5. Spike Sorting

By: Clinton Zheng Da Goh

5.1 Introduction

Recognising the response of stimulated neurons is essential for neurophysiological studies. Most neurons respond to external stimuli by firing action potentials which serve as a means of communication with other neurons. Several devices have been developed for the purpose of recording such activity. These include the stereotrode (McNaughton 1983), tetrode (Reece 1989), and the MEA, or multi-electrode array (Thomas 1972).

Analysis of MEA recordings is a technically challenging process. Besides picking up background noise, electrodes also record spikes from more than one neuron. The goal of spike sorting is to assign detected spikes to specific neurons, and its reliability depends on the ability to detect and classify spikes. Many spike sorting approaches have been developed, but all algorithms are made up of three stages: (1) detecting spikes, (2) extracting spike features and (3) clustering of spikes features.

In this chapter, we will discuss various spike sorting methods and present a MATLAB based Spike Sorting Toolbox. In particular, we have developed an algorithm which builds on the template matching approach (Zhang 2003, Segev 2004) by exploiting calibration capabilities of our system. Calibration can be done by stimulating each neuron in turn and capturing spike patterns across all electrodes. Since extracellular action potentials decay exponentially with distance (Segev 2004, Escola 2008) and because each neuron occupies a specific position, spike patterns are unique to calibrated neurons and can be used to classify detected spikes.

One of the main challenges of template matching algorithms is the high computational load involved in comparing templates to detected spikes. We tackle this problem by extracting only the most prominent features of recorded spike patterns: the peak amplitude and relative phase lags. Using these two features, we reduce the dimensions of spike templates to an $M\times 2$ matrix of eigenvectors which we term “eigenmatrix templates”. These templates are then used in an iterative template matching process which assigns detected spikes to calibrated neurons. The spatiotemporal information obtained from spike sorting can be used in various applications such as implementing a closed-loop stimulation feedback system (Potter 2006) and transferring an image onto a neural network. Finally, we characterise the Spike Sorting Toolbox using artificial data (Quiroga 2004) generated by an MEA based test platform.

5.2 Background

5.2.1 The Basic Problems in Spike Sorting

The challenges of spike sorting are illustrated in the extracellular recording shown in Figure 5.1. Upon observation, one will notice that the recording contains many different types of action potentials. To a neurophysiologist, he or she may be interested to find out how these spikes are related to different neurons, and how this relationship may be determined.

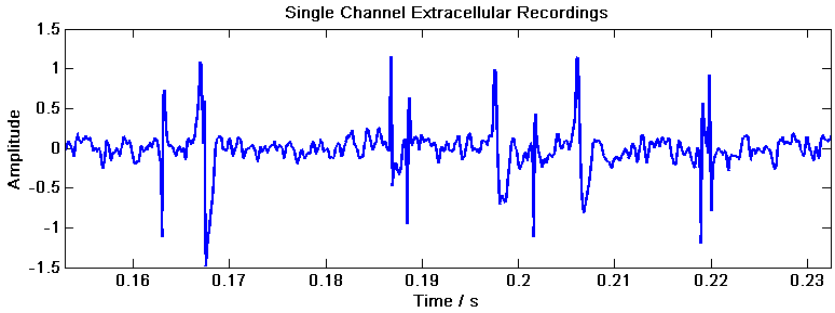


Figure 5.1: Single channel extracellular recordings obtained from Quiroga et. al. 2004.

One of the basic questions asked in spike sorting is how can we identify and classify a spike? To answer this, we must first understand features of action potentials. One of the most prominent features is the peak amplitude. When the cell membrane of a neuron is depolarised past its threshold voltage, voltage-gated sodium channels open, resulting in an influx of sodium ions and a spike. During extracellular recordings, this spike is recorded as a negative peak, since the recording medium is negative with respect to the cell potential. Once the peak amplitude is reached, sodium channels close and potassium channels open, resulting in repolarisation. Repolarisation often occurs past the cell's resting potential, and is seen as a positive peak in extracellular recordings. Figure 5.2 illustrates spike features which allow us to recognise action potentials in the presence of background noise. In this section, we will look at various methods used to tackle the spike sorting problem.

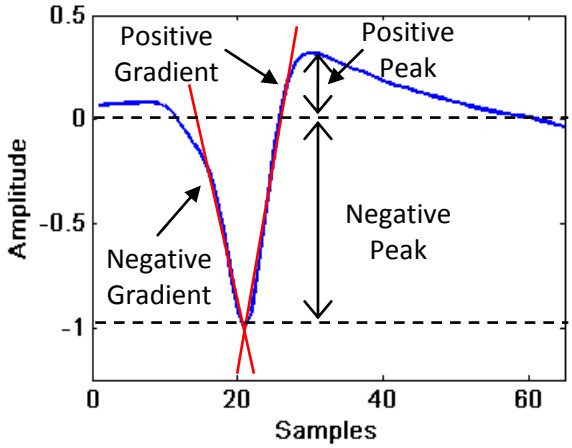


Figure 5.2: Measurement of spike features. Useful spike features which characterise a spike shape include positive and negative gradients, spike width, positive and negative peaks, and peak-to-peak amplitude.

5.2.2 Spike Detection

One of aims of spike detection is to identify data points which form an action potential. Current spike detection methods make use of prominent features such as the peak amplitude, to enable automatic detection of spikes.

Voltage Threshold Detection

The voltage threshold method detects data points which lie between a minimum and maximum threshold value. Many spike sorting algorithms use this method for detecting spikes as it is easy to implement and discards background noise effectively (Lewicki 1998, Zhang 2003, Quiroga 2004, Segev 2004). The minimum threshold detects high amplitude spikes, while the maximum threshold discards large amplitude artefacts (Figure 5.3).

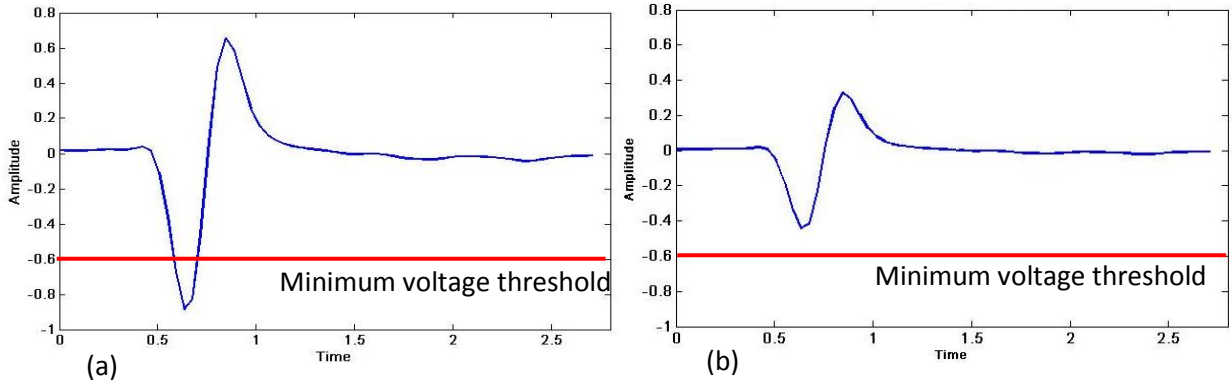


Figure 5.3: Spike detection using the voltage threshold method. (a) Detected spike with amplitude exceeding minimum voltage threshold. (b) Spike amplitude is too small for detection

In order to discard background noise and detect action potentials only, the voltage threshold must be large enough so as to avoid the detection of background noise. If the spiking neuron is located far away from the electrode, low amplitude spikes may not be detected by the voltage threshold method. One solution is to use multiple electrodes. Each time a neuron spikes, recorded waveforms are visualised across all channels of the MEA (Figure 5.4). This ensures that information is not lost even though spikes are not detected on some electrodes.

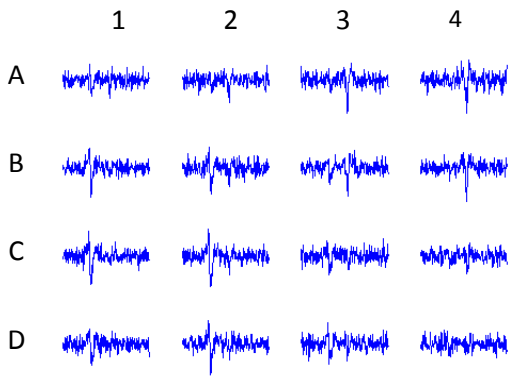


Figure 5.4: A 4×4 MEA recording. The use of multiple electrodes increases spatial resolution of signals and prevents information loss from neurons located far away from electrodes.

5.2.3 Spike Feature Extraction

Neurons typically produce spikes with a characteristic shape (Lewicki 1998). If we assume that a neuron's spike shape is time invariant, we can characterise its shape by extracting spike features. However, this assumption is invalid when spike shapes change during recording. Overlapping spikes may also be recorded when two or more action potentials reach an electrode at the same time. Various spike feature extraction methods deal with these problems and will be discussed in this section.

Feature Analysis

A simple method of analysing spike shapes is through the direct measurement of spike features such as the peak-to-peak voltage amplitude, spike width, and spike gradient (Figure 5.5).

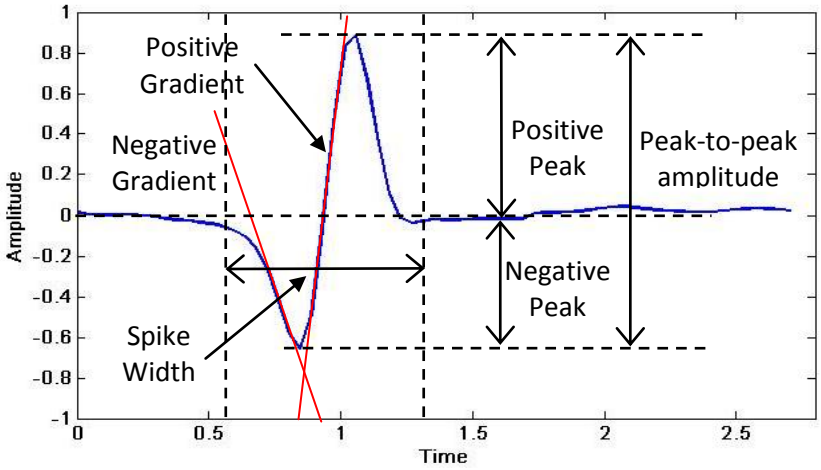


Figure 5.5: Measurement of spike features. Useful spike features which characterise a spike shape include positive and negative gradients, spike width, positive and negative peaks, and peak-to-peak amplitude.

Spike features can be used to discriminate between spikes from different neurons. This method may be used on a single electrode or across multiple electrodes. Figure 5.6 shows a scatter plot of peak amplitudes recorded on 2 neighbouring electrodes. There is a clear clustering of three different spike shapes indicating that the detected spikes originated from three different neurons.

