ crosses a threshold from below. Equation (6.1) indicates that the neuronal membrane potential $u(t)$ depends on the firing history of a neural ensemble. It is also noted that higher membrane potential leads to higher probability of firing an action potential [99].

$$u(t) = \eta(t-\hat{t}) + \sum_j \sum_f w_j \varepsilon(t-\hat{t}, t-t_j^f). \tag{6.1}$$

The firing rate model in [100] describes the interdependence between the mean firing rate h_i of neuron i (post-synaptic neuron) upon its input a_i as shown in (6.2), where a_i is the sum of all incoming activities h_c (c being the pre-synaptic neuron), g is called the gain function of a neuron.

$$h_i = g(a_i). \tag{6.2}$$

Now let Γ_i be the collection of neurons with pre-synaptic inputs to neuron i, and the pre-synaptic input a_i is defined as the sum of pre-synaptic neural activities in (6.3), where $J_{i,c}$ is synaptic efficacy between pre-synaptic neuron c and post-synaptic neuron i.

$$a_i = \sum_{c \in \Gamma_i} J_{i,c} h_c. \tag{6.3}$$

The rate model of neural activity is thus described in (6.4),

$$h_i = g\left(\sum_{c \in \Gamma_i} J_{i,c} h_c\right). \tag{6.4}$$

Introducing time into (6.4), the rate model relating neural mean firing rate to neural ensemble activity is thus described by (6.5),

$$h_i(t+1) = g\left(\sum_{c \in \Gamma_i} J_{i,c} h_c(t)\right). \tag{6.5}$$

Note that the recurrent nature of (6.5) actually has resulted in that the target neuron's firing rate is affected by neural activity history of a neural ensemble. The $J_{i,c}$ in (6.5) may be viewed as a functional synaptic efficacy between two extracellularly recorded neurons. The w_j in the SRM model of (6.1) however, directly reflects the physiological coupling strength or synaptic efficacy of two neurons. Since a neuron's firing rate and a neuron's firing probability both reflect a computed property indicating the likelihood or frequency of a neuron firing a spike, we therefore consider the firing rate model in (6.5) actually relates a neuron's spike firing probability with neural activity history of an ensemble.

In this study, based on the network likelihood model [90], we also consider the firing probability of a target neuron as a function of neural activity history of a neural

ensemble via functional synaptic efficacy. To proceed, let H_t be the neural activity history of an ensemble of C neurons,

$$H_t = \left\{ I_{1,1}(t), \cdots, I_{1,M}(t), I_{2,1}(t), \cdots, I_{2,M}(t), \cdots, I_{c,m}(t), \cdots, I_{C,1}(t), \cdots, I_{C,M}(t) \right\}. \tag{6.6}$$

where at time t, $I_{c,m}(t)$ represents the number of spikes generated by neuron c at the m^{th} time bin during the time window $[t-mW, t-(m-1)W)$. Here, we let W and M be the bin width and the number of bins in an observation window, respectively. In this paper, $W = 1$ msec is used as the default value in all results reported unless stated otherwise. Therefore, H_t defined in (6.6) denotes a collection of $I_{c,m}(t)$'s in the interval $[t-MW, t)$. For $t_1 \neq t_2$, we define two identical histories $H_{t1} = H_{t2}$ if and only if

$$I_{c,m}(t_1) = I_{c,m}(t_2), 1 \leq c \leq C, 1 \leq m \leq M. \tag{6.7}$$

In the network likelihood model [90], for a given firing history H_t as defined in (6.6), the firing probability density of neuron i at time $[t, t+\Delta t)$, denoted by $\lambda_i(t+\Delta t|\alpha_i, H_t)$, is formulated in (6.8),

$$\lambda_i(t+\Delta t|\alpha_i, H_t) = \exp\left[\alpha_{i,0} + \sum_{c=1}^{C} \sum_{m=1}^{M} \alpha_{i,c,m} I_{c,m}(t) \right]. \tag{6.8}$$

where $\alpha_{i,0}$ is neuron i's spontaneous firing probability density that is not related to the firing history of other neurons, $\alpha_{i,c,m}$ is the functional neural interaction strength or functional synaptic efficacy from neuron c to i in observation window m, and

$$\alpha_i = \left[\alpha_{i,0} \cdots \alpha_{i,c,m} \cdots \alpha_{i,C,M} \right]. \tag{6.9}$$

Therefore according to (6.8), given α_i, identical H_t's defined in (6.7) lead to the same $\lambda_i(t+\Delta t|\alpha_i, H_t)$. Let neuron i's firing probability in time interval $[t, t+\Delta t)$ be

denoted by P_t. It can thus be determined by (6.10), where Δt is the time resolution used in calculating P_t. As discussed earlier, $\Delta t = 1$ msec is used in this paper.

$$P_t = \lambda_i(t + \Delta t | \alpha_i, H_t)\Delta t. \tag{6.10}$$

In this study, we can actually generalize our results from the network likelihood model in [90] to one with a more general gain function g as shown in (6.11),

$$\lambda_i(t + \Delta t | \alpha_i, H_t) = g\left[\alpha_{i,0} + \sum_{c=1}^{C}\sum_{m=1}^{M} \alpha_{i,c,m} I_{c,m}(t)\right]. \tag{6.11}$$

Specifically as will be shown below, the gain function g can be more general than an exponential as used in [90], or a sigmoid used in [100]. In this paper, we only require the gain function g to be monotonically increasing. Therefore this generalization may include the actual action potential initiation and firing characteristics as a special case.

From equation (6.11), the neuronal firing probability during $[t, t + \Delta t)$ is a function of the spontaneous firing rate $\alpha_{i,0}$, the functional synaptic interaction strength, $\alpha_{i,c,m}$, between a pair of neurons in the ensemble, and the firing history of the ensemble. The parameter $\alpha_{i,c,m}$ thus places a weight on the spike timing dependency. Once these functional synaptic efficacies are examined while rats learned to perform a behavioral control task, they may provide a functional specification of the spike-timing-dependent plasticity (STDP) based on *in vivo* single unit recordings.

6.3 Estimating Synaptic Efficacy using Iterative Maximum Likelihood

A recursive maximum likelihood estimation of neural interaction strength, or functional synaptic efficacy $\alpha_{i,c,m}$ in (6.8), was proposed in [90]. Using artificial neural spike trains, it was proved that this method was able to preserve direct functional connectivity of a neural ensemble. Under an assumption that only one spike fired

in a history window, the maximum likelihood estimation of α_i in equation (6.9) is determined iteratively from the following procedures (6.12), (6.13) and (6.14), where $j = (c-1)M + m$, $N_i(t)$ is the number of spikes fired by neuron i up to time t, $N_{i,k:k+1} = N_i((k+1)\Delta t) - N_i(k\Delta t)$, $k = 0, 1, \cdots, K$, $K = \lceil T/\Delta t \rceil$, T is the length of the spike train, Δt is time resolution, and n is the iteration numbers.

$$\gamma_{i,j} = \exp(\alpha_{i,j}) = \exp(\alpha_{i,c,m}). \tag{6.12}$$

$$\beta_{i,j} = \frac{\sum_{k=0}^{K-1} I_{j,k} N_{i,k:k+1}}{\sum_{k=0}^{K-1} I_{i,j} \left[\sum_{l=0}^{D} I_{l,k}\right] N_{i,k:k+1}}. \tag{6.13}$$

$$\gamma_{i,j}^{(n+1)} = \gamma_{i,j}^{(n)} \left[\frac{\sum_{k=0}^{K-1} I_{j,k} N_{i,k:k+1}}{\sum_{k=0}^{K-1} I_{i,j} \left[\prod_{l=0}^{D} \left(\gamma_{i,j}^{(n)}\right)^{I_{i,j}}\right] \Delta t} \right]^{\beta_{i,j}}. \tag{6.14}$$

It is noted that, the iterative maximum likelihood method described by equations (6.12), (6.13) and (6.14) make use of model (6.8) where the gain function is exponential. It may not be easy to derive at a similar procedure as in (6.12), (6.13) and (6.14) if the gain function becomes more general than an exponential.

In the following, we will develop a new procedure to estimate the synaptic efficacy parameter α_i for a generalized network likelihood model in (6.11), by which we mean the gain function g in (6.11) can be more general than exponential as used in (6.8) as long as it is monotonically increasing. We will make use of the new procedure to analyze synaptic plasticity represented in the model parameters α_i in conjunction with a rat's behavioral learning process.

6.4 Estimating Spike Firing Probability in the Generalized Network Likelihood Model using a Perceptron Bank

We now develop a new approach to the problem of estimating neuronal interactions based on the generalized network likelihood model in equation (6.11). Given that Δt is small, actually 1 msec in our analysis of the experimental data, firing an action potential within this small window $[t, t+\Delta t)$ for neuron i can be viewed as a Bernoulli trial. As such, consider r, which is a random number from a uniform distribution in $[0,1]$, the spike generation with a firing probability P_t can be modeled by (6.15) as discussed in [101].

$$\begin{cases} \text{neuron } i \text{ fires if} & P_t = \lambda_i(t|\alpha_i, H_t)\Delta t > r, \\ \text{neuron } i \text{ remains silent if} & P_t = \lambda_i(t|\alpha_i, H_t)\Delta t \leq r. \end{cases} \tag{6.15}$$

Next consider a perceptron network defined in (6.16) where y is the output of the perceptron network while x_v $(1 \leq v \leq V)$ is the input, w_v $(1 \leq v \leq V)$ is the weight, b is the bias [102].

$$y = \begin{cases} 1 & \text{if } \sum_{v=1}^{V} w_v x_v + b > 0, \\ \\ 0 & \text{if } \sum_{v=1}^{V} w_v x_v + b < 0. \end{cases} \tag{6.16}$$

Center to the idea of approximating the neural firing probability in (6.11) is to use a set of such perceptron networks described in (6.16), or a perceptron bank, to re-produce the random experiment defined in (6.15). Namely, the perceptron training aims at obtaining parameters w_v, x_v and b in a perceptron bank to correspond with $\alpha_{i,c,m}$, $I_{c,m}(t)$ and $\alpha_{i,0}$. By doing so, a bank of deterministic perceptron networks each of which described by (6.16) can be used to approximate a random spiking process.

