A Framework for Real Time and Automatic Spike Sorting of Multichannel Neuronal Activity

Wanyuan Huang¹, Nansong Wu², Liang Huang³

¹Supermicro Inc, CA, USA,
²Department of Electrical Engineering, Arkansas Tech University, Arkansas, USA
³Department of Information & Electronic Engineering, Zhejiang Gongshang University, Hang Zhou,
Zhejiang, China

ABSTRACT

Spike sorting is a primary and essential procedure for the realization of brain in the neuroscience and it provides a connection between the neural behavior and external behavior of an animal for further application such as movement prediction and brain machine interface (BMI). A real-time and automatic spike sorting system for 16-channel neural recording based on hardware-software co-design is proposed in this study. The feature extraction in this study utilizes the discrete derivative method to improve the spike separation and chooses the principal component analysis to select few dominant features for reduction of indistinctive data. There is a significant improvement for spike separation on feature space with the help of the discrete derivative method, and thus, the accuracy of the spike sorting is enhanced on indistinguishable data set from the result.

Keywords: Spike sorting, brain machine interface, discrete derivative method.

1. INTRODUCTION

In the last decades, neuroscientists endeavor to figure out the brain function and the relationship between nervous system and behavior, such as judgment, thinking, or movement. Brain is the center of the nervous system and contains tens of billions computation units, called neurons, to transmit complex information by chemical signals or electrical signals. The electrical signals, regarded as neural signals, could be measured by the electrode and, thus, the analysis for neural signals is a major process for the neuroscience. Neuroscientists develop a myriad of techniques to record and extract neural signals from different region of the brain and the recorded neural signals are used for further researches on neurophysiology. Nowadays, for the different objectives and the improvement in implantable device technology, multiple electrodes on the same probe are designed and utilized such as Michigan probe [1][2][3] or NCTU probe [4]. Multichannel recording therefore becomes the standard tool for the research on neurophysiology. Several multichannel spike sorting systems or algorithms concerning in multichannel spike sorting [5][6][7][8][9][10] are developed for compatible to these kind multiple electrode probes. Many researchers are focused on efforts to develop devices using silicon instead of GaN, to achieve lower costs [2][7].

The neural spike sorting process typically consists of three standard steps [10][11] and includes the spike detection, the feature extraction, and the spike clustering and classification[12][13][14]. The spike detection is for the separation of the spike event and the background noise[15][16]. Several well-known algorithms for the spike detection have been developed such as the threshold method [17][18][19], the energy-based method [20], the template matching method [21], and the frequency-based method [15]. Above spike detection methods are suitable for real-time implementation, and the threshold method and the energy-based method are widely utilized as the first step of spike sorting process. The feature extraction picks the seldom significant features to represent the collected spike waveform, and the waveform parameter extraction [19], principal component analysis (PCA) [11], and discrete wavelet transform (DWT) [20] are commonly used. PCA and DWT could work well than the waveform parameter extraction while the waveform parameter extraction is the most suitable for real-time implementation[21][22][23][24][25]. The spike classification is used to determine which spikes coming from the same group or recording from the same neuron. K-means [21], the hierarchical clustering [26], linear discriminant analysis (LDA) [27][28][29], and expectation-maximization algorithm (EM) [30] are well-known methods for the spike classification. Another indispensable step for spike sorting process is the spike alignment, and the methodology for the spike alignment typically uses the waveform peak (minimum or maximum magnitude) [31][32] or slope peak [33][34][35][36].
To abate the complicated training phase, another real-time spike sorting structure is presented in [37][38][39][40]. The algorithm for this spike sorting structure does not include complex training phase. Threshold method and waveform parameter method are utilized for spike detection and feature extraction, respectively. As for classification process, the user-defined rule [40] is used to determine the class of neurons. This structure is very suitable for hardware implementation with its lower computation complexity and the whole spike sorting process on a micro circuit is feasible. However, it might get very low accuracy of spike sorting as the difficult conditions on neural data occur.

To solve the above issues and achieve the objectives, a real-time and automatic spike sorting system for 16 channel neural recording based on hardware-software co-design is proposed in this study. The system would accomplish the goal of spike separation improvement on the feature space, validation for spike trains from the same neuron, 16 channels neural recording simultaneously, elimination on training time and storage usage, and adaptive adjustment of system parameters.