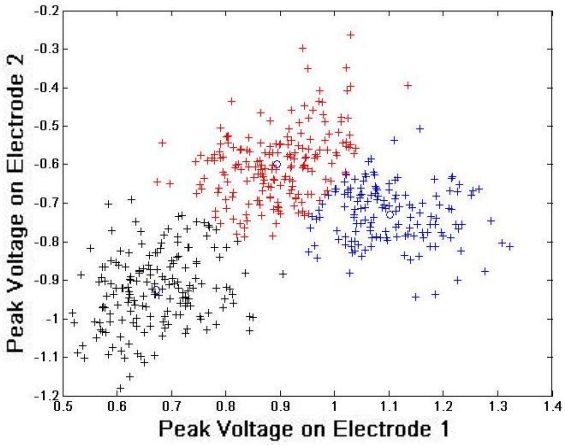


Figure 5.6: Feature analysis applied to two electrodes. 3 clusters are observed, corresponding to 3 different neurons.

Spike Template Construction

Since neurons produce action potentials with a characteristic shape (Lewicki 1998), a way of quantifying spike features is through the use of spike templates. Spike templates are unique action potentials generated by a neuron (Wheeler 1982). Figure 5.7 illustrates 5 unique waveforms (data from Quiroga 2004) which differ in spike features described previously.

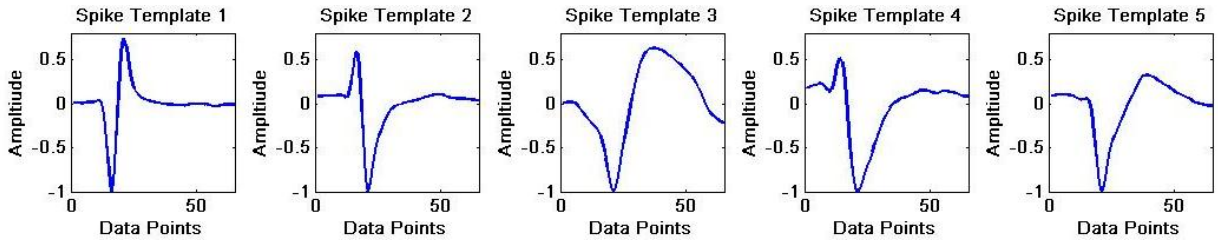


Figure 5.7: Five unique spike template waveforms, each generated by a different neuron.

Spike templates can be extended to MEA recordings. Each template is a set of recordings across all channels when a neuron is stimulated (Figure 5.8). Similarly, each template is unique since neurons occupy a specific location, and because action potentials decay exponentially with distance (Segev 2004, Escola 2008). A challenge of spike template construction is that it requires the stimulation of single neurons. This is possible with our system as each LED is able to target single neurons (Nikolic 2007, Poher 2008). Another challenge is that the transmission of action potentials via neuronal processes may result in recorded spikes from other neurons when a particular neuron is stimulated. One way to isolate clean waveforms is to stimulate each neuron multiple times and discard waveforms which differ from the average by a threshold (Segev 2004).

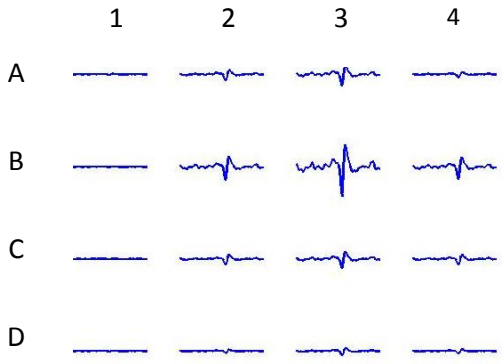


Figure 5.8: An MEA spike template. The spiking neuron is located closest to electrode B2 which shows the greatest spike amplitude relative those observed on other channels.

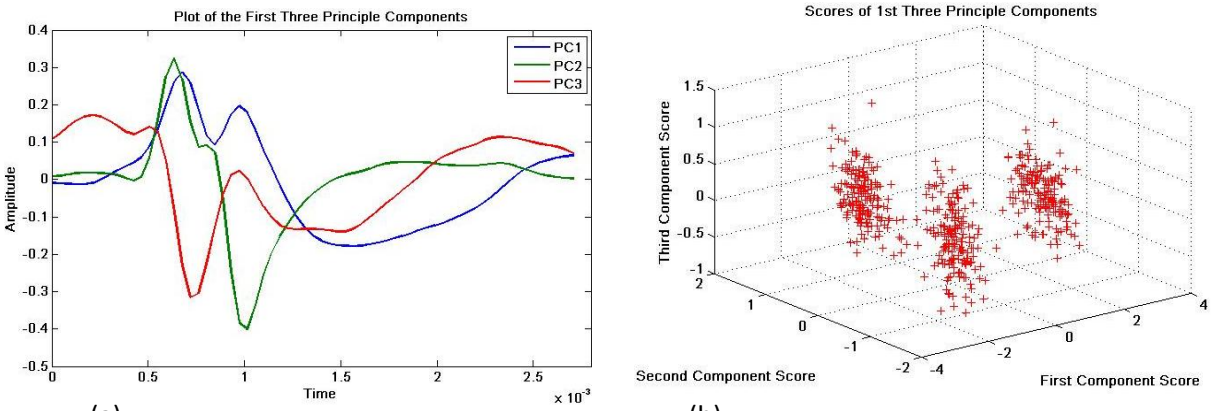
One assumption of spike template construction is that spike shapes are time invariant. This assumption does not always hold, as burst-firing neurons may generate spikes which vary in shape, and electrode drift will result in a gradual change in spike shapes (Lewicki 1998). This problem may be solved by constructing templates for a limited recording period. However, the process requires a higher level of supervision to maintain the accuracy of the template database.

Principle Components Analysis

Principle Components Analysis (PCA) is commonly used to automatically extract spike features for spike classification (Lewicki 1998, Zhang 2003). The aim of PCA is to compute an ordered set of orthogonal basis vectors which can be linearly combined to describe each detected spike. This method assumes that the largest variation in a set of data contains the dynamics of interest (Lewicki 1998, Shlens 2005). A brief mathematical description of PCA is found in Appendix N. Spike features are captured in the score for each principle component:

$$s_i = \sum_t c_i(t)x(t) \tag{5.1}$$

where $x(t)$ is the detected spike, and $c_i(t)$ is the i^{th} principle component. In practice, scores of the first three components are used to identify clusters as they account for approximately 76% of the variations in the data. Subsequent components may represent variability in the data due to background noise (Lewicki 1998). The results of PCA applied to artificial data (Quiroga 2004) are illustrated in Figure 5.9.



(a) (b)

Figure 5.9: Results of PCA applied on detected spikes. (a) The first three principle components. (b) A scatter plot of the scores from the first three components.

One challenge of PCA is that it does not resolve overlapping spikes as these spike shapes appear as outliers during the clustering process. PCA also fails to recognise varying spike shapes where waveforms are not stationary. When applied to MEA recordings, spikes from the same neuron which are recorded by multiple electrodes may exhibit different spike amplitudes. Using PCA on such data may result in the formation of two or more clusters, generating false positives. Hence, PCA alone remains only useful for spike sorting on a single channel.

Independent Components Analysis

Independent Components Analysis (ICA) is a method which is used to deal with multiple electrode signal recordings (Brown 2001, Takahashi 2003, Mamlouk 2005). ICA assumes that signal mixtures are a linear combination of source signals. ICA is similar to blind source separation: it recovers *n* independent signals that have been mixed into *n* channels by an unknown mixing process. ICA may be implemented via maximum likelihood estimation or entropy maximisation (Mitianoudis 2004). An efficient and widely used ICA algorithm is the FastICA which utilises non-Gaussianity maximisation (Hyvärinen 2000). The ICA concept is illustrated in Figure 5.10 and a brief mathematical description of ICA is found in Appendix O.

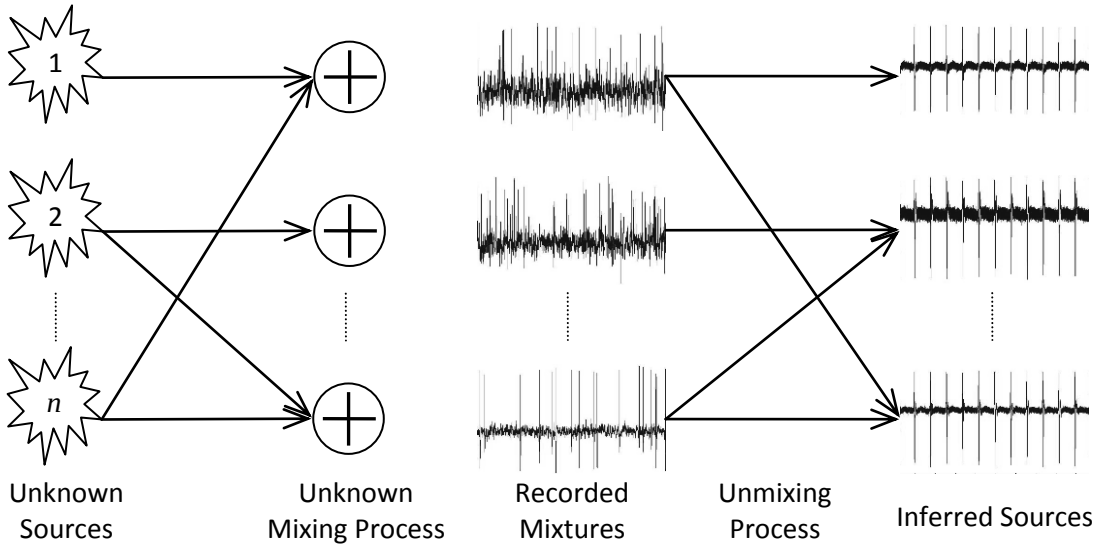


Figure 5.10: A model of Independent Components Analysis (adopted from Lewicki 1998). *n* unknown sources or spiking neurons are mixed linearly by an unknown mixing process and are recorded on *n* electrodes. The unmixing process is found through ICA which transforms recorded mixtures into independent signals.