The key implementation steps can then be developed as follows. Given a neural spike train, the spike histories are classified into S clusters for identical H_t's defined in (6.7), i.e., the histories in the $s^{th}(1 < s < S)$ cluster, denoted by H^s, are identical. Its cardinality is denoted by $N(s)$ as in (6.17).

$$N(s) = \text{card}\{H^s \mid \text{number of identical } H_t\text{'s in the } s^{th} \text{ cluster}\}. \tag{6.17}$$

Note that the trivial cluster of H_t without any spikes is not considered one of the S firing patterns. Given α_i, as indicated in (6.11), the spiking history H_t affects the spike firing probability P_t, and that $P_{t_1} = P_{t_2}$ if $H_{t_1} = H_{t_2}$ $(t_1 \neq t_2)$. Let P^s be the neural firing probability given $H_t \in H^s$. Each firing history in H^s and its associated spiking state (fire or not) inside time window $[t, t+\Delta t)$ can be used as input and desired output, respectively to train one local perceptron. For multiple training data sets from multiple time segments, multiple perceptrons consisting of a perceptron bank can be trained in the same way and then used to estimate the strength of neural interactions.

Let $H_t^s(l)$ $(1 \leq l \leq N(s))$ denote the l^{th} element in H^s, i.e., $H_t^s(l) \in H^s$. Let $w_{v,l}$ $(1 \leq l \leq N(s), 1 \leq v = (c-1)M + m \leq V = CM)$ be the weight w_v learned in the l^{th} local perceptron network corresponding to $H_t^s(l)$. Let $y_l(t+1)$ be neuron i's firing state at time interval $[t, t+\Delta t)$ given $H_t^s(l)$. The error between the perceptron network output and the desired output defined by $e = y_l(t+1) - y$ can be used to train the network. The training procedure is summarized below in the table of Algorithm 1.

Given a training data set $H_t^s(l)$ and its corresponding desired output $y_l(t+1) = 1$, the output of the network with updated weights $w_{v,l} = I_{c,m}(t)$ is 1 according to (6.18) since only nontrivial cluster of H_t are considered.

$$y = \sum_{v=1}^{V} w_{v,l} I_v(t) = \sum_{v=1}^{V} I_v(t)^2 > 0. \tag{6.18}$$

Algorithm 1 Local perceptron network training

initialize $w_{v,l} = 0, v = (c-1)M+m,\ 1 \leq c \leq C, 1 \leq m \leq M$;
calculate training error e;
if $e = 0$ **then**
  no change to $w_{v,l}$;
else
  while $e \neq 0$ **do**
    update $w_{v,l} = I_{c,m}(t),\ v = (c-1)M+m, 1 \leq c \leq C, 1 \leq m \leq M$;
    calculate training error e with updated weights;
  end while
end if

On the other hand, $w_{v,l} = 0$ if $y_l(t+1) = 0$. As such, depending on the value of the desired output of being 1 or 0, the weight of the perceptron network will be either zero or equal to $I_{c,m}(t)$ in the next iteration consequently.

6.4.1 Estimating Firing Probability Given Neural Ensemble Firing History

Figure 6.1 and Figure 6.2 provide a summary for training the single perceptron network and the perceptron bank, respectively as discussed previously.

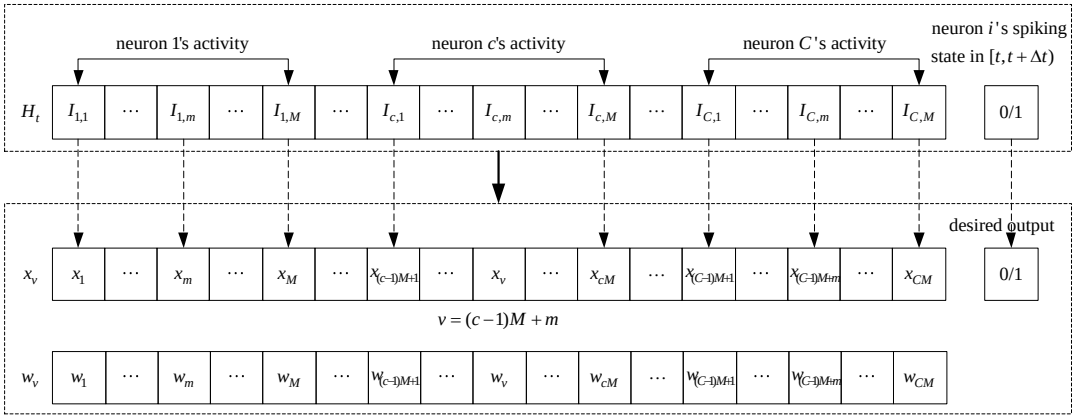


Figure 6.1: Illustration of a single perceptron network training. The neural ensemble activity $I_{c,m}(t)$ is used as the perceptron network's input denoted by x_v where $v = (c-1)M+m$. The respective weight w_v is trained with respect to the desired output which is neuron i's firing state at $[t, t+\Delta t)$.

In this study, the probability of firing a spike in $[t, t+\Delta t)$ given a spike history $H_t^s \in H^s$ is considered a Bernoulli trial with probably P^s. According to (6.15), a spike train can be generated and simulated by a Binomial process with its Bernoulli

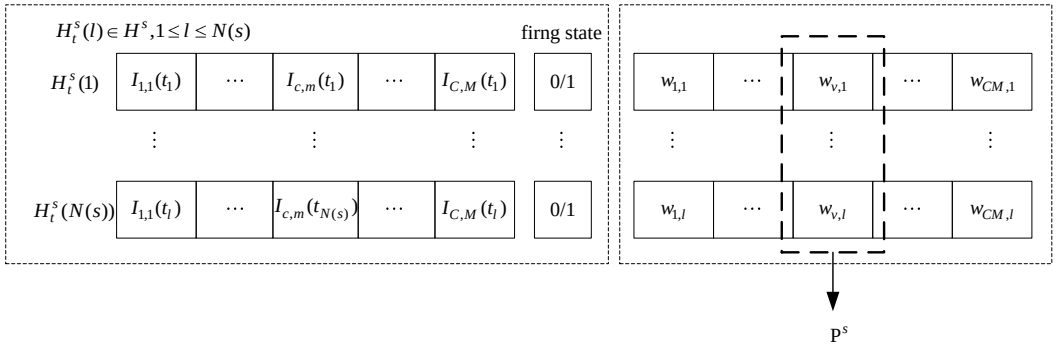


Figure 6.2: Illustration of perceptron bank training. Each firing history $H_t^s(l) \in H^s$ corresponds to the weights of one local perceptron network. After training, the weight $w_{v,l}$ corresponding to $\alpha_{i,c,m}$ is equal to one or zero depending on neuron i firing or not at $[t, t+\Delta t)$. The mean of $\{w_{v,l}\}$ over the perceptron bank is neuron i's firing probability given H^s denoted by P^s as shown in (6.19), where s is the cluster index.

parameter $p = P^s$ given H^s. If we allow $N(s)$ in (6.7) to be sufficiently large, the Binomial distribution can be approximated by a Gaussian distribution. This condition can be relatively easily satisfied since it corresponds to increasing the length of the neural recording. For a Gaussian distribution, its mean also is the maximum likelihood estimate of P^s, the estimation confidence interval is determined by its standard deviation.

In order to estimate the probably P^s from the perceptron bank, we take into account the fact that after training, the weight vector in a perceptron network corresponds to the firing history H_t^s. Also, the total number of $N(s)$ identical firing histories in H^s is the total number of perceptron networks in the s^{th} perceptron bank. Therefore, the perceptron bank can be viewed as an outcome map of the stochastic event of neuron i^{th} firing profile as a function of the firing history from a neuron ensemble. The neuron i^{th} firing probability can then be estimated as the fraction of the total number of firing events $n(s)$ over the total number of perceptron networks $N(s)$, i.e.,

$$P^s = \frac{n(s)}{N(s)}. \tag{6.19}$$

The estimation of the neuronal firing probability according to (6.19) is a result of the law of the large numbers. It is easy to prove theoretically that our estimation method in (6.19) asymptotically approaches the true value of α_i. For other methods, such as the iterative maxim likelihood (ML) method [90] and maximum a posteriori (MAP) [103], it is not obvious or intuitive to draw the same conclusion. Note that, the estimation algorithm based on generalized linear model (GLM) [104] as well as the maximum likelihood method [90] make use of the exponential transfer function as the link function. The estimation method may need to be modified if other link functions are to be used in the respective models. However, in our proposed perceptron bank approach, the gain function in (6.11) can be more general and it does not need to be known, so long as it is a monotonically increasing, which includes the exponential link function. This is because as long as the gain function is monotonically increasing, neuron i^{th} firing probability according to (6.11) is proportional to the synaptic efficacy to neuron i. Therefore, the statistical significance of neuron i^{th} spike firing probability is proportional to the statistical significance of the synaptic efficacy to neuron i. Thus. the proposed method can be considered a generalized method for estimating neural synaptic efficacy. Note that in the next subsection, we will clearly define statistical significance of a synaptic efficacy.