2. METHODS

A. System Architecture

The proposed 16-channel adaptable on-line spike sorting system consists of two major blocks: neural signal processor (NSP) and neural spike sorter (NSS), as showed in Fig.1. As seen in the Fig.1, there are two major blocks: neural signal processor and neural spike sorter. The digitalized neural data are fed to neural signal processor which is implemented in hardware design and focus on spike detection and feature extraction. Extracted features for each spike are received by neural spike sorter to determine which cluster this spike belongs to. Classification results are utilized as a criterion to adjust parameters in neural signal processor by close-loop design.

For real-time multi-channel processing, 16-channel neural data are digitalized with 20 KHz sampling rate and 12 bit-width resolution for each channel in advance, and are merged to serial data with time-multiplexed and channel-interleaved framework. NSP receives continuous samples of each channel and processes those for reducing data rate and extracting significant spike information. It is implemented on a field-programmable gate array (FPGA) (Cyclone II, EP2C35F484C8N, Altera, USA) with 50 MHz system clock to achieve on-line and autonomous neural signal processing. There are three main tasks on NSP: two-stage spike detection, discrete derivative transformation, and fast principal component analysis (fast-PCA) extraction method. The two-stage detection acquires the detected spike waveforms and their respectively relative time, and the discrete derivative transformation emphasizes high-frequency spectrum of spike waveform while depresses low-frequency one for better spike separation. The fast principal component analysis extraction method extracts significant features from spike waveform after discrete derivative transformation. After preprocessing the neural data, the spike information for each channel is received to personal computer by data acquisition card (DAQ-card, USB-6259, National Instruments, USA). NSS attempts to identify different spikes using their features generated from neural signal processor and then rebuilds the spike train of each neuron by the respective inter-spike interval. There are three main functions: single linkage clustering, online classification, and feedback control, and is implemented by LabVIEW 8.6 (National Instruments, USA). The single linkage clustering automatically computes and obtains the number of cluster and the center of each cluster for every channel, and then the classification determines which spike belongs to which cluster on real-time. A close-loop is composed between NSP and NSS to lower the false detection by varied background noise. NSS transmits the feedback command to NSP for adjusting discriminate criterion for spike detection or retraining principal components for any channel per the sorting performance if necessary. Finally, the firing pattern of different recorded neurons for each channel would be displayed on the client to assist user for further analysis or application.

![Diagram](image.png)

**Figure 1.** System architecture of the 16-channel adaptable on-line spike sorting system.
The first step of processing the neural signals sampled from ADC is the detection of time points of firing events and this procedure is called spike detection. It preliminarily reduces data rate by extracting only action potentials (or spikes) from neural signal which consists of redundant information such as noise. Some spike detection algorithm which is suitable for hardware implementation is discussed in [42] and [43]. The proposed spike detection method in this work is two-stage spike detection which combines the advantage of adaptive threshold method [44][45][46] and nonlinear energy operator (NEO) [47], and is beneficial to lower the case that noise be detected as spikes. The detected and aligned spike waveforms are then transmitted out with firing time for spike extraction step. Adaptive threshold and NEO are well-known algorithms for spike detection and the computation of this two methods are appropriate for hardware implementation [43]. Adaptive threshold concerns on the time-domain of processed signal while NEO works on the energy-domain of that. Both spike detection methods are positive and negative effects. Adaptive threshold sets a threshold based on the background noise and directly detects potential spikes when the magnitude of neural signal is higher than this threshold. However, the drawback of adaptive threshold method is that it is sensitive to changing background noise [48]. NEO method has more capacity for distinguishing spike and noise at low signal-to-noise ratio (SNR) [47] while it is necessary to predetermine the criterion for telling spikes apart noise [43]. Adaptive threshold method would be suitable to directly find the potential spikes and NEO method would be helpful to discriminate spike from noise instead of generating threshold on energy-domain.

Two-stage detection method is proposed as shown in Fig. 2. As seen in the Fig. 2, first stage uses adaptive threshold to trigger spike event, and second stage uses NEO to check whether the energy of spike event exceeds the criterion which is derived by multiplication of the feedback discrimination factor and noisy energy.