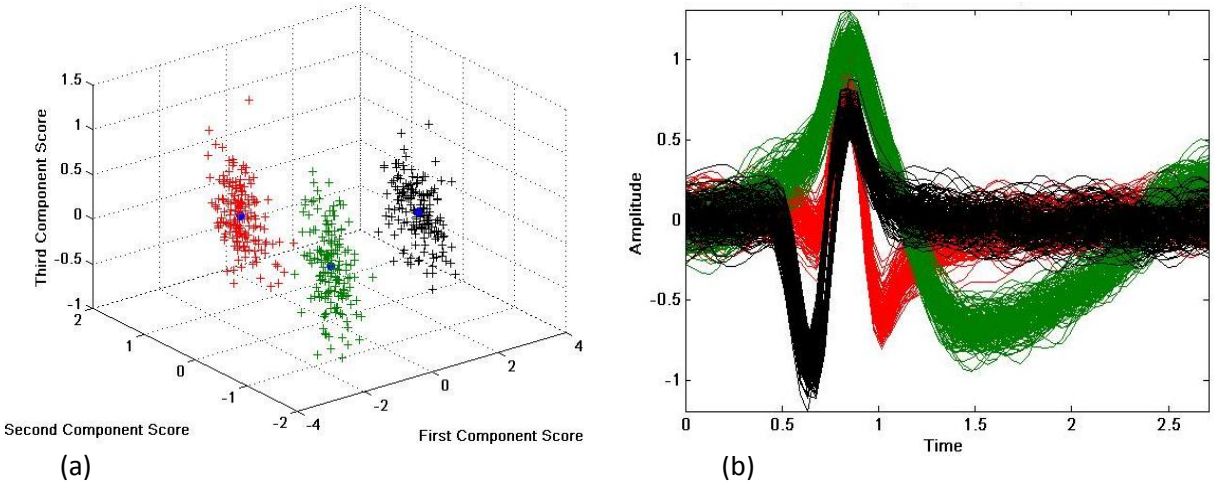
ICA solves the problem of overlapping spikes as it assumes that observations are linear mixtures of source signals. However, the number of neurons must be less than or equal to the number of electrodes. Although methods which deal with over-complete problems have been developed (Takahashi 2003, Mitanoudis 2004), few of them have been verified to be capable of accurately recovering original signals fully. Another assumption which ICA makes is the instantaneous mixture of source signals. This does not hold as depolarisation waves take time to propagate through the recording medium. Although powerful, ICA makes many assumptions which must be fully understood when applied to spike sorting.

5.2.4 Spike Feature Clustering

By plotting spike features, we are able to visually identify clusters which correspond to spikes from a particular neuron. However, to manually assign each spike to its cluster is a very time consuming process. In this section, we discuss algorithms which perform this task automatically with varying amounts of supervision.

K-means Clustering

K-means clustering classifies data into k number of clusters, based on features of the data set. Spike features include principle component scores and peak-to-peak amplitude ratio. After the user has specified the number of clusters, the algorithm uses an iterative process which groups features based on the minimum Euclidean distance from each data element to cluster centroids. The aim is to minimise the total variance within each cluster. K-means clustering applied to PCA (artificial data from Quiroga 2004) is illustrated in Figure 5.11.



(a) (b)

Figure 5.11: PCA and k-means clustering. (a) Scores of the first 3 principle components. K-means clustering was applied to obtain three clusters. (b) Clustered and aligned spikes from spike detection.

Since k-means clustering requires the user to specify the number of clusters, a poor choice will result in inaccuracies. By using the minimum Euclidean distance as the criteria for clustering, the distribution of data within the cluster is ignored. Hence, clusters whose shapes differ from a spherical distribution are not recognised (Lewicki 1998). In addition, omission of outliers has to be carried out manually as k-means clustering assumes that all data points belong to a cluster. Despite these drawbacks, this algorithm is easy to implement and is computationally efficient.

Template Matching

Spike template matching algorithms assume a pre-existing database of templates. The aim is to assign best fit templates to detected spikes, hence clustering spike features based on prior knowledge obtained from constructing spike templates. One way to match templates is through an iterative matching process (Segev 2004). Each template is aligned and subtracted from the recorded waveform. The resulting mean squared error (MSE) is used to measure the quality of fit, such that the template which produces the minimum mean squared error is assigned to the detected spike. The MSE is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(X_{ij} - Y_{ij}\right)^2 \tag{5.2}$$

where X_{ij} and Y_{ij} represents j^{th} sample of the detected spike and spike template recorded on the i^{th} electrode respectively. To resolve overlapping spikes, templates are iteratively subtracted from the resulting residual until the best matched template is a blank template. The iterative template matching process applied to MEA recordings in illustrated in Figure 5.12 (Segev 2004).

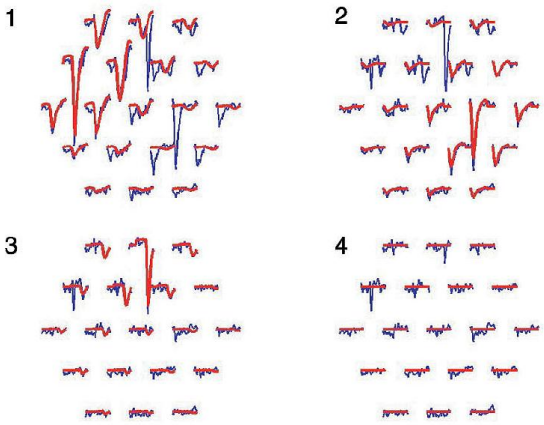


Figure 5.12: Iterative template matching carried out on a hexagonal MEA. After each match (labelled 1 to 4), the template is subtracted from the residue until the best match is a blank template (Segev 2004).

One pitfall of template matching is that it is limited to generated templates. If spike templates are not constructed for a class of neurons, then inaccuracies may be introduced during template matching. In addition, spike shapes may not be stationary, and may evolve with time. Solutions which compensate for time varying spikes require user supervision, which is an obstacle in implementing real time spike sorting. Spike template matching also has a high computational load when analysing MEA data. Users of the template matching approach have developed ways to limit the order of complexity by reducing the possible search space of templates (Lewicki 1998, Segev 2004). Despite these drawbacks, spike template matching is reliable if the user has prior knowledge of spiking patterns of each neuron. Since our system allows for neuron calibration, this serves as the main motivation for our choice in employing the template matching approach in spike sorting.

5.3 Methods

We present a modular Spike Sorting Toolbox based on the voltage threshold detection and a novel spike template matching approach. The user may use each module to perform a different stage in spike sorting, or the entire toolbox as a single module to automate the spike sorting process. We have also developed a test platform to simulate MEA based recordings, so as to validate the toolbox.

5.3.1 Spike Sorting Toolbox

The Spike Sorting Toolbox consists of 4 modules: Signal Filter, Spike Detector, Template Generator, Event Matrix Generator and Template Matcher which are illustrated in Figure 5.13.

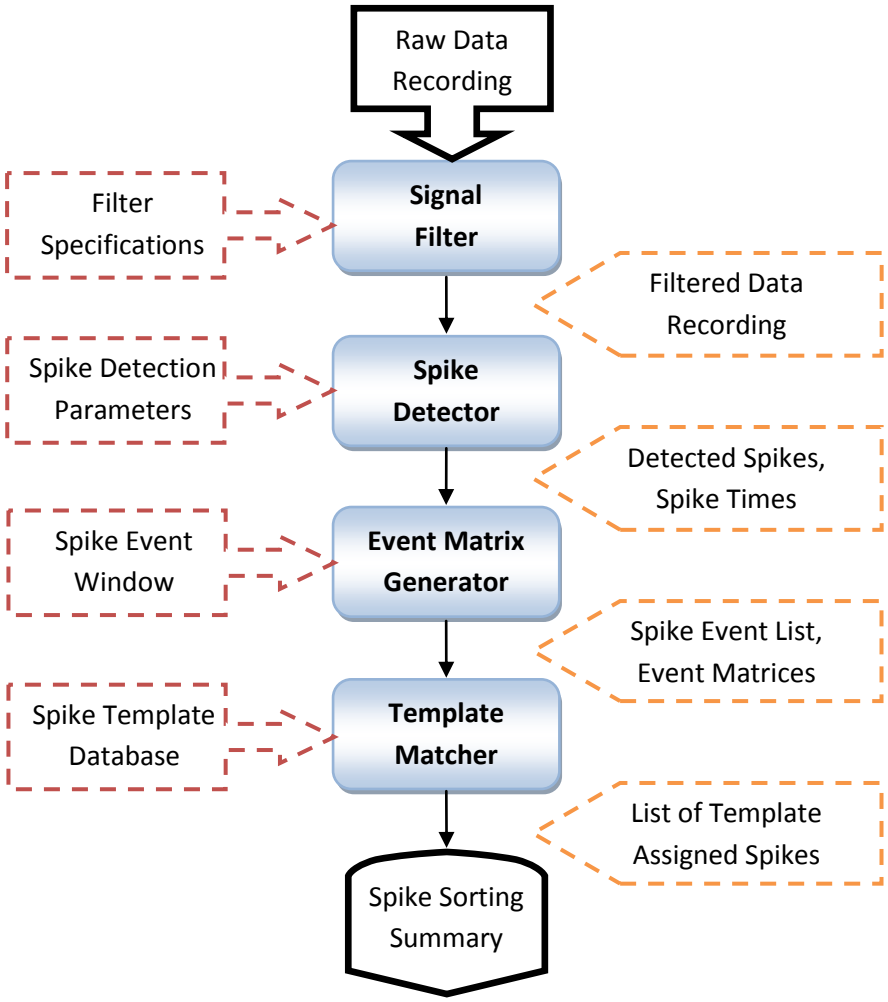


Figure 5.13: Data flow chart of the Spike Sorting Toolbox. Each module (blue box) performs a separate stage in the spike sorting process, based on user defined input parameters (red box). Modules are designed such that the output (orange box) of one module is used as an input for another.

Signal Filter

A digital IIR filter module was implemented in MATLAB for the purpose of filtering microelectrode recordings. This module is fully configurable by the user to implement the following bandpass IIR filters: Butterworth filter, Chebyshev Type II filter and Elliptic filter. The default digital filter was set as a 2nd order Elliptic filter, due to its steep transition band. Filter specifications were set to a passband ripple of 0.1dB, and a stopband attenuation of -40dB.

The filter module receives an input of MEA signal recordings and performs digital filtering channel wise, producing an output of filtered recordings. Filter taps are computed using the MATLAB Filter Design Toolbox functions `butter`, `cheby2` and `ellip`, while zero-phase filtering is performed using the Signal Processing Toolbox function `filtfilt`. The user manual and MATLAB code for the signal filter module can be found in Appendix P1. Figure 5.14 illustrates the effects of bandpass filtering a single channel recording for which the passband is specified to be in the range 300 Hz to 3 kHz.