6.4.2 Estimating Synaptic Efficacy $\alpha_{i,c,m}$

Once neural firing probability P^s is estimated from (6.19), then a set of linear equations can be formed according to (6.20) by integrating (6.11) and (6.10) with (6.19).

$$g^{-1}(\frac{P^s}{\Delta t}) = \left[\alpha_{i,0} + \sum_{c=1}^{C}\sum_{m=1}^{M} \alpha_{i,c,m} I_{c,m}(t)\right]. \tag{6.20}$$

Define $\mathbf{I_H}$, α_i, and $\mathbf{P}$ in (6.21) , (6.9) and (6.23), respectively.

$$\mathbf{I_H} = \begin{bmatrix} 1 & \cdots & 1 & \cdots & 1 \\ \vdots & & \vdots & & \vdots \\ I^1_{c,m} & \cdots & I^s_{c,m} & \cdots & I^S_{c,m} \\ \vdots & & \vdots & & \vdots \\ I^1_{C,M} & \cdots & I^s_{C,M} & \cdots & I^S_{C,M} \end{bmatrix} \tag{6.21}$$

$$\mathbf{P} = \left[g^{-1}(\frac{P^1}{\Delta t}) \cdots g^{-1}(\frac{P^s}{\Delta t}) \cdots g^{-1}(\frac{P^S}{\Delta t}), \right]^T \tag{6.22}$$

then (6.20) becomes

$$\alpha_i \mathbf{I_H} = \mathbf{P}. \tag{6.23}$$

Neural synaptic efficacy in (6.9) can be solved as least squares solution to (6.23) or specifically,

$$\alpha_i = \mathbf{I_H}^+ \mathbf{P}, \tag{6.24}$$

where $\mathbf{I_H}^+$ is the pseudo-inverse of $\mathbf{I_H}$. This least-squares solution is clearly simpler and more intuitive than the computations presented in (6.13) and (6.14) which require the assumption that only one spike fires in a history window as indicated in [90].

The estimated P^s in (6.19) asymptotically approaches the true spike firing probability modeled in (6.11) as $N(s)$ increases. To achieve a desired approximation accuracy, we would like to quantitatively determine the minimum $N(s)$ value. As discussed previously, given a spike firing history H^s_t, we consider the spiking outcome, b_n, in $[t, t+\Delta t)$ a Bernoulli trial, where $1 \leq n \leq N(s)$. Therefore $b_n = 1$ corresponds with the firing of a spike while $b_n = 0$ corresponds with no spike firing status. The probability of spike firing can then be modeled as a Binomial process,

As $N(s)$ increases, or practically for $N(s) > 5$, if condition (6.25) holds as shown in [105],

$$\left| \frac{1}{\sqrt{N(s)}} \left(\sqrt{\frac{1-P^s}{P^s}} - \sqrt{\frac{P^s}{1-P^s}} \right) \right| < 0.3, \tag{6.25}$$

the Binomial variable B can be approximated by a Gaussian process, where the parameters of the Gaussian process is determined by (6.26) and (6.27) below.

$$B = \frac{1}{N(s)} \sum_{n=1}^{N(s)} b_n \sim N(\mu, \sigma^2), \tag{6.26}$$

$$\begin{cases} \mu = & \lambda_i(t+\Delta t|\alpha_i, H_t)\Delta t \\ \sigma^2 = & N(s)\mu(1-\mu) \end{cases} \tag{6.27}$$

Table 6.1: Adequate $N(s)$ for Gaussian Approximation in (6.25)

Mean firing rate (Hz)	P^s (msec)	$N(s)$
5	0.005	2189
50	0.05	189
100	0.1	79

Table 6.1 shows the adequate $N(s)$s for different mean firing rates. To quantitatively determine an appropriate $N(s)$ for estimating a neuron's firing probability given an ensemble firing history, we make use of measure D, which is the deviation of the estimation from the truth.

$$D(L) = \left| \frac{g^{-1}(\mu + L\sigma) - g^{-1}(\mu)}{g^{-1}(\mu)} \right|. \tag{6.28}$$

For a given set of Gaussian parameters μ and σ in (6.27), L is an integer signifying the number of standard deviation away from the mean, g is an exponential function.

Table 6.2 illustrates different D values as a function of $N(s)$ and L. When $\mu = 0.05$, which corresponds to a neuron's mean firing rate of 50 Hz, and $N(s) = 3000$, from (6.28) we obtain that $D(-3) = 9.11\%$ and $D(3) = 7.15\%$. Making use the Gaussian approximation discussed above, the probability that D is beyond 9.11% is less than 0.27%, i.e., $Prob(D > 9.11\%) = Prob(|P^s - P| > 9.11\%) < 0.27\%$. Therefore in this study, the ensemble history with $N(s) > 3000$ is selected when estimating a neuron's firing probability P^s.

Table 6.2: Relationship between $N(s)$ and P^s estimation accuracy

$N(s)$	L	$D(-L)$	$D(L)$	probability
1000	2	10.77%	8.13%	4.55%
1000	3	17.81%	11.55%	0.27%
2000	2	7.24%	5.94%	4.55%
2000	3	11.55%	8.56%	0.27%
3000	2	5.79%	4.93%	4.55%
3000	3	9.11%	7.15%	0.27%

6.4.3 Statistically Significant Synaptic Efficacy

In neuroscience, excitatory post-synaptic potential (EPSP) refers to the post-synaptic neural membrane potential depolarization due to positive ion flows from pre-synaptic neurons. The membrane potential depolarization is excitatory since it increases spiking firing probability of the post-synaptic neuron. On the contrary, inhibitory post-synaptic potential (IPSP) caused by negative ions flow from pre-synaptic neurons decreases post-synaptic neuron firing probability. Motivated by the concept of EPSP and IPSP, we consider the functional synaptic efficacy in (6.11) excitatory or inhibitory if it increases or decreases a target neuron's firing probability. In another word, a functional synaptic efficacy is excitatory or inhibitory if it is statistically significantly higher or lower than a baseline synaptic efficacy corresponding to the mean firing rate. To obtain the baseline value of a synaptic efficacy, we consider a randomly shuffled spike train to

abolish any inherent neuronal interactions. The estimated interactions from a shuffled spike train can thus be used as the baseline reference corresponding to insignificant interaction. Once randomly shuffled, a neuron's firing probability in $[t, t+\Delta t)$ becomes independent of the spiking history due to shuffling, but only dependent on the neuron's mean firing rate. The significance of a synaptic efficacy embedded in a spike train should thus be determined by comparing the estimated functional efficacy with that embedded in the shuffled spike train, which corresponds with the mean firing rate.

In this study, an excitatory or inhibitory interaction is claimed if its synaptic efficacy value is significantly $(p = 0.1\%)$ greater or less than that estimated from the corresponding shuffled spike train. Similarly, an insignificant interaction is defined when it is not significantly $(p = 0.1\%)$ different from that estimated from the corresponding shuffled spike train.

6.4.4 Significant Functional Efficacy $\alpha_{i,c,m}$ and Direct Link between a Neuron Pair

The functional efficacy $\alpha_{i,c,m}$ in (6.11) signifies neuron c's direct influence on neuron i's firing probability. Consider neuron i's conditional firing probability $\lambda_{c\to i}$ taking into account only neuron c's input to target neuron i, i.e.,

$$\lambda_{c\to i}(t+\Delta t|\alpha_i, H_t) = g\left[\alpha_{i,0} + \sum_{m=1}^{M} \alpha_{i,c,m} I_{c,m}(t)\right]. \tag{6.29}$$

Comparing (6.11) and (6.29), it is not difficult to notice that $\lambda_{c\to i}$ is a special case of λ_i if only neuron c fires in H_t as shown in (6.30).

$$\lambda_{c\to i} = g\left[\alpha_{i,0} + \sum_{c=1}^{C}\sum_{m=1}^{M} \alpha_{i,c,m} I_{c,m}(t)\right] \Bigg| \begin{array}{l} I_{c',m}(t) = 0, \\ 1 \le c' \ne c \le C. \end{array} \tag{6.30}$$

Under this condition, no other neurons $c' \neq c$ contributes to the input of target neuron i to influence neuron i's firing probability. Thus $\alpha_{i,c,m}$ measures the direct

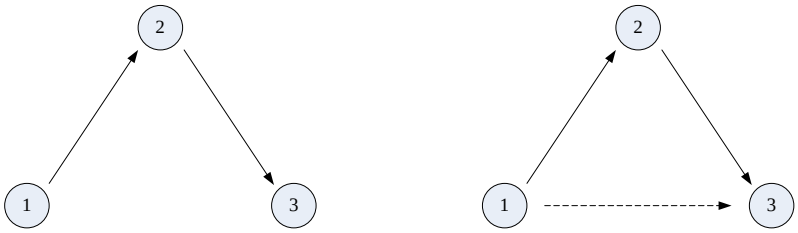


Figure 6.3: (a) Interactions among neurons 1, 2, and 3: true direct causal connections from 1 to 2, and from 2 to 3, but no direct link between 1 and 3. The indirect causal influence from 1 to 3 is only via 2. (b) Granger causality is unable to differentiate the nature of direct or indirect causal influence [106]. Similarly, cross correlation, mutual information also are unable to differentiate direct from indirect link such as that from 1 to 3 shown in dashed line.

influence from neuron c to i. It is also noticed that the functional efficacy $\alpha_{i,c,m} \neq 0$ actually reflects direct interactions between neurons i and c, and that there is no direct link between i and c if $\alpha_{i,c,m} = 0$. This property specially associated with model (6.11) is preferred since other correlation based interaction measures such as correlation coefficient, Granger causality, and other similar methods cannot distinguish if two neurons are functionally directly linked or not.