First stage uses the adaptive threshold method to find the potential spike event. Pair of thresholds could be expressed as following equation:

\[
Thr_H = mean(x) + 4 \cdot \hat{\sigma}_N
\]

(1)

\[
Thr_L = mean(x) - 4 \cdot \hat{\sigma}_N
\]

(2)

\[
\hat{\sigma}_N = IQR \times 0.75
\]

(3)

Where \(Thr_H\) and \(Thr_L\) mean high threshold and low threshold, respectively, \(x\) denotes as samples of the neural signal (including spikes and background noises) and \(\hat{\sigma}_N\) is an estimate of the background noise. The threshold equation works on the hypothesis that the distribution of background noise is approximately Gaussian. The conventional estimation of
the amplitude threshold using the standard deviation of neural signals may result in very large threshold values due to high firing rates and large spike amplitudes. Therefore, using the interquartile range (IQR) multiplied with 0.75 as an estimation of the background noise may diminish spike interferences. Besides, the noise level is not always uniform under the situation of neural signal recording. Thus, the adaptive threshold is utilized as mentioned in [44]. The adaptive threshold modifies the estimation of background noise by maintaining that the duty cycle of the neural signal is 15.9% because the probability of exceeding the estimation of background is 0.159.

The second stage of the proposed spike detection is NEO-based method for distinguishing whether the potential spike event detected by first stage method is indeed a spike event or maybe noise with high magnitude. The NEO-based spike detection method judges the potential spike event as real spike event by the comparison between average energy of noise and energy of spike event. The estimation of energy is derived by NEO as following equation:

\[
\psi(x[n]) = x[n]^2 - x[n+1] \cdot x[n-1]
\]

where \( \psi( ) \) is the NEO operator and \( X_s[n] \) is the centered sample of neural signal on the time n. The average energy of noise \( E_{\text{noise}} \) and the energy of spike event \( E_{\text{event}} \) is defined as below:

\[
E_{\text{noise}} = \frac{1}{N_{\text{noise}}} \sum_{n=L_0}^{L+n_{\text{event}}} e_{\text{noise}}[n]
\]

\[
E_{\text{event}} = \frac{1}{N_{\text{event}}} \sum_{n=L_0}^{L+n_{\text{event}}} \psi(x_{\text{event}}[n])
\]

where \( N_{\text{noise}} \) and \( N_{\text{event}} \) are the number of noise samples and the number of spike event samples in the signal, respectively, and \( e_{\text{noise}} \) is the energy of noise which exceeds the threshold as the following equation:

\[
e_{\text{noise}}[n] = \begin{cases} 
\psi(x[n]), & \text{if } x[n+1] < x[n] \cdot x[n-1] < Thr \ni \\
0, & \text{otherwise}
\end{cases}
\]

Once the energy of spike event is \( F \) times as great as the average energy of noise, the potential spike event is regarded as the real spike event. The discrimination equation is as below:

\[
S[n] = 1, \text{ if } E_{\text{event}} > F \cdot E_{\text{noise}}
\]

where \( F \) is the discriminate factor adjusted by the feedback control of neural spike sorter based on the performance of online spike sorting.

**C. Discrete Derivative**

The basic concept of the discrete derivatives is to compute the slope at each sample point over a number of different time scales and is presented in [43]. In this work, the time scale is set to 1 and the dimension of the transformed signal is same as that of previous signal. Discrete derivative can improve the spike separation and thus enhance the accuracy of spike sorting on the neural signal with resemble spike shapes derived from distinct neurons. It works by accentuating the sharp slope of the signal and diminishing the influence of the dull slope one. In [49], it summaries that the separation of similar neurons is improved when emphasizes on the high-frequency signal spectrum and decreases the effect on the low-frequency signal spectrum. The discrete derivative could also emphasize on the high-frequency domain and decrease the effect on the low-frequency domain, and thus, the discrete derivative could be utilized on the spike waveform before doing feature extraction. For efficient hardware implementation and to maintain the DC level of the transformed signal on the value 2048, the equation of discrete derivative is modified as below:

\[
\diff(n) = \text{Trunc}(s(n) + 4096) - s(n-1)
\]

where \( S[n] \) is the sample point of spike waveform on time n and \( \text{Trunc}[ ] \) is the truncation operator to truncate the signal for most significant 12 bits. The present sample point of spike is expanded to 13 bits with one bit 1 on MSB instead of directly adding 4096 to spike signal and then the truncation is operated for obtaining only 12 bits data after subtract operation is done.