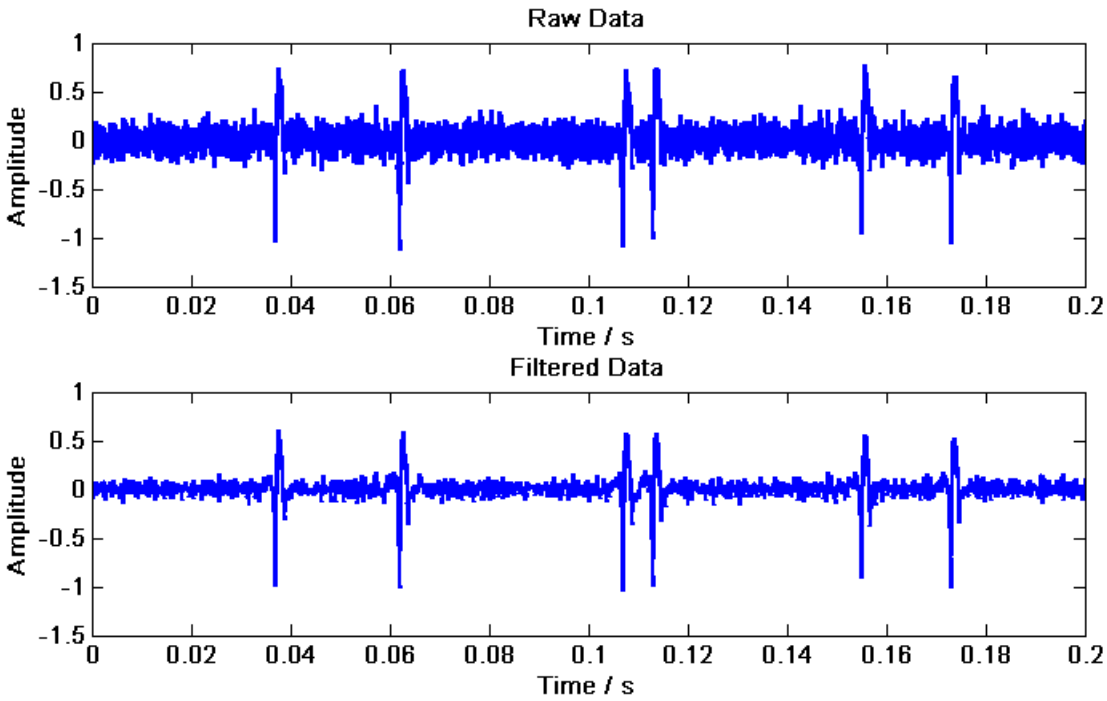


Figure 5.14: Effect of bandpass filtering (300 Hz to 3 kHz) a single channel signal recording. The 2nd order Elliptic filter effectively attenuates high frequency noise and preserves low frequency action potentials (data from Quiroga 2004).

Spike Detector

The spike detection module uses a voltage threshold to detect spikes. The threshold voltage is set by the user as a multiple of σ_n, the estimate of the standard deviation of background noise (Quiroga 2004). This estimate is robust to signals containing high frequency spikes, which may cause high threshold values if we simply consider the standard deviation of the signal (Quiroga 2004). To prevent multiple detection of the same spike, the user may specify a time window for which no spike may be detected after a threshold crossing occurs. We term this as the "detector dead time".

$$\sigma_n = median\left\{ x / 0.6745 \right\} \tag{5.3}$$

Spike detection may be performed on a set of original signal recordings or filtered signals from the signal filter module. The noise level of each channel is computed, and threshold voltages are determined based on user defined multiples applied to the noise level.

For each detected spike, the spike time and a snapshot of signal recordings across all electrodes is stored (Figure 5.15). However, the module does not recognise simultaneous detection of spikes across multiple electrodes. As a result, the output may contain multiple snapshots and spike times which correspond to a single spike event. The identification of unique spike events is performed by the spike event generator module, which is discussed in later sections. The user manual and MATLAB code for the spike detection module can be found in Appendix P2.

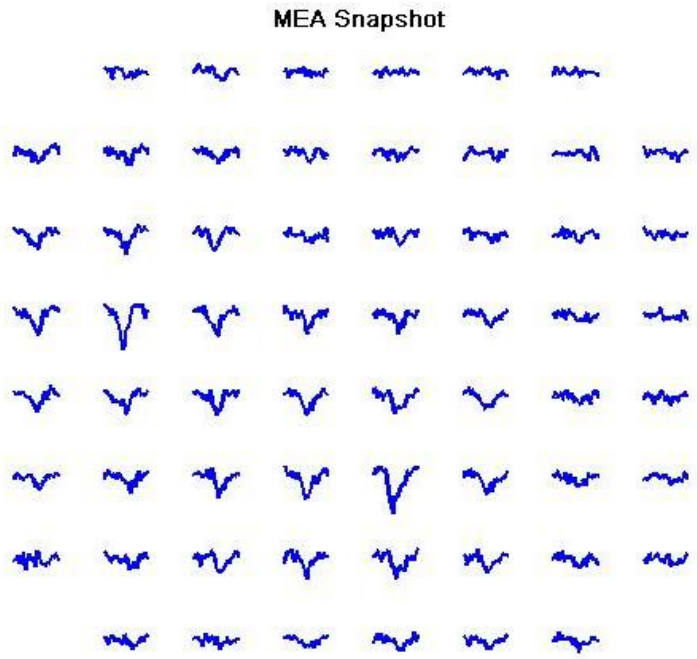


Figure 5.15: Recorded signals on all electrodes. The spike detection module takes a snapshot of activity recorded across all electrodes of the MEA whenever a spike is detected on any electrode.

Template Generator

The main function of the template generation module is to calibrate neurons of the system. This is done by storing typical spike patterns and generating eigenmatrix templates which are unique to each calibrated neuron. During system calibration, each neuron is stimulated in turn, producing sets of MEA signal recordings. Each MEA recording is filtered and spikes are detected using the signal filter module and spike detection module respectively. The module isolates a clean spike pattern by only considering electrodes which detect spikes in more than a certain percentage of the calibration input stimulus. This eliminates the possible detection of high amplitude noise and spikes from other neurons which were not stimulated.

To generate a set of eigenmatrix templates, the template generator first computes the mean spike recordings for electrodes which satisfy the above criteria. The peak amplitude and phase lags relative to the spike with the greatest peak amplitude are extracted and ordered within the template to preserve spatial information. Electrodes which do not detect a spike are set to have zero amplitude and phase. The database of eigenmatrix templates can then be used for template matching. Figure 5.16 illustrates this process of reducing template dimensions by extracting prominent features of spike patterns. The user manual and MATLAB code for the template generator module can be found in Appendix P3.

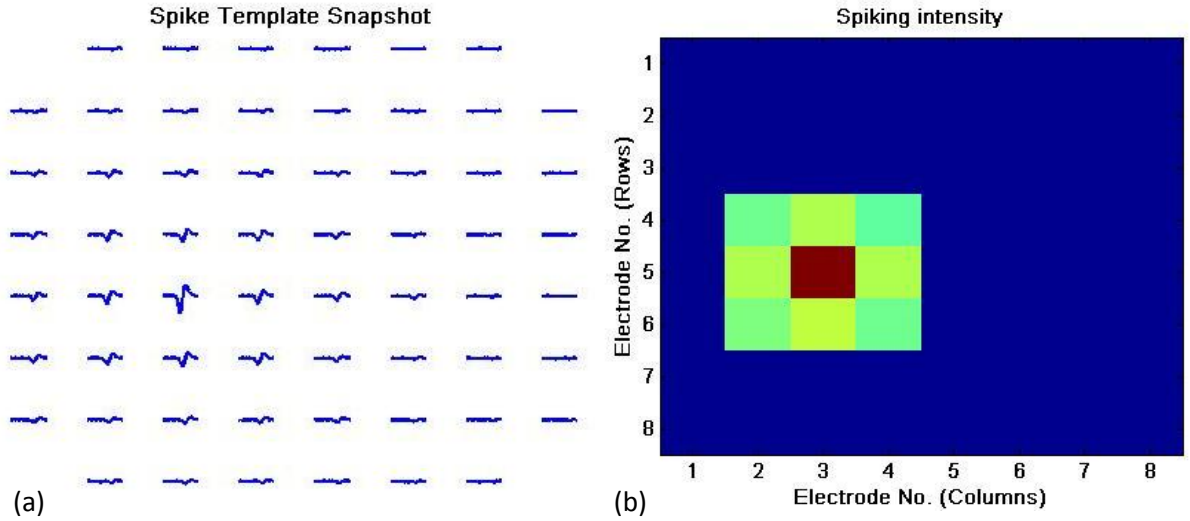


Figure 5.16: Reducing spike template dimensions through feature extraction. (a) A snapshot of spike patterns from a single spiking neuron. (b) An intensity plot of extraction of peak amplitudes. Spatial resolution is preserved by ordering values according to electrode layout.

Event Matrix Generator

The main function of the event matrix generator is to identify and generate a list of unique spike events. To illustrate this, consider a spike which is recorded on electrode F3 in Figure 5.17. Observe that neighbouring electrodes such as F2 and F4 also record the same spike event. Our aim is to identify the group of electrodes which record spikes of common origin in time.

During an action potential, depolarisation waves take time to propagate through the recording medium. This results in a phase lag between multiple electrode recordings of the same spike. To find out which electrodes detect a common spike, the user specifies a time window within which a spike must be detected in order for electrodes to be grouped under one event. Spike times falling outside this window are classified as a separate event. Figure 5.17 illustrates spikes appearing on a single snapshot, but corresponding to different events.

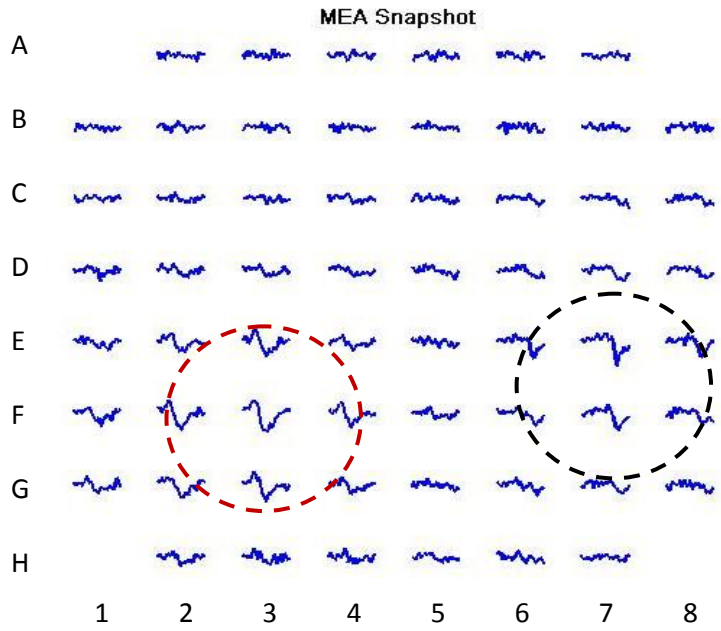


Figure 5.17: Spike event generation for a snapshot window of 3.0ms and a spike event window of 0.5ms. An action potential recorded on electrodes F3, F2, F4, E3 and G3 (circled in red) correspond to a single spike event. Electrodes E7 and F7 (circled in black) record a spike whose peak falls outside the spike event window are grouped separately under a different spike event.

Another function of the event matrix generator is to generate a set of event matrices. Each event matrix captures the peak amplitude and relative phase lags with respect to the highest peak, for electrodes which were grouped under a single spike event. Electrodes which did not detect a spike are set to have zero amplitude and phase. The event matrices are then iteratively matched to templates as described in the next section. The user manual and MATLAB code for the event matrix generator can be found in Appendix P4.

Template Matcher

The template matching module fits detected spikes with a linear combination of eigenmatrix templates generated from the calibration process. Each eigenmatrix template is subtracted from the event matrix and the mean squared error (Equation 5.2) is used to assess the quality of fit. The minimum mean squared error resulting from this matching process indicates the successful matching of a template to the detected spike.