6.4.5 Direct Link among Neurons Reduce False Positive Connection

To see how $\alpha_{i,c,m} \neq 0$ in (6.11) directly links neurons i and c, consider the example in Figure 6.3a where neuron 3 is directly linked to neuron 2, but not neuron 1. According to (6.11), neuron 3's firing probability density given firing history H_t and neuronal interaction vector α_i in (6.11), denoted by $\lambda_i(t+\Delta t|\alpha_i, H_t)$, can be modeled by (6.11) and re-written in (6.31) below,

$$\lambda_3(t+\Delta t|\alpha_3, H_t) = g\left[\alpha_{3,0} + \sum_{c=1}^{3}\sum_{m=1}^{M} \alpha_{3,c,m} I_{c,m}(t)\right]. \tag{6.31}$$

According to Figure 6.3a, neuron 2 may fire spontaneously or due to input from neuron 1. Other than its own spontaneous activities, neuron 3's firing probability di-

rectly depends on input from neuron 2. Such network connectivity is expected to be represented in equation (6.11) where ideally $\alpha_{2,1,m}$ and $\alpha_{3,2,m}$ should be significantly different from zero, while $\alpha_{3,1,m}$ should be zero in (6.31). To compute $\alpha_{3,1,m}$ and $\alpha_{3,2,m}$ using the currently proposed estimation method, we consider the following two scenarios. First, if neuron 1 does not fire, then neuron 1 has no direct impact on the firing probability of neuron 3 according to (6.11). Second, if neuron 1 fires, by the following derivations, we will show that neuron 3's firing probability will also only depend on neuron 2's firing activities. To start, let $\hat{\lambda_3}(t+\Delta t|\alpha_3, H_t)$ be the true estimate of neuron 3's firing probability based on (6.11) given the neural network connectivity in Figure 6.3a,

$$\hat{\lambda_3}(t+\Delta t|\alpha_3, H_t) = g\left[\alpha_{3,0} + \sum_{m=1}^{M} \alpha_{3,2,m} I_{2,m}(t) + \sum_{m=1}^{M} \alpha_{3,3,m} I_{3,m}(t)\right]. \tag{6.32}$$

To see that (6.32) actually is the result of our proposed parameter estimation algorithm, we proceed as follows. Equate (6.31) and (6.32), we find that

$$\begin{aligned}
g\left[\alpha_{3,0} + \sum_{c=1}^{3}\sum_{m=1}^{M} \alpha_{3,c,m} I_{c,m}(t)\right] &= g\left[\alpha_{3,0} + \sum_{m=1}^{M} \alpha_{3,2,m} I_{2,m}(t) + \sum_{m=1}^{M} \alpha_{3,3,m} I_{3,m}(t)\right] \\
\left[\alpha_{3,0} + \sum_{c=1}^{3}\sum_{m=1}^{M} \alpha_{3,c,m} I_{c,m}(t)\right] &= \left[\alpha_{3,0} + \sum_{m=1}^{M} \alpha_{3,2,m} I_{2,m}(t) + \sum_{m=1}^{M} \alpha_{3,3,m} I_{3,m}(t)\right] \\
\sum_{c=1}^{3}\sum_{m=1}^{M} \alpha_{3,c,m} I_{c,m}(t) &= \sum_{m=1}^{M} \alpha_{3,2,m} I_{2,m}(t) + \sum_{m=1}^{M} \alpha_{3,3,m} I_{3,m}(t) \\
\sum_{m=1}^{M} \alpha_{3,1,m} I_{1,m}(t) &= 0.
\end{aligned} \tag{6.33}$$

Under the second case scenario, it is apparent that there exists at least one m value such that $I_{1,m}(t) \neq 0$. Therefore, for (6.33) to hold it has to be true that $\alpha_{3,1,m}$ is 0.

We have thus verified that using the currently proposed estimation algorithm, the synaptic efficacy parameters in (6.11) actually preserve the direct functional neural connectivity in the network.

From another perspective, the direct functional interaction between two neurons may be viewed as a first order Markov process where the current state only depends on the immediate past state. Therefore, referring to Figure 6.3b, the direct functional dependency of neuron 3 on neurons 1 and 2 is represented by an insignificant interaction ($\alpha_{3,1,m} = 0$) between neurons 3 and 1 but a significant interaction ($\alpha_{3,2,m} \neq 0$) between neurons 3 and 2. Actually our derivations shown in (6.33) illustrated this point.

In deriving (6.33), we assume that neuron 3's conditional probability density given neural ensemble activities can be estimated accurately. With the law of large numbers, the conditional probability can be estimated by relative frequency which approaches the truth asymptotically. Practically, this estimation based on the law of large numbers approximates the truth with some errors. To avoid false positive from estimation, the significance of a direct link is determined by comparing it with the mean firing rate in section 6.4.3. The theoretical analysis on reducing a false positive connection is discussed in the current section. The artificial neural spikes with known direct links are used to evaluate the estimation performance in the next section.

6.5 Estimating Neuronal Interactions from Artificially Generated Spike Trains

To illustrate how model (6.11) can be used to predict a neuron's conditional firing probability and how the proposed algorithm can effectively provide an accurate estimate of the interaction parameters $\alpha_{i,c,m}$'s, we first make use of artificial neural spike trains where the artificially created neural networks have known interconnection topology and interaction strength. After that, we will use model (6.11) and the associated parameter estimation algorithm proposed in this paper to analyze cortical recordings from rat's motor cortical areas in relation to the rat's cognitive learning control behavior.

6.5.1 Generating Artificial Spike Trains with Known Neuronal Interaction

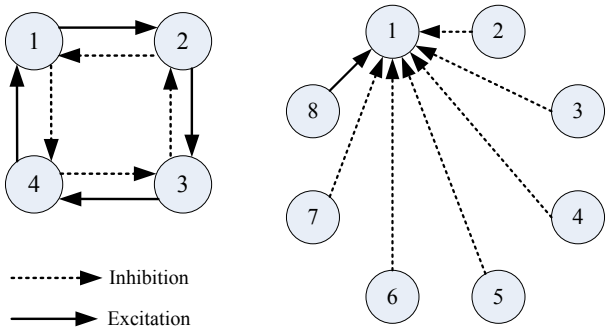
The artificial neural data sets with known neuronal interactions were generated the same way as in [90, 101]. Three artificial neural networks with different topologies are shown in Figure 6.4. By using the same neural network configurations, it is convenient to compare parameter estimation performances and modeling results between the proposed method in this study and the maximum likelihood (ML) method used in [90]. In Figure 6.4a, each neuron provides an excitatory input to the next neuron in its clockwise direction and an inhibitory input to the next neuron in its counterclockwise direction. It was noted that neurons 1 and 3 interacted indirectly via neuron 2. The second artificial neural network is made up of 8 neurons (Figure 6.4b), all neurons are interconnected with each other. One neuron excites the next neuron in its clockwise direction and receives inhibitory input from other neurons. To avoid laying out 8 interconnected neurons, only one neuron's inter-connectivity is illustrated in Figure 6.4a. The third network with 20 neurons is shown in Figure 6.4c. Interconnectivity among neurons are as illustrated in the figure. For all three networks, additional details on data generation and model parameter assignments can be found in [90]. The excitatory and inhibitory interactions used in the three different topologies (Figure 6.4a, 6.4b, 6.4c) are defined in (6.34) and (6.35), respectively.

$$\alpha^{+}(t) = 2sin(2\pi 0.06^{-1}t)\exp(-0.04t^{-1}). \tag{6.34}$$

$$\alpha^{-}(t) = -3sin(2\pi 0.12^{-1}t)\exp(-0.04t^{-1}). \tag{6.35}$$

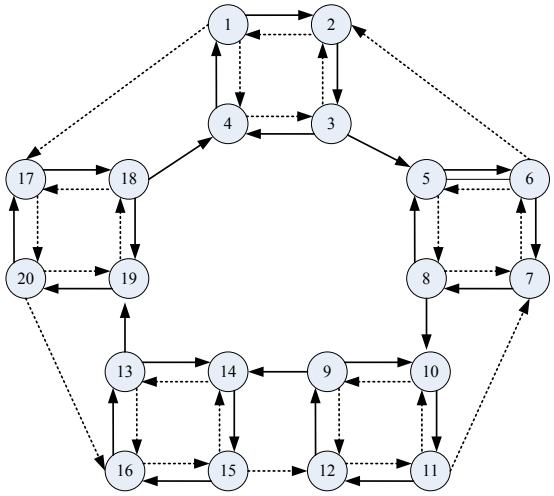
6.5.2 Estimation Performance using Artificial Neural Spike Trains

Artificial neural spike trains were generated according to (6.15), (6.34), and (6.35). For the network with 8 and 20 neurons, the spontaneous firing rates $\alpha_{i,0}$ was set to 5 Hz,



(a) An artificial neural network with 4 neurons.

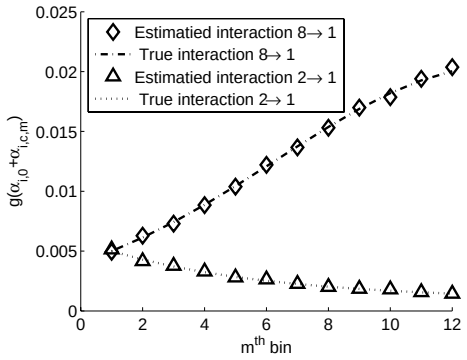
(b) An artificial neural network with 8 neurons.



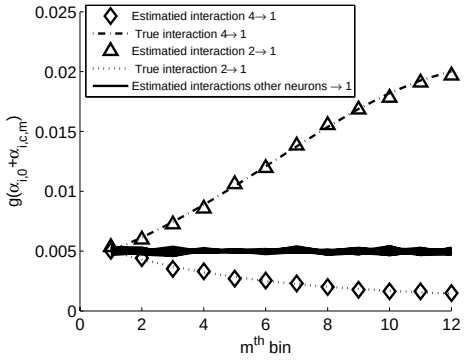
(c) An artificial neural network with 20 neurons.

Figure 6.4: The topology of artificial neural networks with different numbers of neurons.