**D. Feature Extraction**

For the spike clustering purpose, the problem of dimensional reduction is revealed because the computational complex of spike clustering increases with the number of dimensions. Therefore, feature extraction is essential on the flow of spike
sorting and it translates the hyper dimensional spike event into a more tractable one. Principal component analysis (PCA) is the most widely used for spike feature extraction [11]. The PCA is a method which retains major information of the spike event by transforming the data into significant features with smaller size of dimensions.

To get the significant features with few dimensions, it is necessary to find the principal components (PCs) of the spike events and the PCs are derived by computing the eigenvectors of the covariance matrix of the detected spike waveforms. However, the major cost of PCA implementation in hardware design is to calculate the covariance matrix and the leading eigenvectors of the covariance matrix. The huge memory to store spike waveforms for the covariance matrix and the high complex computation of the eigenvalue decomposition (EVD) which computes the eigenvector of covariance results in non-efficient hardware implementation. Therefore, the pre-estimate covariance matrix hardware structure [50] is used to derive the calculation of covariance matrix effectively, and the PCA hardware structure [51] based on the iterative eigenvector algorithm [52] is used to replace the EVD process.

The block diagram of the principal component analysis.

![Diagram](image)

The block diagram of the PCA method in hardware design is shown in Fig. 3 and there are five major modules including: (1) receiver, (2) covariance generator, (3) PCs generator, (4) scores generator, and (5) transmitter. The receiver and the transmitter are used to acquire the spike information (spike waveforms) after doing discrete derivative and transfer the spike information (features) out after scores are calculated, respectively. The input is the spike information with 25-point waveform and the output is the spike information consisted of three scores from PC-1 to PC-3. The covariance generator is used to accumulate spike waveforms for covariance matrix. The PCs generator focuses on the computation of PCs based on the covariance matrix. The scores generator calculates the scores from the PCs and the spike waveforms [53][54][55][56].

Mean pre-estimate method [50] is utilized for reducing the memory size because if using the same data to calculate both mean and covariance, it is necessary to measure the mean first and then derive covariance matrix by the mean, and the problem of storing all the data on memory is revealed. It operates that calculating the mean uses a set of data and then computing the covariance matrix uses another set of data without storing any previous spike waveform [57][58].

The iteration eigenvector distilling algorithm is presented in [51] to avoid using such complex computation method as EVD, matrix diagonalization, symmetric rotation and matrix inverse process for computing eigenvector of the covariance matrix. The number of PC is set to 3 and the time of iteration is set to 10 per the simulated result. However, the dynamic range of $\varphi_p$ is increased by $24+6$ bits for one distilling process and $12 \times 2 + 6$ bits for one orthogonal process. To limit the dynamic range of $\varphi_p$, the scaling process is presented to quantize the maximum and minimum value of $\varphi_p$ between 12-bit signed-magnitude numbers. Note that the maximum bit-width of the PCs generator is $3 \times 12 + 6$ bits at the orthogonal process.

The scores generator consists of three multiply-accumulate (MAC) structure for deriving scores by calculating the inner product between spike waveform and PCs. There are three scores for a spike waveform after computing the inner product. The size of register for storing inner product of each score is $(12 \times 2 + 5)$ bits. However, the bit-width of output is set to 12 bits. Therefore, the right-shift process is necessary to fit the limitation of bit-width. The 12-bit right-shift process could fit the bit-width from experimental result.

### E. Clustering and Online Classification

The method of the clustering in this work is the single linkage method (SLM) [26] which operates by calculating the distances between clusters in hierarchical clustering and the distance is computed as the distance between the closest elements in the two clusters. The objective of SLM is to present the hierarchy of the distance of objects, and cluster to the
group per the linkage threshold. Before listing the detail steps of the SLM clustering, some parameters and assumptions of the SLM clustering are described. First, the collection of the training cluster data is set to 300. Second, the maximum number of neuron in one channel is set to 5. Third, the minimum number of clustering \( C_{\text{min}} \) is set to 50 per the assumption of maximum cluster[59]. The algorithm flow of the SLM clustering is as following steps.