Since each spike event can be made up of 2 or more simultaneously recorded spikes, the matching process is done iteratively on the residual until the best matched template is a blank template. This form of template matching also resolves overlapping spikes which are detected as a single spike event. The result of this iterative matching process is a set of matched templates which are assigned to each spike event. Figure 5.18 illustrates the results of matching two eigenmatrix templates to a spike event which contains two simultaneously recorded spikes. The user manual and MATLAB code for the template matcher can be found in Appendix P5.

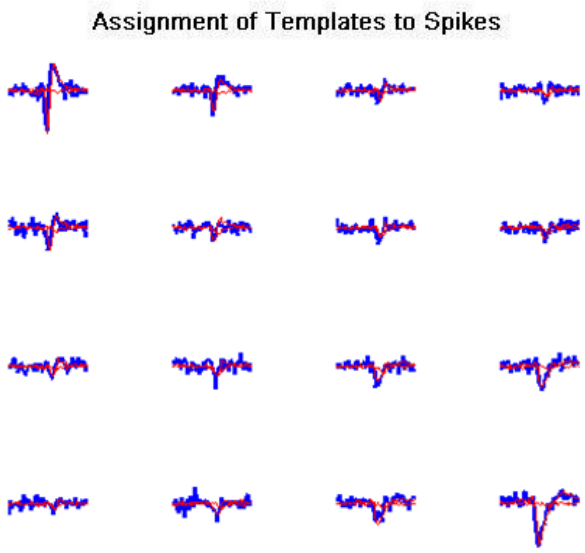


Figure 5.18: Spike Templates (blue) overlaid with detected spikes of a single spike event (red). Iterative eigenmatrix template matching identifies the presence of two simultaneously recorded spikes and assigns two different templates to this spike event.

The template matching module, supported by other modules described in previous sections assigns detected spikes to calibrated neurons of our system. The results of spike sorting are then summarised as a raster plot of sorted spikes, a spiking intensity map which allows the user to visualise areas of spiking activity, a spiking frequency histogram and an inter-spike interval histogram plot for each template and its corresponding neuron (Figure 5.29).

5.3.2 MEA-Based Test Platform

Current neural network simulators (Wilson 1989, Escolá 2008) are capable of reproducing realistic MEA-based recordings, but lack the calibration capabilities which our system offers. To assess the robustness and accuracy of the Spike Sorting Toolbox, we developed a simple MEA-based Test Platform in MATLAB to simulate both MEA recordings and the calibration process of generating eigenmatrix templates. The MEA-Based Test Platform comprises of 3 components: System Model Generator, MEA Recording Simulator and Calibration Simulator. These components mimic the actual experiment by laying out neurons on the MEA, calibrating neurons by generating eigenmatrix templates, and stimulating neurons while recording multiple electrode signals.

System Model Generator

Before data simulation can be carried out, the user has to initialise the electrode and neuron layout (Figure 5.19a). The user also has to include a database of spike shapes (Figure 5.19b) which allows the test platform to randomly assign spike shapes to each neuron on the MEA. Currently the database consists of 8 unique spike shapes extracted from test data by Quiroga, 2004. We model the spike amplitude such that it decreases exponentially with increasing electrode-neuron distance (Segev 2004), while the recording phase lag is a linearly dependent on this distance:

$$V_{ij} = V_{0i}\, \boldsymbol{exp}\left(-r_{ij} \middle/ \alpha\right) \tag{5.4}$$

$$\phi_{ij} = \beta r_{ij} \tag{5.5}$$

where V_i and V_{0i} are the recorded and normalised spike amplitudes of neuron i respectively, ϕ_{ij} is the recording phase lag, r_{ij} is the distance between the i^{th} neuron and j^{th} electrode and α and β are constants. The system generator computes scaling factors and phase lags for each pair of electrodes and neurons, which can be used to simulate MEA recordings or the calibration process. The user manual and MATLAB code for the system model generator is found in Appendix Q1.

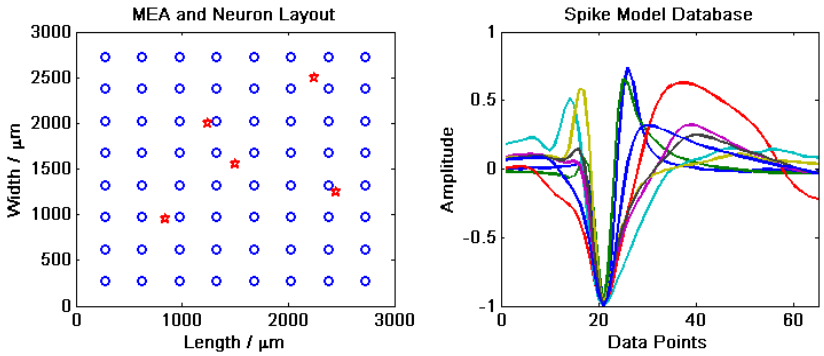


Figure 5.19: System model for the test platform. (a) Typical MEA and neuron layout. (b) Database of normalised spike models

MEA Recording Simulator

Based on the model described in Equations 5.3 and 5.4, the MEA Recording Simulator generates artificial spike recordings on each electrode of the MEA. The signal recorded on each electrode is a summation of the spiking activity of all neurons in the network and Gaussian distributed background noise, as shown in Equation 5.5:

$$E_j(t) = \sum_{i=1}^{N} V_i\left(t + \phi_{ij}\right) \boldsymbol{exp}\left(\alpha / r_{ij}\right) + V_{noise} \tag{5.5}$$

where E_j is the signal recorded on the j^{th} electrode, V_i is the spike amplitude of the i^{th} neuron, ϕ_{ij} and r_{ij} are the phase lags and distances between neuron i and electrode j respectively, α is a constant, and V_{noise} is the amplitude of Gaussian distributed background noise.

The simulation mimics the process of recording neuronal spikes in response to an external stimulus, such as optical illumination of ChR2 transfected neurons. Using the assigned spike shapes, scaling factors and phase lags, the MEA Recording Generator scales spikes and places them at user specified time points within the test data. The user manual and MATLAB code for the MEA recording simulator is found in Appendix Q2.

Calibration Simulator

The calibration simulator mimics the process of calibrating neurons by first generating sets of MEA recordings from the controlled stimulation of each neuron, and then generating eigenmatrix templates using the template generator module of the Spike Sorting Toolbox. Although the calibration simulator is effectively a combination of the MEA recording simulator and template generator, we have developed it as a individual module so as to automate the calibration process. Figure 5.20 illustrates the intensity maps of generated eigenmatrix templates. The user manual and MATLAB code for the calibration simulator is found in Appendix Q3.

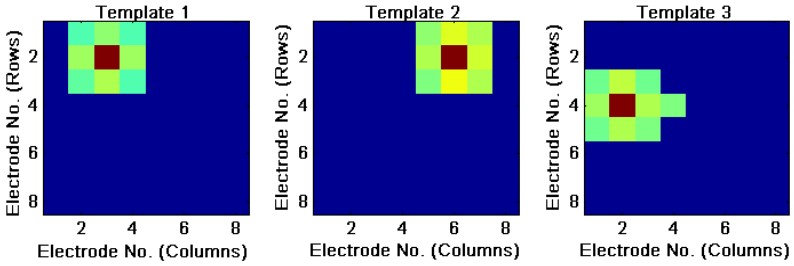


Figure 5.20: Calibration of each neuron results in the generation of a database of eigenmatrix templates. Each template contains spike amplitudes and relative phase lags on electrodes which detect spikes in more than 10% of the stimulus frequency. Intensity maps of each template show that spatial information is preserved.

5.4 Results

In this section, we present results from the characterisation of the Spike Sorting Toolbox using simulated data generated from the MEA-based Test Platform. In all simulations, signals were constructed at a sampling frequency of 10 kHz by superimposing Gaussian noise and spike trains of neurons as described in Equation 5.5. α was set to 500, while β was set to 0.5×10^{-6}, such that the velocity of the depolarisation wave was 2×10^{6} m/s (Equations 5.3 and 5.4). MEA recordings were pre-processed by using a 2nd order Elliptic bandpass filter (300 Hz to 3 kHz) to attenuate background noise. Various neuron layouts were used to test the capabilities of each module. The noise level was determined as the standard deviation of background noise, and was set to vary between 0.05 to 0.25.

Spike Detection

The spike detection module was characterised against varying noise levels. We placed the neurons far apart (Figure 5.21) so as to avoid overlapping spikes which will be detected as a single spike. Overlapping spikes are resolved by the template matching module, and will be described in later sections. Two sets of spike models (Figure 5.22) were used by the test platform to simulate MEA recordings, for the following noise levels at 0.05, 0.10, 0.15, 0.20 and 0.25.

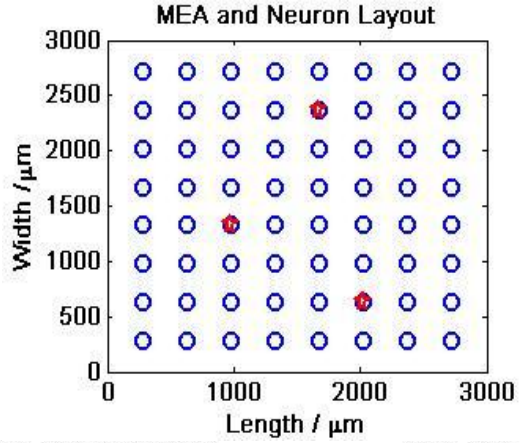


Figure 5.21: A simple MEA and neuron layout used in assessing the performance of the spike detection module. 3 neurons (red stars) were laid out on an 8 × 8 electrode MEA (blue circles).

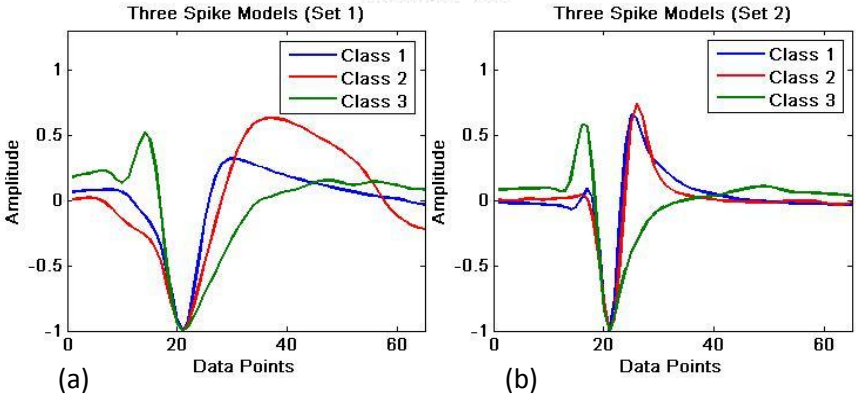


Figure 5.22: Two sets of spike models were used for assessing the performance of the spike detection module which uses the voltage threshold detection method.

Spike detection was then performed using a minimum and maximum threshold value of 5 and 50 multiplied to the background noise level respectively. To prevent multiple detection of the same spike, we set the detector dead time to be 1.5ms such that we only consider threshold crossings of 1.5ms apart. The spike detection module calculates the threshold values as a factor of the background noise level of each channel. The performance of the spike detection module in detecting spikes for all noise levels is summarised in Table 5.1.