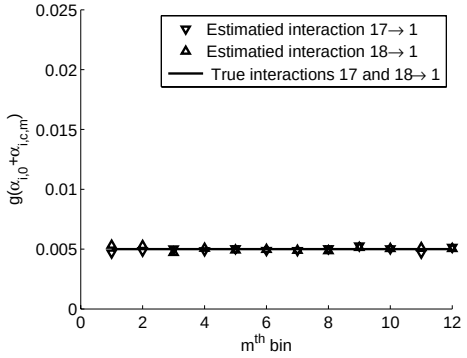
which is equivalent to 0.005 spikes / msec. The window length parameter $M = 12$ was chosen the same way as in [90]. The bin size W of 1 msec was used instead of 10 msec used in [90]. Actually the functional interaction estimated from using large bin sizes can be viewed as integration of the same parameter with small time bins. As a result, the large bin size would have a smooth impact on estimation errors. A small bin size would magnify estimation errors. In this study, a small bin size of 1 msec was used to evaluate the estimation performance of the proposed method. Within each bin, there is at most 1 spike in each bin. The equivalent neural data recording length of each spike train in the 8 and 20 neuron networks was 30 hours. For the 4 neuron network,



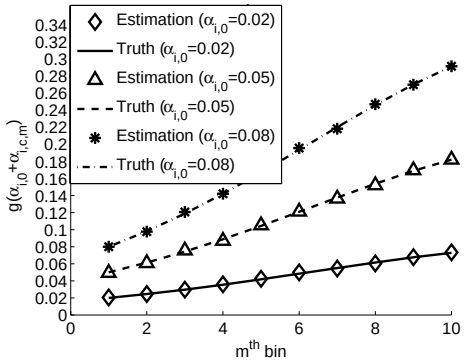
(a) Estimation results for 8 neuron artificial network.



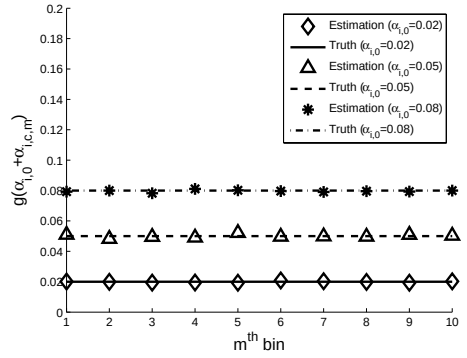
(b) Estimation results for 20 neuron artificial network.



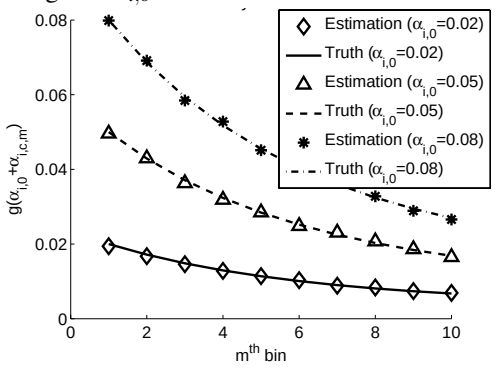
(c) Selected view for Figure 6.5b, 20 neuron network: $\alpha_{17,1,m}$ and $\alpha_{18,1,m}$ to illustrate details.



(d) Estimated excitatory interactions for 4 neuron artificial network at different spontaneous firing rate $\alpha_{i,0}$.



(e) Estimated insignificant interactions for 4 neuron artificial network at different spontaneous firing rate $\alpha_{i,0}$.



(f) Estimated inhibitory interactions for 4 neuron artificial network at different spontaneous firing rate $\alpha_{i,0}$.

Figure 6.5: Estimation performance with artificial neural networks in Figure 6.4. The vertical axis is the contribution of $\alpha_{i,0}$ and $\alpha_{i,c,m}$ to neural firing probability density. The $g(\alpha_{i,0}+\alpha_{i,c,m})$ was computed with the gain function g set to be exponential. Note however, that other monotonically increasing gain functions can also be used without changing the outcome illustrated in the figure.

three different spontaneous firing rates, 20 Hz, 50 Hz and 80 Hz, were used since the firing rate from our real neural recordings was in the same range. The corresponding spontaneous firing rate for the 4 neuron network was 0.02 spikes / msec, 0.05 spikes / msec and 0.08 spikes/ msec , respectively. The time resolution $\Delta t = 1$ msec was used for all three artificial neural networks. Insignificant interactions were expected to correspond with spontaneous firing rate as discussion in section 6.4.3.

Due to symmetry of the three neural network configurations as shown in Figure 6.4, the neural interaction efficacies from an input neuron to target neuron 1 were estimated using the proposed algorithm to illustrate results from each of the three networks in this study. Specifically in Figure 6.4b, neuron 1 is connected to all other 7 neurons, neuron 8 provides excitatory input to neuron 1 while neuron 2 provides inhibitory input to neuron 1. Figure 6.5a shows that the estimated neuronal interactions from neuron 8 and 2 to neuron 1 are close matches to the real parameters. In the 20-neuron network shown in Figure 6.4c, only neuron 2 and neuron 4 provide inputs to neuron 1, while all other neurons do not have any direct impact on neuron 1. This network topology is again well preserved by the estimation algorithm with results shown in Figure 6.5b. Insignificant interactions were clearly obtained which correspond with the lack of direct links from neurons 17 and 18 to neuron 1 as shown in Figure 6.5c. For the 4-neuron network, the excitatory, insignificant and inhibitory interactions with 3 different $\alpha_{i,0}$'s were accurately estimated as shown in Figure 6.5d, 6.5e and 6.5f, respectively.

6.6 Neuronal Interactions Estimated with Real Neural Recording

As shown in the previous section that the proposed estimation method of the synaptic efficacies in the neuron firing probability model (6.11) is a simple to implement computational approach and it also preserves network connectivity. In this section, we will show that the proposed method of estimating neuronal interaction strength can also be used to analyze cortical neural activities in relation to cognitive control behaviors.

Specifically, rat's motor cortical neural recordings were analyzed and associated with the rat's behavioral parameters in a learning decision and control task.

6.6.1 Neural Data Preparation

The behavioral experiment involves a rat working inside a Skinner box to perform a decision and control task. The rat self-paces a trial by pressing a center paddle to signal the start of a trial. Immediately after the center paddle press, one of the four LEDs (two on each side) above the control levers was lit. The rat would learn to decide which one of the two control paddles to press in order the move the LED light toward the center location. A correct press would move the light toward the center for one step. Once the light reaches the center location and remains there for 2 seconds, the trial was considered a success and the rat was given a sugar pellet reward.

Four rats named A, B, O, and W were recorded and their data used in this study. Each rat started neural recording while being naive to the task. As trial and error learning progressed, task performance accuracy for rats B, O, and W improved up to 96% in a varying length of time, but generally no more than 40 days. However, rat A was only able to learn left side trials but never performed right side trials sufficiently accurately. The neural recordings from the four rats were spike detected and sorted using algorithms developed in [107] and [108]. The stored waveforms were sorted offline into single unit action potentials using a multi-scale correlation of wavelet coefficients (MCWC) spike detection algorithm [107] followed by a template matching sorting procedure. MCWC is a high performance detection algorithm and was tested by several artificial benchmark data sets and real data with at least comparable performance to popular commercial and academic algorithms. Events in the behavioral task such as cue on, paddle release, paddle press and food reward were registered simultaneously and time stamped by the TDT system (Alachua, FL). Spike data were extracted between the time of cue on and 2 seconds after that, which typically corresponded to the time just before directional control paddle pressing. During this 2 second time period,

rats observed a cue and planed for actions. Neural spike train data were divided into two pools: a naive set and a proficient set according to the rat's behavioral performance accuracy, respectively. Behavioral performance accuracy beyond 80% is considered proficient, otherwise naive.

As discussed in the SRM model (6.1), a neuron's membrane potential depends on the most recent neural activities of a neural ensemble. The inter-spike interval (ISI) is a good reflection of the length of the neural ensemble history that should be taken into account for computing the neuron's cumulative membrane potential leading to firing a spike. The ISI value, the reciprocal of which leads to the mean firing rate of a neuron should be a good candidate for estimating the ensemble history length. In this study, without loss of generality and also comparable to the literature, we chose $M = 10$ bins as the ensemble history length to be used in (6.11), where each bin was 1 msec long.

6.6.2 Plasticity in Synaptic Efficacy as Learning Processes

We are now in a position to estimate the synaptic efficacies, $\alpha_{i,c,m}$, in (6.11) and try to reveal the pattern of changes in these synaptic efficacies as a rat's behavioral learning performance progresses to perform the designed directional control task.

First, Figure 6.6 is used to illustrate significant neural synaptic efficacies estimated based on real spike trains and used to illustrate notations used in the subsequent figures. In this study, the term "negative plasticity" is named for events such that the neural synaptic efficacies changed from excitatory to inhibitory or insignificant. On the contrary, "positive plasticity" is named for events such that an inhibitory synaptic efficacy became excitatory or insignificant.

Figure 6.7 is a summary of significant $(p = 0.1\%)$ neural synaptic efficacies for rats A, B, O, and W. A tally of each of the two types of positive or negative plasticities is shown in Figure 6.8. It clearly reveals that negative plasticity was dominant for rats B, O, and W when rats learned the tasks. The same pattern was observed for rat A's

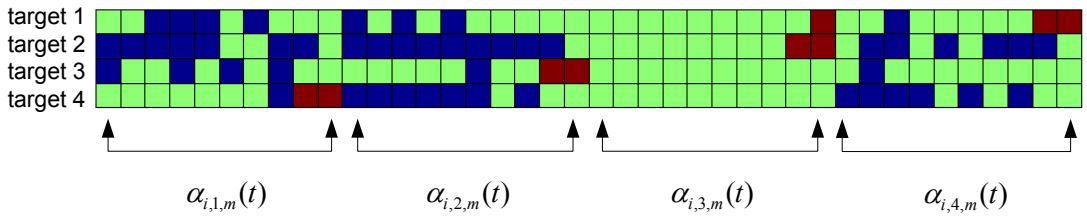


Figure 6.6: Illustration of estimated neuronal interactions in an ensemble of four neurons. Each of the four target neurons received triggers imposed upon temporally via multiple time bins and spatially from multiple neurons. The blue, green, and red colors represented significant ($p = 0.1\%$) inhibitory, insignificant and excitatory interactions, respectively.

left side trials. The only exception is that positive plasticity was dominant for rat A on the right side trials. It is interesting that rat A never learned to perform right side trials correctly. These results suggest that functional synaptic plasticity may be able to provide a neural substrate to the rat's behavioral learning process.