**Calculating the distance between each data:**

After collecting all training objects (spikes), the first step is to calculate Mahalanobis distance \( (d_o) \) between each objects. The Mahalanobis distance is better than the Euclidean distance on this work because the distribution of the feature clusters after doing discrete derivative would be like elliptic shape when the different spikes are similar. The mean \( (d_m) \) and standard deviation \( (d_\sigma) \) of distance are then computed for calculating the linkage threshold.

**Calculating the linkage threshold:**

The linkage threshold is separated into twenty scales to derive the best clustering result. The linkage threshold is set as following equation:

\[
T_q = T_{\text{max}} - (q - 1) \cdot T_0
\]

\[
T_0 = \frac{T_{\text{max}} - T_{\text{min}}}{20},
\]

\[
T_{\text{min}} = d_m - 2 \cdot d_\sigma, \quad T_{\text{max}} = d_m + 2 \cdot d_\sigma \quad (10)
\]

where \( q \) is the segment parameter for the level of the linkage threshold, and is initially set to 1. The segment parameter, \( q \), ranges from one to twenty.

**Clustering starts:**

In the beginning, all objects are identified as separate clusters, \( C_1, \ldots, C_{300} \). The increment variable \( t \) is initially set to 1 and the distance between two clusters is defined to be \( d_{ij} \).

**Combining clusters:**

The clusters \( C_i \) and \( C_j \) are combined to a new cluster denoted \( C_{300+t} \), if the distance between this two clusters is minimum compared with others and the following conditions is fitted: (1) \( d_i \leq T_q \), (2) \( i \neq j \) and (3) \( C_i \) and \( C_j \) are different clusters. The distance between the new cluster \( C_{300+t} \), and all remaining clusters \( C_k \) is as \( d_{C_{300+t}, C_k} = \min\{d_{C_i, C_k} : d_{ij} \leq T_q \} \). The new cluster \( C_{300+t} \) is added while the clusters \( C_i \) and \( C_j \) are removed. Finally, \( t \) is incremented as \( t = t + 1 \) and the procedure is returned to step 4 until there are no new clusters that could be added to.

Increasing the level of linkage threshold:

The clustering result of each q-level linkage threshold is set as \( G_q \) where \( G_q = \{C_i\} \) and \( C_i \) includes at least one object. \( q \) is then incremented as \( q = q + 1 \) and the procedure is returned to step 2 until \( q \) is equal to 20.

Determining the optimum linkage threshold:

The optimum linkage threshold is selected for the best spike clustering result. The segment parameter with the maximum number of clusters and spikes is the optimum segment parameter. Then, the number of clusters is determined by the linkage threshold with the optimum segment parameter. There are two steps to find the optimum linkage threshold. First, the element number of \( C_i \) that is less than \( C_{\text{min}} \) are removed for each \( G_q \) and new \( G_q \) is denoted as \( G_q' \). Second, the optimum linkage threshold is the potential linkage threshold with maximum objects of \( G_q' \), and then the centroid of each clusters \( (m_c) \) are calculated.

When the SLM clustering is completed, the online classification starts to classify the new spike \( (x) \) for \( K \) clusters as listed as following equation:

\[
C(x) = \begin{cases} 
0, & d_{\text{min}} \geq w \cdot T_0 \\
\arg \min_c \{M(m_c - x)\}, & \text{otherwise}, \quad c = 1, \ldots, K
\end{cases}
\]

\[
d_{\text{min}} = \min_c \{M(m_c - x)\}, \quad c = 1, \ldots, K \quad (11)
\]

where \( w \) is a weight for the optimum linkage threshold and influences the classified boundaries of cluster, and \( M(\cdot) \) means Mahalanobis distance. Finally, the classified label \( (C(x)) \) of new spike is determined. Note that label 0 means the unclassified spike.
F. Feedback Rule