Data Set	Noise level	Generated Spikes	Detected Spikes	Misses	False Positives
1	0.05	100	133	0	33
	0.10	100	109	0	9
	0.15	100	100	0	0
	0.20	100	82	18	0
	0.25	100	34	66	0
2	0.05	100	101	0	1
	0.10	100	100	0	0
	0.15	100	100	0	0
	0.20	100	92	8	0
	0.25	100	62	48	0

Table 5.1: Performance of the Spike Detection Module

As compared to the spike models of set 2, a large number of false positives were generated for set 1 at low noise levels. Since the noise level is low, the minimum threshold is also low. This results in the detection of the re-polarisation wave of the class 2 spike (Figure 5.22a). False positives may be avoided by increasing the detector dead time, such that the re-polarisation wave is not detected as a threshold crossing. We may also increase the threshold voltage in cases of low background noise levels so as to ensure that only peak amplitudes are detected. However, these solutions may result in detection misses and loss of information.

For both sets of spike models, there is an increase in the number of misses for the highest noise levels of 0.20 and 0.25. A high noise level leads to a high threshold value, resulting in a large number of un-detected spikes. A way to solve this problem is to lower the threshold value in order to detect more spikes, but this may lead to the detection of high amplitude noise, generating false positives.

System Calibration

The template generator module was characterised using the calibration simulator of the test platform. Calibration was carried out by stimulating each neuron in turn at a rate of 10 Hz, and spikes were detected using detection parameters described in the previous section. To test the robustness of the template generator, the module only considered electrodes which detected spikes in more than 0%, 5% and 10% of the total calibration stimulus. Figure 5.23 summarises the results through intensity maps of eigenmatrix templates generated for the above threshold values at noise levels of 0.10 and 0.15.

The results show that eigenmatrix template generation is prone to inaccuracies caused by false positives as shown in Figure 5.23a and 5.23d. By comparing Figure 5.23a to 5.23c and Figure 5.23d to 5.23f, we also see that the spatial resolution of each eigenmatrix template decreases with increasing threshold values. Accurate templates of high spatial resolution were found to be generated by considering electrodes which detected spikes in more than 5% of the total calibration stimulus.

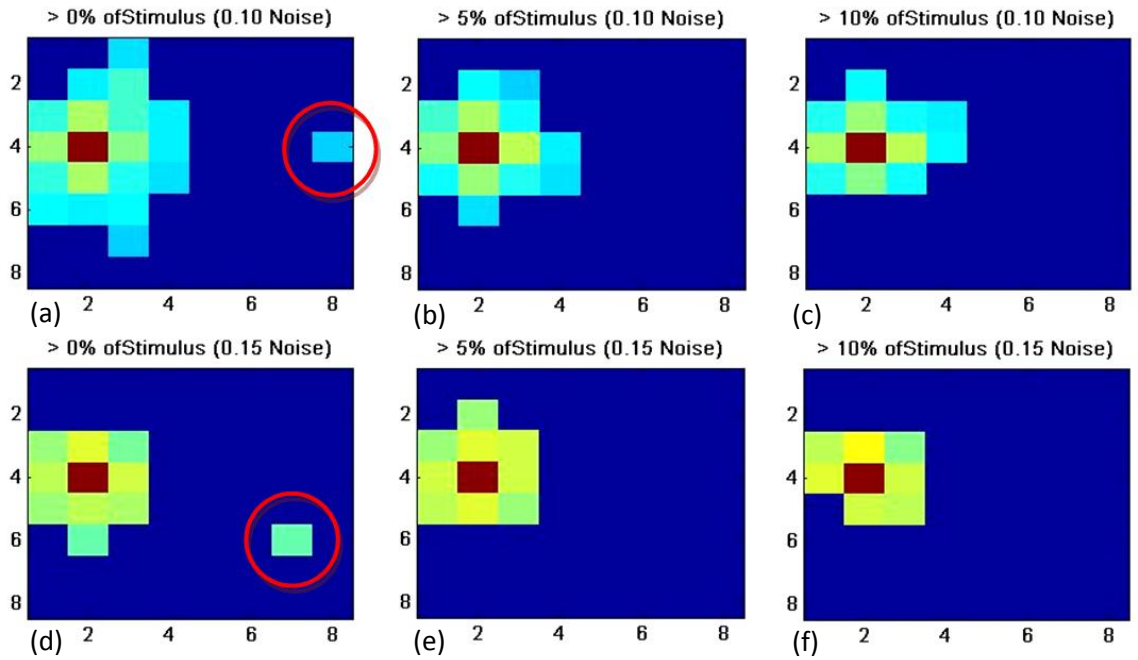


Figure 5.23: Intensity maps of eigenmatrix templates generated by considering electrodes which detect spikes in more than 0%, 5% and 10% of the total calibration stimulus. False positives (circled in red) cause unreliable eigenmatrix templates to be generated. By setting higher thresholds, we sacrifice spatial information for accuracy.

Event Matrix Generation

The event matrix generator module was characterised with respect to the user defined spike event window. Spikes detected were grouped into unique spike events by setting the spike event window at 0.1ms, 0.5ms, 1.0ms and 1.5ms. The results are illutrated through an intensity plot (Figure 5.24) of the relative phase lags for a spike event whose corresponding snapshot is shown in Figure 5.17. The snapshot records two spike events which do not occur simultaneously. Depending on the window, these recorded spikes may be classified as one or two separate spike events. Figure 5.25 shows how the ratio of spike events to generated spikes varies with the spike event window.

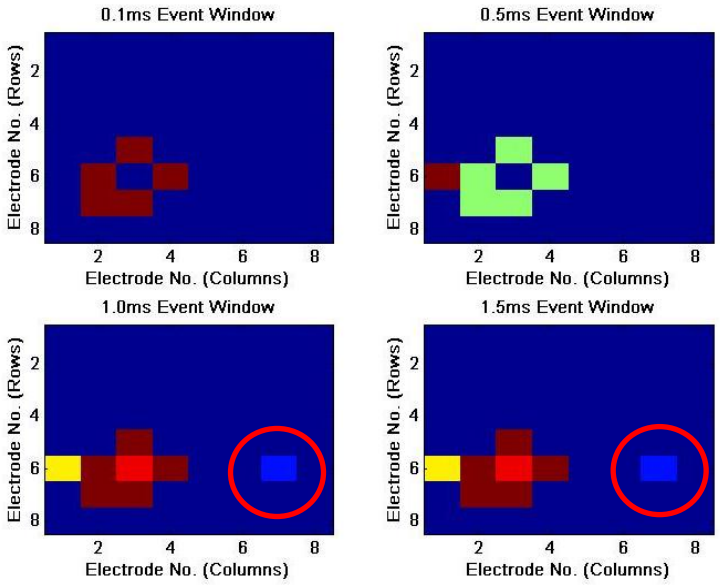


Figure 5.24: Intensity map of relative phase lags. As we increase the spike event window, the temporal resolution of event matrices drops. For the 1.0ms and 1.5ms spike event windows, a second spike is recorded (circled in red) and grouped under the same event as the main spike.

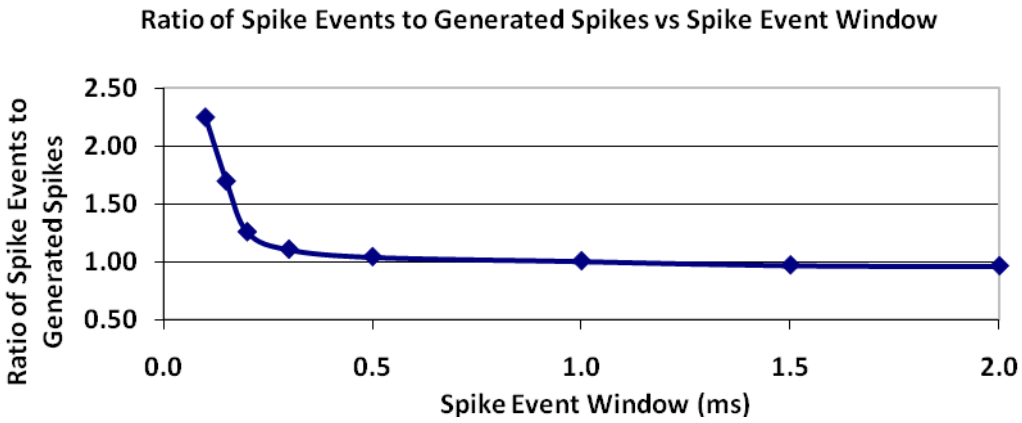


Figure 5.25: A plot of the ratio of spike events to detected spikes, against the spike event window. Decreasing the spike event window exponentially increases the number of spike events.

As seen from Figure 5.24 and 5.25, a smaller spike event window gives greater temporal resolution, but the ratio of spike events to generated spikes increases exponentially. A larger window generates fewer events, but introduces errors due to recording of two separate spike peaks which occur within the spike event window.

Template Matching

To test the reliability of eigenmatrix template matching at resolving overlapping spikes, we used a system layout as shown in Figure 5.26. Each pair of neurons was placed close to each other so that neighbouring electrodes would record spikes from both neurons simultaneously.

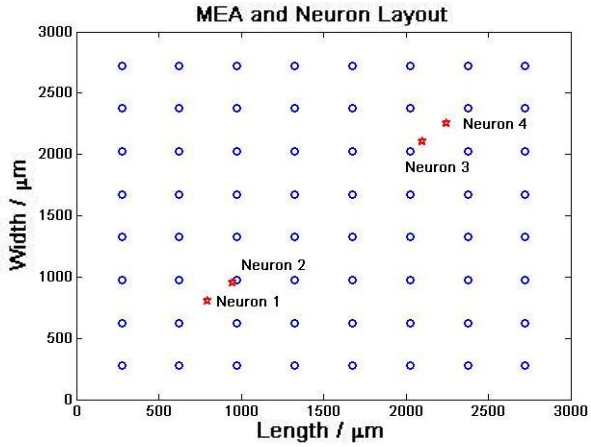


Figure 5.26: MEA and neuron layout used to characterise the template matching module. Two pairs of neurons (neurons 1 & 2 and neurons 3 & 4) were placed fairly close to each other so that neighbouring electrodes would record overlapping spiking activity from each pair.

Spikes were detected using the detection parameters described in previous sections, and the system was calibrated using a 10% detection threshold, producing eigenmatrix templates as shown in Figure 5.27. Due to the spatial proximity of each pair of neurons, the location of peak amplitudes are close, but differ in intensity.