Based on the notion of neural synaptic efficacy, $\alpha_{i,c,m}$ in (6.11) signifies the dependency of spike timings between neurons. The adaptation of neural efficacy during behavioral learning suggests that the dependency of relative spike timing has changed as learning progresses. Such changes in functional synaptic efficacy as a result computed from *in vivo* neuronal recording indicates that relative spike timing adapted by adjusting neuronal interaction strength during the behavioral learning process. The process may be viewed as functional spike time dependent plasticity (STDP), a potential hypothesis for explaining brain adaptation during behavioral learning.

6.7 Conclusion

This study centers on analyzing functional neuronal interactions or functional synaptic efficacies as a means of revealing how neurons interact in an ensemble that underlies a behavioral learning process. Toward this end, the paper made the following significant contributions. The first of which is the development of a generalized spike firing probability based modeling technique. The associated model parameter estimation algorithm is new, which is based on analyzing spiking patterns and therefore is

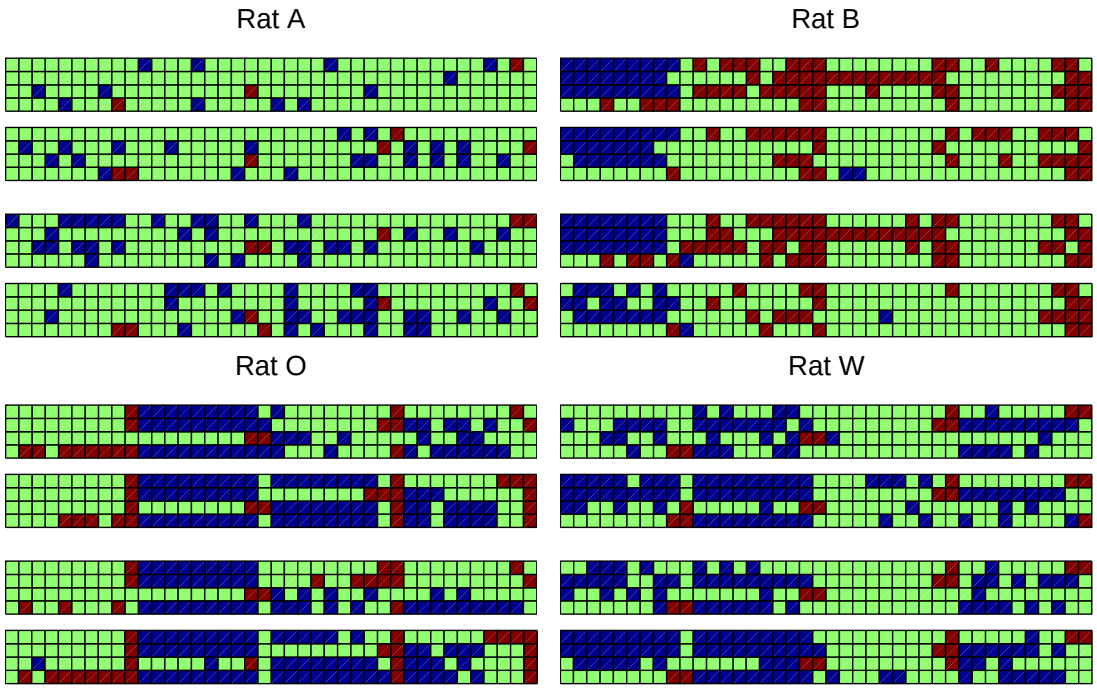


Figure 6.7: The significant neuronal interactions estimated with real neural recording data by the proposed method. Only correct trials were used in this figure. For each rat, the first two rows were the estimated interaction of naive state and proficient state for left trial, respectively. The bottom two rows were the estimated interaction of naive state and proficient state for right trial, respectively.

straightforward. The proposed estimation algorithm does not restrict the applicability of the proposed model and estimation algorithm for low or high spiking rates in principle as demonstrated in the examples. Due to the simple computational structure of the proposed algorithm, it can be used for analyzing long term recordings of over a month as in our case. Here in this study, we propose for the first time the idea of functional spike-timing-dependent plasticity (STDP), which links behavioral learning with functional synaptic efficacy between neurons. Specifically, our results show that as learning progresses when rats performed the learning control task, the functionally excitatory synaptic connections become more inhibitory.

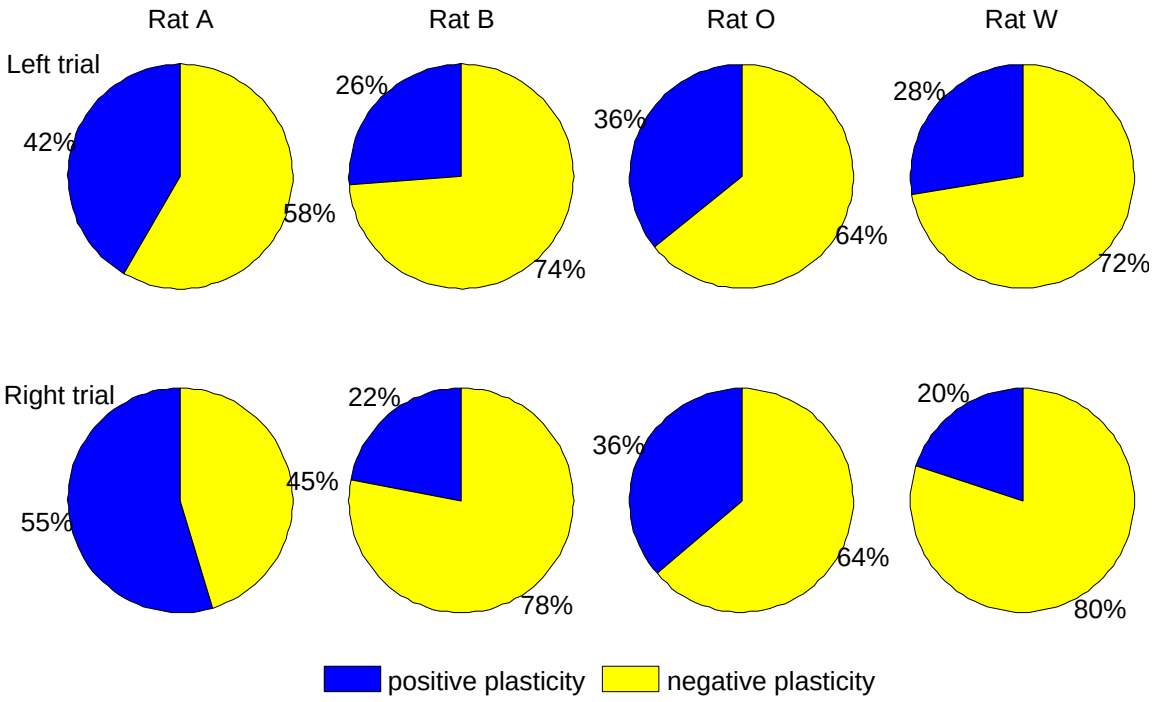


Figure 6.8: For rat B, O, W and A's left trial, negative plasticity was the majority. However, as an exception that rat A never learned right trial, positive plasticity was majority for A's right trials.

REFERENCES

[1] T. Riley, M. Bernhardt, C. Cowell, D. Hickman, and M. Smith, "Implementing advanced image processing technology in sensor systems for security and surveillance," vol. 6741, p. 67410W, SPIE, 2007.

[2] M. Heikkilä and M. Pietikäinen, "An image mosaicing module for wide-area surveillance," in *VSSN '05: Proceedings of the third ACM international workshop on Video surveillance & sensor networks*, (New York, NY, USA), pp. 11–18, ACM, 2005.

[3] R. Bindschadler, P. Vornberger, A. Fleming, A. Fox, J. Mullins, D. Binnie, S. J. Paulsen, B. Granneman, and D. Gorodetzky, "The landsat image mosaic of antarctica," *Remote Sensing of Environment*, vol. 112, no. 12, pp. 4214 – 4226, 2008.

[4] A. Rango, A. Laliberte, J. E. Herrick, C. Winters, K. Havstad, C. Steele, and D. Browning, "Unmanned aerial vehicle-based remote sensing for rangeland assessment, monitoring, and management," *Journal of Applied Remote Sensing*, vol. 3, p. 033542, 2009.

[5] S. R. Herwitz, L. F. Johnson, S. E. Dunagan, R. G. Higgins, D. V. Sullivan, J. Zheng, B. M. Lobitz, J. G. Leung, B. A. Gallmeyer, M. Aoyagi, R. E. Slye, and J. A. Brass, "Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support," *Computers and Electronics in Agriculture*, vol. 44, no. 1, pp. 49 – 61, 2004.

[6] R. Szeliski and R. Szeliski, "Image mosaicing for tele-reality applications," pp. 44–53, 1994.

[7] K. Loewke, D. Camarillo, K. Salisbury, and S. Thrun, "Deformable image mosaicing for optical biopsy," *Computer Vision, IEEE International Conference on*, vol. 0, pp. 1–8, 2007.

[8] N. Gracias, S. van der Zwaan, A. Bernardino, R. Bernardino, and J. Santos-Victor, "Mosaic based navigation for autonomous underwater vehicles," *Journal of Oceanic Engineering*, 2003.