The feedback rule is used to adjust the discriminate factor of the spike detection for avoiding the occurrence that background noise is detected and regarded as spike event. In the order word, the feedback rule is beneficial to decrease the false-alarm rate ($P_{fa}$). The adjustment of the discriminate factor is performed on a close-loop between neural signal processor and neural spike sorter in real-time. The typical feedback method is to increase or decrease the threshold depending on the performance impact of previous adjustments [40]. In the real-time experiment, it is difficult to directly measure the false-alarm rate ($P_{fa}$) and, therefore, the percentage of unclassified event on the spike sorter is treated as the false-alarm rate. The feedback rule has three conditions that are based on the false-alarm rate ($P_{fa}$) as listed on Table 1. When the false-alarm rate is larger than 30%, the discrimination factor increases 3-unit steps and the retraining command is set to retrain PCs and clusters. When the false-alarm rate is equal to 0% and the previous false-alarm rate is also 0%, there is no need to give adjustment. However, if the false-alarm rate increases from 0% to higher, the discrimination factor increases 1 step, and vice versa. When the false-alarm rate comes between 15% and 30%, the discrimination factor increases 3-unit steps. When the false-alarm rate is under 15%, the adjustment is based on the performance change after previous adjustment. If the performance change is improvement, the new adjustment is the same as the previous adjustment, and if is degradation, the new adjustment is the opposite of the previous adjustment. The feedback command is ordered per 100 spikes for every channel.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Performance change</th>
<th>Previous adjustment</th>
<th>New adjustment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{fa}=0%\rightarrow0%$</td>
<td>Regardless</td>
<td>Regardless</td>
<td>no adjustment</td>
</tr>
<tr>
<td>$P_{fa}=0%\rightarrow$ higher</td>
<td>Regardless</td>
<td>Regardless</td>
<td>+1 step</td>
</tr>
<tr>
<td>$P_{fa}=higher\rightarrow0%$</td>
<td>Regardless</td>
<td>Regardless</td>
<td>-1 step</td>
</tr>
<tr>
<td>$P_{fa}&gt;30%$</td>
<td>Regardless</td>
<td>Regardless</td>
<td>+3 step and retrain</td>
</tr>
<tr>
<td>$15%&lt;P_{fa}\leq30%$</td>
<td>Regardless</td>
<td>Regardless</td>
<td>+3 step</td>
</tr>
<tr>
<td>$P_{fa}\leq15%$</td>
<td>Improvement</td>
<td>Increase</td>
<td>+1 step</td>
</tr>
<tr>
<td></td>
<td>Improvement</td>
<td>Decrease</td>
<td>-1 step</td>
</tr>
<tr>
<td></td>
<td>Degradation</td>
<td>Increase</td>
<td>-1 step</td>
</tr>
<tr>
<td></td>
<td>Degradation</td>
<td>Decrease</td>
<td>+1 step</td>
</tr>
</tbody>
</table>

3. RESULTS

The simulation results in this work are all evaluated by the Matlab and includes: spike detection for evaluation of two-stage spike detection, feature extraction for comparison between the $P_{CA}$ with neural signal processor and neural spike sorter in real-time. The typical feedback method is to increase or decrease the threshold depending on the performance impact of previous adjustments [40]. In the real-time experiment, it is difficult to directly measure the false-alarm rate ($P_{fa}$) and, therefore, the percentage of unclassified event on the spike sorter is treated as the false-alarm rate. The feedback rule has three conditions that are based on the false-alarm rate ($P_{fa}$) as listed on Table 1. When the false-alarm rate is larger than 30%, the discrimination factor increases 3-unit steps and the retraining command is set to retrain PCs and clusters. When the false-alarm rate is equal to 0% and the previous false-alarm rate is also 0%, there is no need to give adjustment. However, if the false-alarm rate increases from 0% to higher, the discrimination factor increases 1 step, and vice versa. When the false-alarm rate comes between 15% and 30%, the discrimination factor increases 3-unit steps. When the false-alarm rate is under 15%, the adjustment is based on the performance change after previous adjustment. If the performance change is improvement, the new adjustment is the same as the previous adjustment, and if is degradation, the new adjustment is the opposite of the previous adjustment. The feedback command is ordered per 100 spikes for every channel.