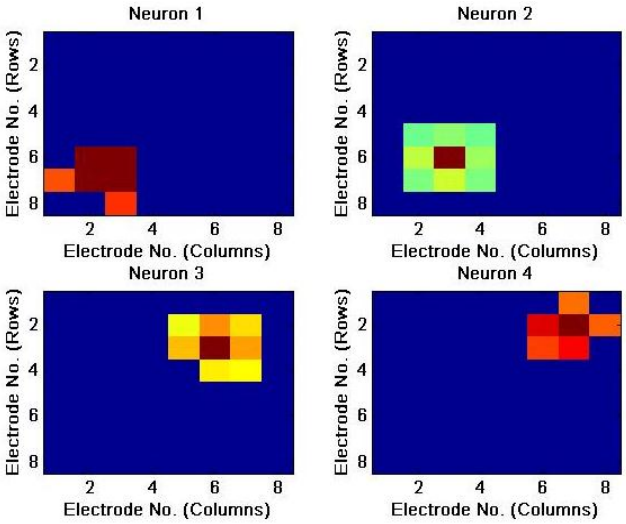


Figure 5.27: Eigenmatrix templates for 4 neurons laid out as shown in Fig 4.7. The close spatial proximity of each pair of neurons (neurons 1 & 2 and neurons 3 & 4) is reflected in the intensity plot of each eigenmatrix template.

MEA recordings were first simulated such that neurons 1 and 2 spiked simultaneously, while neurons 3 and 4 spiked 0.2ms apart. This was then repeated with all neurons 3 and 4 spiking simultaneously but spikes from neuron 1 and 2 were 0.2ms apart. Event matrices were generated with a 0.5ms spike event window so as to capture multiple spikes as a single spike event. Figure 5.28 shows an example of event matrices generated from overlapping of spikes. Templates were matched to event matrices by the template matching module and the results are summarised in Figure 5.29 and 5.30.

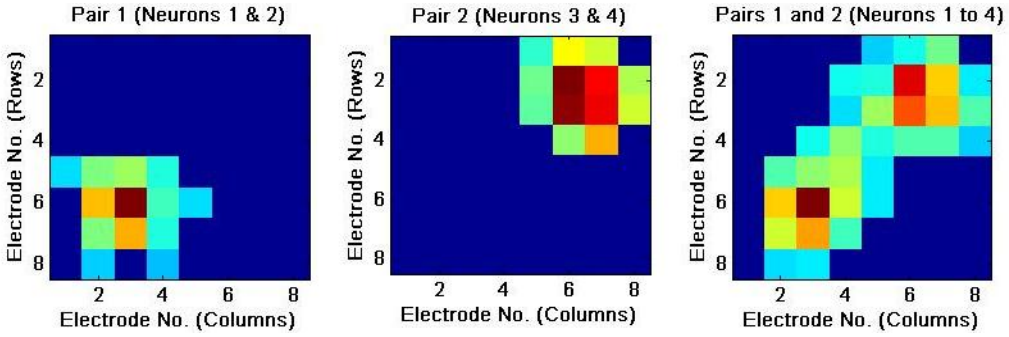


Figure 5.28: Event matrices for each pair of neurons. Due to the close spatial and temporal proximity of spikes, spikes from each pair of neurons overlap and are detected as one event.

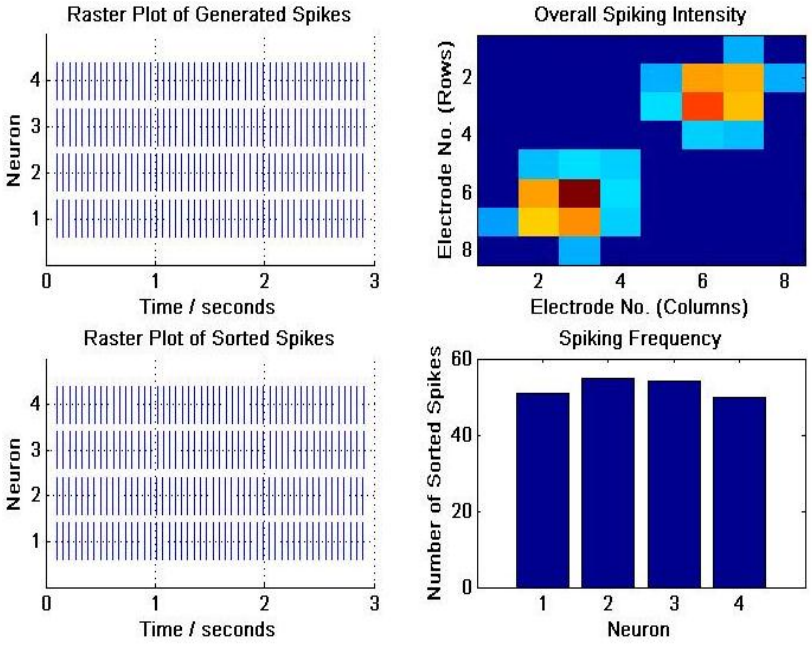


Figure 5.29: Results of resolving overlapping spikes for simultaneous spiking of neurons 1 & 2, and spiking of neurons 3 & 4, 0.2ms apart. Overlapping spikes were resolved effectively as seen from the raster plots of generated and sorted spikes.

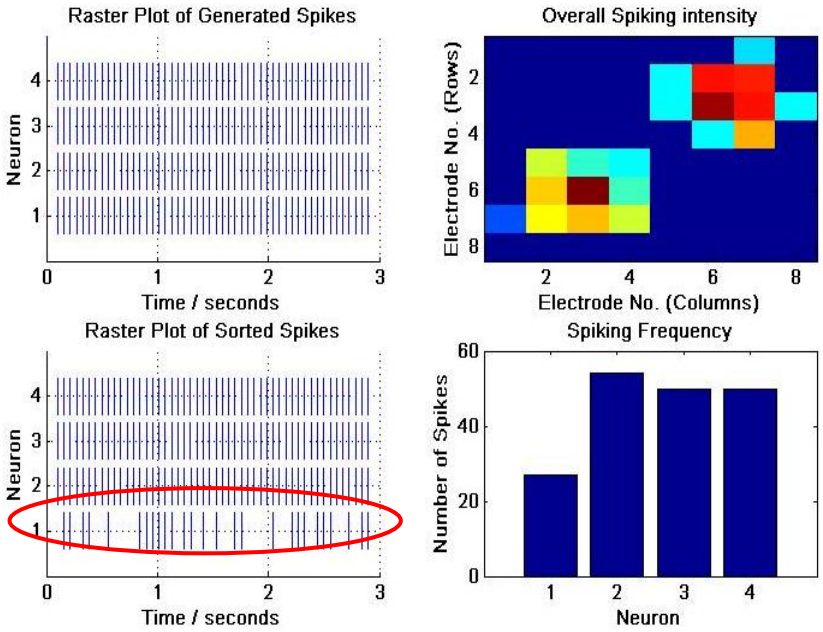


Figure 5.30: Results of resolving overlapping spikes for simultaneous spiking of all neurons, with spikes from neurons 1 & 2 spaced 0.2ms apart. Iterative eigenmatrix template matching does not fully resolve overlapping spikes for neurons 1 & 2 as seen from the raster plots (circled in red).

From Figure 5.29 and 5.30, we see that the template matching module gives varying results when resolving overlapping spikes. This is due to the spike event window being set at 0.5ms. Hence, we cannot temporally resolve spikes which occur less than 0.5ms of each other. This introduces errors which results in the assignment of spikes to neuron 2 but not neuron 1. A way to solve this is to use a smaller spike event window. When the window was reduced to 0.2ms, more spikes were assigned to neuron 1 as seen from Figure 5.31. However, this creates more spike events and requires more computational time for template matching.

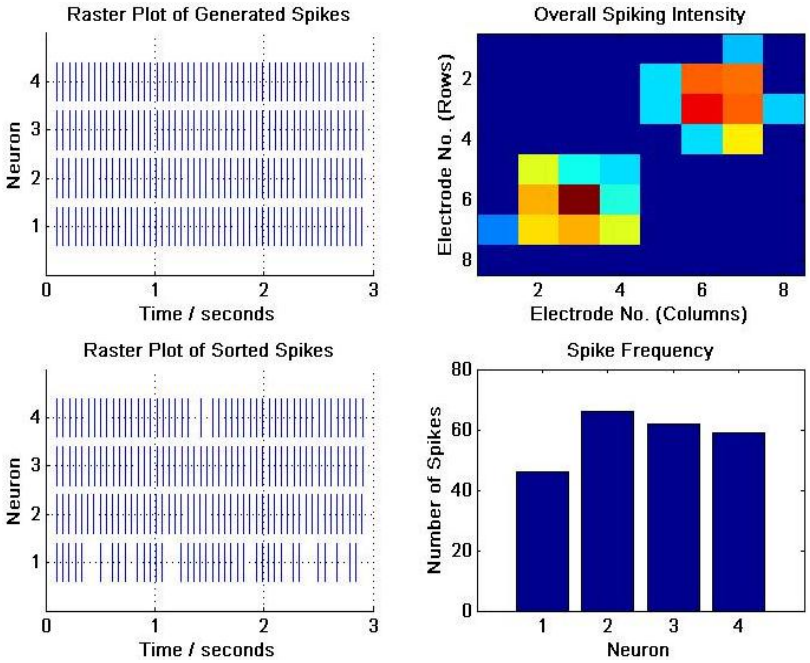


Figure 5.31: Results of resolving overlapping spikes for simultaneous spiking of all neurons, with spikes from neurons 1 & 2 spaced 0.2ms apart. Reducing the spike event window from to 0.2ms improves the temporal resolution, resulting in more effective assignment of spikes to neurons.

A System Application: Image Transfer

To demonstrate the capabilities of the Spike Sorting Toolbox, we simulate an application of transferring an image onto a neural network. The entire toolbox was used as a single module to reconstruct an optical stimulus pattern from the set of simulated MEA recordings. Data was generated using the MEA and neuron layout as shown in Figure 5.32.

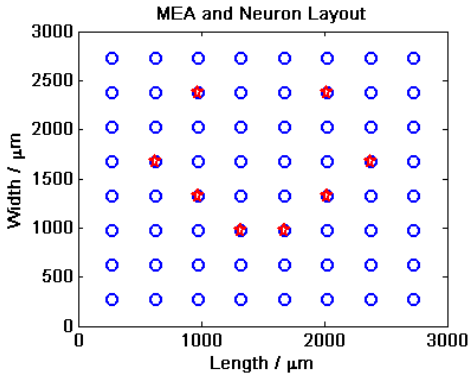


Figure 5.32: Layout of neurons (red stars) on an 8×8 MEA with 4 ground pins at each corner (blue circles). The layout resembles a "smiley-face" so as to demonstrate the possible application of image transfer. The micro-LED array optically stimulates the neural network with a "smiley-face" pattern which is to be reconstructed by spike sorting on MEA signal recordings.

Simulated system calibration was carried out over a 3 second period for each neuron, with a 10 Hz stimulus. The detection threshold was set to 10% and 8 distinct eigenmatrix templates were generated, each corresponding to one of the system's neurons. To assess the computational efficiency, 5 sets of MEA recordings were generated for the following durations of 3s, 5s, 7s, 9s, and 11s. The background noise level was set to 0.10 and spike trains were modelled with a Poisson distribution of inter-spike intervals with an average spiking frequency of 10 Hz. The algorithm was run on a CPU with a 2.33GHz, Intel Core 2 Duo processor and 3.48GB RAM. Computational time required to perform spike sorting was estimated using the MATLAB `tic` and `toc` functions.