[9] S. Lu, B. M. Chen, and C. C. Ko, "Perspective rectification of document images using fuzzy set and morphological operations," *Image and Vision Computing*, vol. 23, pp. 541–553, 2005.

[10] A. Rav-Acha, G. Engel, and S. Peleg, "Minimal aspect distortion (mad) mosaicing of long scenes," *Int. J. Comput. Vision*, vol. 78, no. 2-3, pp. 187–206, 2008.

[11] A. Roman, G. Garg, and M. Levoy, "Interactive design of multi-perspective images for visualizing urban landscapes," in *VIS '04: Proceedings of the conference on Visualization '04*, (Washington, DC, USA), pp. 537–544, IEEE Computer Society, 2004.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[13] S. Mann and R. W. Picard, "Video orbits of the projective group: A simple approach to featureless estimation of parameters," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 6, pp. 1281–1295, 1997.

[14] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[15] T. Ko, "A survey on behavior analysis in video surveillance for homeland security applications," in *Applied Imagery Pattern Recognition Workshop, 2008. AIPR '08. 37th IEEE*, pp. 1 –8, oct. 2008.

[16] J. L. Crowley and K. Schwerdt, "Robust tracking and compression for video communication," in *Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, RATFG-RTS '99, (Washington, DC, USA), pp. 2–, IEEE Computer Society, 1999.

[17] I. D. Peikon, N. A. Fitzsimmons, M. A. Lebedev, and M. A. Nicolelis, "Three-dimensional, automated, real-time video system for tracking limb motion in brain-machine interface studies," *Journal of Neuroscience Methods*, vol. 180, no. 2, pp. 224 – 233, 2009.

[18] Q. Xu, R. J. Hamilton, R. A. Schowengerdt, B. Alexander, and S. B. Jiang, "Lung tumor tracking in fluoroscopic video based on optical flow," *Medical Physics*, vol. 35, no. 12, pp. 5351–5359, 2008.

[19] B. Gloyer, H. Aghajan, K.-Y. Siu, and T. Kailath, "Vehicle detection and tracking for freeway traffic monitoring," in *Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on*, vol. 2, pp. 970 –974 vol.2, oct-2 nov 1994.

[20] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Computer Vision and Pattern Recognition, 2004.*

CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, vol. 2, pp. II–302 – II–309 Vol.2, june-2 july 2004.

[21] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *Proceedings of the 6th European Conference on Computer Vision-Part II*, ECCV '00, (London, UK), pp. 751–767, Springer-Verlag, 2000.

[22] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Comparative study of background subtraction algorithms," *Journal of Electronic Imaging*, vol. 19, no. 3, p. 033003, 2010.

[23] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial Intelligence*, vol. 78, no. 1-2, pp. 87 – 119, 1995. Special Volume on Computer Vision.

[24] S. Araki, T. Matsuoka, H. Takemura, and N. Yokoya, "Real-time tracking of multiple moving objects in moving camera image sequences using robust statistics," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 2, pp. 1433 –1435 vol.2, Aug. 1998.

[25] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping.," in *ECCV (4)'04*, pp. 25–36, 2004.

[26] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," 2000.

[27] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *Int. J. Comput. Vision*, vol. 56, pp. 221–255, February 2004.

[28] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593 – 600, 1994.

[29] H. Lin, J. Si, and G. P. Abousleman, "Fast and robust image mosaicking for monocular video," vol. 5809, pp. 443–452, SPIE, 2005.

[30] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. New York, NY, USA: John Wiley & Sons, Inc., 1987.

[31] C. Yang, H. Mao, G. Abousleman, and J. Si, "Correction of projective distortion in long-image-sequence mosaics without prior information," vol. 7668, p. 76680Q, SPIE, 2010.

[32] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Cambridge, MA: O'Reilly, 2008.

[33] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in C: the art of scientific computing*. New York, NY, USA: Cambridge University Press, 1988.

[34] H. Mao, C. Yang, G. P. Abousleman, and J. Si, "Automated multiple target detection and tracking in uav videos," vol. 7668, p. 76680J, SPIE, 2010.

[35] P. G. Musial, S. N. Baker, G. L. Gerstein, E. A. King, and J. G. Keating, "Signal-to-noise ratio improvement in multiple electrode recording," *Journal of Neuroscience Methods*, vol. 115, no. 1, pp. 29 – 43, 2002.

[36] A. K. Kreiter, A. M. Aertsen, and G. L. Gerstein, "A low-cost single-board solution for real-time, unsupervised waveform classification of multineuron recordings," *Journal of Neuroscience Methods*, vol. 30, no. 1, pp. 59 – 69, 1989.

[37] F. Wood, M. Black, C. Vargas-Irwin, M. Fellows, and J. Donoghue, "On the variability of manual spike sorting," *Biomedical Engineering, IEEE Transactions on*, vol. 51, pp. 912–918, June 2004.

[38] M. J. Song and H. Wang, "A spike sorting framework using nonparametric detection and incremental clustering," *Neurocomputing*, vol. 69, no. 10-12, pp. 1380 – 1384, 2006. Computational Neuroscience: Trends in Research 2006.

[39] N. Dedual, M. Ozturk, J. Sanchez, and J. Principe, "An associative memory readout in esn for neural action potential detection," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pp. 2295 –2299, aug. 2007.

[40] P. H. Thakur, H. Lu, S. S. Hsiao, and K. O. Johnson, "Automated optimal detection and classification of neural action potentials in extra-cellular recordings," *Journal of Neuroscience Methods*, vol. 162, no. 1-2, pp. 364 – 376, 2007.

[41] M. S. Fee, P. P. Mitra, and D. Kleinfeld, "Variability of extracellular spike waveforms of cortical neurons," *Journal of Neurophysiology*, vol. 76, no. 6, pp. 3823–3833, 1996.

[42] X. Yang and S. Shamma, "A totally automated system for the detection and classification of neural spikes," *Biomedical Engineering, IEEE Transactions on*, vol. 35, no. 10, pp. 806–816, 1988.

[43] K. H. Kim and S. J. Kim, "A wavelet-based method for action potential detection from extracellular neural signal recording with low signal-to-noise ratio," *Biomedical Engineering, IEEE Transactions on*, vol. 50, pp. 999 –1011, aug. 2003.

[44] E. Hulata, R. Segev, Y. Shapira, M. Benveniste, and E. Ben-Jacob, "Detection and sorting of neural spikes using wavelet packets," *Phys. Rev. Lett.*, vol. 85, pp. 4637–4640, Nov 2000.

[45] E. Hulata, R. Segev, and E. Ben-Jacob, "A method for spike sorting and detection based on wavelet packets and shannon's mutual information," *Journal of Neuroscience Methods*, vol. 117, no. 1, pp. 1 – 12, 2002.

[46] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[47] K. Oweiss and D. Anderson, "A multiresolution generalized maximum likelihood approach for the detection of unknown transient multichannel signals in colored noise with unknown covariance," *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, vol. 3, pp. III–2993–III–2996 vol.3, 2002.

[48] K. Oweiss and D. Anderson, "A unified framework for advancing array signal processing technology of multichannel microprobe neural recording devices," in *Microtechnologies in Medicine Biology 2nd Annual International IEEE-EMB Special Topic Conference on*, pp. 245 –250, 2002.

[49] Z. Nenadic and J. Burdick, "Spike detection using the continuous wavelet trans-

form," *Biomedical Engineering, IEEE Transactions on*, vol. 52, pp. 74 –87, jan. 2005.

[50] R. Benitez and Z. Nenadic, "Robust unsupervised detection of action potentials with probabilistic models," *Biomedical Engineering, IEEE Transactions on*, vol. 55, pp. 1344 –1354, april 2008.

[51] S. Santaniello, G. Fiengo, L. Glielmo, and G. Catapano, "A biophysically inspired microelectrode recording-based model for the subthalamic nucleus activity in parkinson's disease," *Biomedical Signal Processing and Control*, vol. 3, no. 3, pp. 203 – 211, 2008.

[52] Y. Xu, J. Weaver, D. Healy, and J. Lu, "Wavelet transform domain filters: a spatially selective noise filtration technique," *Image Processing, IEEE Transactions on*, vol. 3, no. 6, pp. 747–758, 1994.

[53] S. Mallat, *A wavelet tour of signal processing*. San Diego: Academic Press, 1989.

[54] D. Humphrey and E. Schmidt, *Extracellular Single-Unit Recording Methods*. New York: Humana Press, 1991.

[55] G. Wu, R. G. Hallin, and R. Ekedahl, "Multiple action potential waveforms of single units in man as signs of variability in conductivity of their myelinated fibres," *Brain Research*, vol. 742, no. 1-2, pp. 225 – 238, 1996.

[56] B. Sadler and A. Swami, "Analysis of multiscale products for step detection and estimation," *Information Theory, IEEE Transactions on*, vol. 45, pp. 1043 –1051, apr 1999.

[57] P. Bao and L. Zhang, "Noise reduction for magnetic resonance images via adaptive multiscale products thresholding," *Medical Imaging, IEEE Transactions on*, vol. 22, pp. 1089 –1099, sept. 2003.

[58] L. Smith, "Noisy spike generator, matlab software." Retrieved August 24, 2008, from University of Stirling, Department of Computing Science and Mathematics Web site: `http://www.cs.stir.ac.uk/~lss/noisyspikes/`, 2006.

[59] Z. Nenadic, "Spike detection with the continuous wavelet transform, matlab software." University of California, Irvine , Center for BioMedical Signal Processing and Computation. Web site: `http://cbmspc.eng.uci.edu`, 2005.

[60] B. Olson, J. Si, J. Hu, and J. He, "Closed-loop cortical control of direction using support vector machines," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 13, no. 1, pp. 72 –80, 2005.

[61] B. Naundorf, F. Wolf, and M. Volgushev, "Unique features of action potential initiation in cortical neurons," *Nature*, vol. 440, no. 7087, pp. 1060–1063, 2006.

[62] M. Volgushev, A. Malyshev, P. Balaban, M. Chistiakova, S. Volgushev, and F. Wolf, "Onset dynamics of action potentials in rat neocortical neurons and identified snail neurons: Quantification of the difference," *PLoS ONE*, vol. 3, no. 4, p. e1962, 2008.

[63] Z. Nenadic and J. Burdick, "A control algorithm for autonomous optimization of extracellular recordings," *Biomedical Engineering, IEEE Transactions on*, vol. 53, pp. 941 –955, May 2006.

[64] S. Prescott and T. Sejnowski, "Spike-rate coding and spike-time coding are affected oppositely by different adaptation mechanisms," *The Journal of Neuroscience*, vol. 28, no. 50, pp. 13649–13661, 2008.

[65] E. Brown, R. Kass, and P. Mitra, "Multiple neural spike train data analysis: state-of-the-art and future challenges," *Nat Neurosci*, vol. 7, pp. 456–461, May 2004. 10.1038/nn1228.

[66] M. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *NETWORK-COMPUTATION IN NEURAL SYSTEMS*, vol. 9, pp. R53–R78, November 1998.

[67] C. Pouzat, O. Mazor, and G. Laurent, "Using noise signature to optimize spike-sorting and to assess neuronal classification quality," *Journal of Neuroscience Methods*, vol. 122, no. 1, pp. 43–57, 2002.

[68] M. Salganicoff, M. Sarna, L. Sax, and G. Gerstein, "Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. i. algorithms and implementation," *Journal of Neuroscience Methods*, vol. 25, no. 3, pp. 181–187, 1988.

[69] S. Takahashi, Y. Sakurai, M. Tsukada, and Y. Anzai, "Classification of neuronal activities from tetrode recordings using independent component analysis," *Neurocomputing*, vol. 49, no. 1–4, pp. 289–298, 2002.

[70] S. Takahashi, Y. Anzai, and Y. Sakurai, "Automatic sorting for multi-neuronal activity recorded with tetrodes in the presence of overlapping spikes," *Journal of Neurophysiology*, vol. 89, no. 4, pp. 2245–2258, 2003.

[71] S. Takahashi, Y. Anzai, and Y. Sakurai, "A new approach to spike sorting for multi-neuronal activities recorded with a tetrodehow ica can be practical," *Neuroscience Research*, vol. 46, no. 3, pp. 265–272, 2003.

[72] E. Salinas and T. Sejnowski, "Correlated neuronal activity and the flow of neural information.," *Nature Reviews Neuroscience*, vol. 2, no. 8, pp. 539–550, 2001.

[73] A. Kreiter, A. Aertsen, and G. Gerstein, "A low-cost single-board solution for real-time, unsupervised waveform classification of multineuron recordings," *Journal of Neuroscience Methods*, vol. 30, no. 1, pp. 59–69, 1989.

[74] J. Letelier and P. Weber, "Spike sorting based on discrete wavelet transform coefficients," *Journal of Neuroscience Methods*, vol. 101, no. 2, pp. 93–106, 2000.

[75] E. Hulata, R. Segev, and E. Ben-Jacob, "A method for spike sorting and detection based on wavelet packets and shannon's mutual information," *Journal of Neuroscience Methods*, vol. 117, no. 1, pp. 1–12, 2002.

[76] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering.," *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[77] R. Quian Quiroga, A. Kraskov, T. Kreuz, and P. Grassberger, "Performance of different synchronization measures in real data: A case study on electroencephalographic signals," *Phys. Rev. E*, vol. 65, p. 041903, March 2002.

[78] R. Vogelstein, K. Murari, P. Thakur, C. Diehl, S. Chakrabartty, and G. Cauwenberghs, "Spike sorting with support vector machines," in *Engineering in*

Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE, vol. 1, pp. 546–549, September 2004.

[79] J. Stitt, R. Gaumond, J. Frazier, and F. Hanson, "A comparison of neural spike classification techniques [caterpillar taste organs application]," in *Engineering in Medicine and Biology Society, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, vol. 3, pp. 1092–1094 vol.3, October 1997.

[80] H. Jung, J. Choi, and T. Kim, "Solving alignment problems in neural spike sorting using frequency domain pca," *Neurocomputing*, vol. 69, no. 79, pp. 975 – 978, 2006. ¡ce:title¿New Issues in Neurocomputing: 13th European Symposium on Artificial Neural Networks¡/ce:title¿ ¡xocs:full-name¿13th European Symposium on Artificial Neural Networks 2005¡/xocs:full-name¿.

[81] Z. Yang, Q. Zhao, and W. Liu, "Improving spike separation using waveform derivatives," *Journal of Neural Engineering*, vol. 6, no. 4, p. 046006, 2009.

[82] G. Holt and C. Koch, "Electrical interactions via the extracellular potential near cell bodies," *Journal of Computational Neuroscience*, vol. 6, pp. 169–184, 1999. 10.1023/A:1008832702585.

[83] Z. Yang, Q. Zhao, and W. Liu, "Energy based evolving mean shift algorithm for neural spike classification," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 966–969, September 2009.

[84] M. Blatt, S. Wiseman, and E. Domany, "Superparamagnetic clustering of data," *Phys. Rev. Lett.*, vol. 76, pp. 3251–3254, April 1996.

[85] P. Inc., *Plexon Offline Sorter v2.8 manual*, 2006.

[86] J. Kretzberg, T. Coors, and J. Furche, "Comparison of valley seeking and t-distributed em algorithm for spike sorting," *BMC Neuroscience*, vol. 10, no. Suppl 1, p. P47, 2009.

[87] L. Smith and N. Mtetwa, "Manual for the noisy spike generator matlab software, at http://www.cs.stir.ac.uk/65lss/noisyspikes," 2006.

[88] D. Brugger, M. Bogdan, and W. Rosenstiel, "Automatic cluster detection in kohonen's som," *Neural Networks, IEEE Transactions on*, vol. 19, pp. 442–459, March 2008.

[89] G. qiang Bi and M. ming Poo, "Synaptic modification by correlated activity: Hebb's postulate revisited," *Annual Review of Neuroscience*, vol. 24, no. 1, pp. 139–166, 2001.

[90] M. Okatan, M. A. Wilson, and E. N. Brown, "Analyzing functional connectivity using a network likelihood model of ensemble neural spiking activity," *Neural Computation*, vol. 17, no. 9, pp. 1927–1961, 2005.

[91] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps," *Science*, vol. 275, no. 5297, pp. 213–215, 1997.

[92] E. Salinas and T. J. Sejnowski, "Correlated neuronal activity and the flow of neural information," *Nature Reviews Neuroscience*, vol. 2, pp. 539–550, 2001.

[93] C. J. N. G. D. A. Stam, "Phase lag index: Assessment of functional connectivity from multi channel eeg and meg with diminished bias from common sources," *Human Brain Mapping*, vol. 28, pp. 1178–1193, 2007.

[94] K. Narayanan, D. Weber, J. He, A. Prasad, and L. Iasemidis, "Analysis of neuronal interactions during adaptation and learning in motor control of primates," in *Engineering in Medicine and Biology, 2002. 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society EMBS/BMES Conference, 2002. Proceedings of the Second Joint*, vol. 3, pp. 2552–2553 vol.3, 2002.

[95] C. W. J. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica*, vol. 37, pp. 424–438, Aug 1969. ArticleType: research-article / Full publication date: Aug., 1969 / Copyright Â©1969 The Econometric Society.

[96] M. Krumin and S. Shoham, "Multivariate autoregressive modeling and granger causality analysis of multiple spike trains," *Intell. Neuroscience*, vol. 2010, pp. 10:6–10:6, January 2010.

[97] L. P. S. A. F. K. E. S. Chornoboy, "Maximum likelihood identification of neural point process systems," *Biological Cybernetics*, vol. 59, no. 4, pp. 265–275, 1988.

[98] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Berlin, Germany: Cambridge University Press, 2002.

[99] M. Gilson, A. Burkitt, and L. J. V. Hemmen, "Stdp in recurrent neuronal networks," *Frontiers in Computational Neuroscience*, vol. 4, no. 23, 2010.

[100] W. Gerstner, "Time structure of the activity in neural network models," *Phys. Rev. E*, vol. 51, pp. 738–758, Jan 1995.

[101] E. N. Brown, R. Barbieri, V. Ventura, R. E. Kass, and L. M. Frank, "The time-rescaling theorem and its application to neural spike train data analysis," *Neural Comput.*, vol. 14, no. 2, pp. 325–346, 2002.

[102] S. Haykin, *Neural Networks: A Comprehensive Foundation*. NY: Prentice Hall, second ed., 1998.

[103] I. Stevenson, J. Rebesco, N. Hatsopoulos, Z. Haga, L. Miller, and K. Kording, "Bayesian inference of functional connectivity and network structure from spikes," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 17, pp. 203–213, june 2009.

[104] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects," *Journal of Neurophysiology*, vol. 93, no. 2, pp. 1074–1089, 2005.

[105] G. E. P. Box, W. G. Hunter, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. Hoboken, NJ: John Wiley & Sons, 1978.

[106] M. Ding, Y. Chen, and S. L. Bressler, "Granger Causality: Basic Theory and Application to Neuroscience," Aug 2006.

[107] C. Yang, O. Byron, and J. Si, "A multiscale correlation of wavelet coefficients approach to spike detection," *Neural Computation*, vol. 23, pp. 215–250, Octo-

ber 2010. doi: 10.1162/NECO_a_00063.

[108] C. Yang, Y. Yuan, and J. Si, "High performance spike detection and sorting using neural waveform phase information and som clustering," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pp. 1 –7, July 2010.