Spikes were detected by the spike detection module using a detector dead time of 1.5ms and a minimum and maximum threshold value of 5 and 50 multiplied to the background noise level. The detected spike snapshots and spike times were then used as an input to the spike event matrix generator. A list of spike events and corresponding event matrices were generated using a spike event window of 0.5ms. Lastly, eigenmatrix templates were matched with each event matrix to assign spikes to neurons. The results of using the Spike Sorting Toolbox to reconstruct an image from a 7s long MEA recording are shown in Figure 5.33. Figure 5.34 summarises the computational time required for spike sorting to be carried out for each set of MEA recordings generated.

Figure 5.33 shows that iterative eigenmatrix template matching performs well at recovering the input stimulus. Scaling each eigenmatrix template by the corresponding spike frequencies reconstructs the initial pattern used to optically stimulate the neural network.

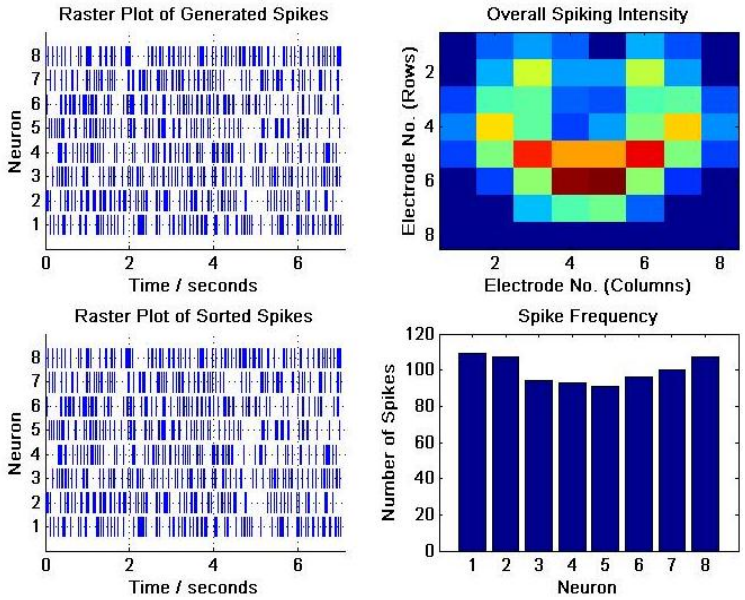


Figure 5.33: Results of performing spike sorting on artificial MEA recordings obtained from simulating the process of optically stimulating a neural network with a "smiley-face" pattern. Spikes were accurately assigned to neurons as seen from the raster plots of sorted and generated spikes. By scaling each eigenmatrix template by the corresponding spike frequency and then summing them, we are able to reconstruct the image from the set of MEA recordings as seen from the intensity plot.

From Figure 5.34, we see that computational time required for event matrix generation and template matching is much smaller than signal filtering. Hence, bottlenecks are unlikely to appear in the template matching process. The total amount of time required for spike sorting is approximately one-third of the length of signal recordings. Ignoring signal filtering, the amount of time required for spike detection, event matrix generation and template matching is reduced further by another half. Computational time also increases linearly with the length of simulated data. These results show that iterative eigenmatrix template matching is capable of accurately assigning detected spikes to neurons, at high computational efficiency.

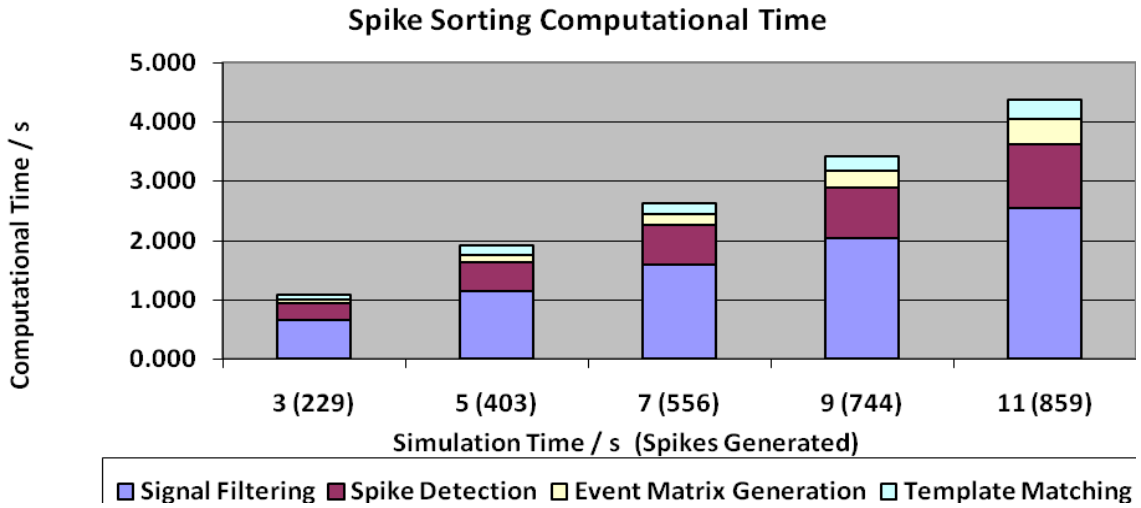


Figure 5.34: Computational time required for the modules of the Spike Sorting Toolbox to perform each stage in the spike sorting process. Computational time increases linearly with simulation time.

5.5 Evaluating the Spike Sorting Toolbox

In this chapter, we presented a modular toolbox for detecting and sorting neuronal activity recorded by multiple electrodes. Exploiting the calibration capabilities of our system, we developed a novel method of matching eigenmatrix templates to spike event matrices. To reduce the computational complexity of spike template matching, we extracted the most prominent features of each spike pattern to generate eigenmatrix templates. The method is fully automatic, unsupervised and is computationally fast, making it suitable for use in real time spike sorting.

Spike feature extraction was performed by calibrating each neuron and generating eignematrix templates. We reduced the dimensions of each unique spike pattern by extracting the peak amplitude and relative phase lags. By varying the detection criteria during system calibration, we can sacrifice spatial resolution for more reliable eigenmatrix templates which do not include low probability threshold crossings. This is important as templates which accurately capture the spike patterns of a neuron are required for the template matching process.

Spike detection was carried out using an amplitude threshold on bandpass filtered data. The threshold value was calculated automatically based on the estimate of the background noise level. Spike detection performed well in terms of the number of misses and false positives for moderate noise levels ranging from 0.10 to 0.20. Although this method did not work well for low and high noise levels, we can vary detection parameters so as to optimise the performance of spike detection.

Since multiple electrodes are able to detect spikes from a single neuron, we identified each unique spike event using a spike event window. By varying the window, we effectively change the temporal resolution of each event matrix. A smaller window gives us greater temporal resolution, but computational complexity is increased as uncessesary spike events are created through double counting. A larger window reduces computational complexity, but may introduce errors due to its low temporal resolution. In cases where confuency is low, the use of a larger window is more efficient, since the occurrence of overlapping spikes is low. On the other hand, the use of a smaller window is more efficient when dealing with highly confluent neuron cultures.

Of all user defined parameters we found that iterative eigenmatrix template matching is most sensitive to changes in the spike event window. Reducing the spike event window not only increases temporal resolution, but also increases the reliability of iterative eigenmatrix template matching. In cases where overlapping spikes occur frequently, reducing the spike event window optimises the accuracy of iterative template matching in resolving overlapping spikes. Finally, we showed that the order of complexity is linear, and is thus suitable for real time implementation.

5.6 Comparison to Previous Methods

Several methods have been developed for the purpose of spike sorting. Of all the methods presented, ICA appeared to address our problem very well: through ICA, we are able to recover the original spiking patterns of individual neurons from a mixture of signals observed on multiple recording electrodes on the MEA. However, ICA makes several assumptions which are easily violated. On careful examination, our system does not require a blind source separation technique like ICA to sort spikes. This is because our system can be calibrated by targeting and stimulating single neurons. This simplifies the spike sorting problem, turning it into that of partial blind source separation. Hence, we decided to employ the template matching approach which fully utilises the calibration capabilities of our system.

Our method of using iterative eigenmatrix template matching eliminates the need for clustering to be performed on spike features extracted through PCA or feature analysis methods. The clustering process is one of the main obstacles in realising real time spike sorting, as current clustering algorithms like k-means clustering required a high level of supervision. Although automatic clustering algorithms are available (Cheeseman 1996), they have a high computational load (Erman 2006) and are unsuitable for use in real time spike sorting.

The use of iterative spike template matching for sorting multi-channel recordings was demonstrated by Segev, 2004. Segev's approach involved identifying spike patterns from individual neurons, and creating multiple shifted spike patterns to achieve a 0.1ms temporal resolution. The expanded database of original and shifted spike templates was then used to match detected spikes and resolve overlapping spikes. In our algorithm, we take a different approach by identifying unique spike events with a spike event window. This is faster and reduces computational complexity, since the size of the template database is limited by the number of calibrated neurons. By reducing the spike event window, we are also able to achieve a similar temporal resolution.

As compared to Segev's approach, our method also offers higher computational efficiency in matching templates. We reduced the $M \times N$ dimensional spike templates used by Segev to $M \times 2$ dimensional eigenmatrix templates which capture the most important features of each spike pattern. Although information is lost when dimensions are reduced, we have shown that iterative eigenmatrix template matching remains reliable and accurate in assigning detected spikes to calibrated neurons, and in resolving overlapping spikes. Among all spike template matching algorithms, iterative eigenmatrix template matching would be the best suited for use in real time spike sorting due to its low computational load.

5.7 Future Work

A lot of effort has been put into developing and testing the basic capabilities of iterative eigenmatrix template matching. Currently, the MATLAB based algorithm is used via a command line interface. We hope to improve user experience by developing a Graphical User Interface (GUI) for the Spike Sorting Toolbox. This will serve as a user friendly tool for neurophysiologists to analyse multiple electrode recordings.

Although simple tests have been carried out to assess the performance of our algorithm in resolving simultaneously recorded overlapping spikes, we recommend that the algorithm be put through more rigorous testing to assess its full range of capabilities. These tests may include evolving spike shapes and burst firing neurons, where system re-calibration has to be carried out to maintain the accuracy of the eigenmatrix template database. We also recommend the use of realistic neural network simulators (Wilson 1989, Escola 2008) to generate MEA-based recordings, so as to include the possibility of action potentials which propagate from one neuron to another in a neural network. Besides this, we hope to characterise the performance of the Spike Sorting Toolbox with respect to the number of simultaneously spiking neurons. This will enable us to make recommendations for the optimal level of confluency required to give best spike sorting results with iterative eigenmatrix template matching.

Finally, we hope to make full use of the high computational efficiency of our algorithm by implementing it in real time. This will enable our system to deal with a possible real time, closed-loop optical stimulation feedback system and real time image recognition for use in a retinal prosthesis. In the long run, we aim to develop a powerful toolbox which is fully capable in achieving real time response recognition of stimulated neurons to stimuli of our system.