The simulation results in this work are all evaluated by the Matlab and includes: spike detection for evaluation of two-stage spike detection, feature extraction for comparison between the $P_{CA}$ with neural signal processor and neural spike sorter in real-time. The typical feedback method is to increase or decrease the threshold depending on the performance impact of previous adjustments [40]. In the real-time experiment, it is difficult to directly measure the false-alarm rate ($P_{fa}$) and, therefore, the percentage of unclassified event on the spike sorter is treated as the false-alarm rate. The feedback rule has three conditions that are based on the false-alarm rate ($P_{fa}$) as listed on Table 1. When the false-alarm rate is larger than 30%, the discrimination factor increases 3-unit steps and the retraining command is set to retrain PCs and clusters. When the false-alarm rate is equal to 0% and the previous false-alarm rate is also 0%, there is no need to give adjustment. However, if the false-alarm rate increases from 0% to higher, the discrimination factor increases 1 step, and vice versa. When the false-alarm rate comes between 15% and 30%, the discrimination factor increases 3-unit steps. When the false-alarm rate is under 15%, the adjustment is based on the performance change after previous adjustment. If the performance change is improvement, the new adjustment is the same as the previous adjustment, and if is degradation, the new adjustment is the opposite of the previous adjustment. The feedback command is ordered per 100 spikes for every channel.

The simulation results in this work are all evaluated by the Matlab and includes: spike detection for evaluation of two-stage spike detection, feature extraction for comparison between the $P_{CA}$ with neural signal processor and neural spike sorter in real-time. The typical feedback method is to increase or decrease the threshold depending on the performance impact of previous adjustments [40]. In the real-time experiment, it is difficult to directly measure the false-alarm rate ($P_{fa}$) and, therefore, the percentage of unclassified event on the spike sorter is treated as the false-alarm rate. The feedback rule has three conditions that are based on the false-alarm rate ($P_{fa}$) as listed on Table 1. When the false-alarm rate is larger than 30%, the discrimination factor increases 3-unit steps and the retraining command is set to retrain PCs and clusters. When the false-alarm rate is equal to 0% and the previous false-alarm rate is also 0%, there is no need to give adjustment. However, if the false-alarm rate increases from 0% to higher, the discrimination factor increases 1 step, and vice versa. When the false-alarm rate comes between 15% and 30%, the discrimination factor increases 3-unit steps. When the false-alarm rate is under 15%, the adjustment is based on the performance change after previous adjustment. If the performance change is improvement, the new adjustment is the same as the previous adjustment, and if is degradation, the new adjustment is the opposite of the previous adjustment. The feedback command is ordered per 100 spikes for every channel.

Each simulated data set is simulated at four different noise levels with varied F-factor which is used for second-stage discrimination. Per the results, the detection rate ($P_d$) for every data set decreases very slightly as the F-factor increases, while the false-alarm rate ($P_{fa}$) decreases obviously to zero as the F-factor increases. The high $P_d$ appears from noise level 0.05 to noise level 0.15 and the detection accuracy is approximately beyond 90% for each data sets. At noise level 0.2, the $P_d$ is down to the range from 50% to 80%. The $P_{fa}$ for each data sets approaches to zero when the F-factor is more than fifteen.

3. RESULTS

The simulation results in this work are all evaluated by the Matlab and includes: spike detection for evaluation of two-stage spike detection, feature extraction for comparison between the $P_{CA}$ with neural signal processor and neural spike sorter in real-time. The typical feedback method is to increase or decrease the threshold depending on the performance impact of previous adjustments [40]. In the real-time experiment, it is difficult to directly measure the false-alarm rate ($P_{fa}$) and, therefore, the percentage of unclassified event on the spike sorter is treated as the false-alarm rate. The feedback rule has three conditions that are based on the false-alarm rate ($P_{fa}$) as listed on Table 1. When the false-alarm rate is larger than 30%, the discrimination factor increases 3-unit steps and the retraining command is set to retrain PCs and clusters. When the false-alarm rate is equal to 0% and the previous false-alarm rate is also 0%, there is no need to give adjustment. However, if the false-alarm rate increases from 0% to higher, the discrimination factor increases 1 step, and vice versa. When the false-alarm rate comes between 15% and 30%, the discrimination factor increases 3-unit steps. When the false-alarm rate is under 15%, the adjustment is based on the performance change after previous adjustment. If the performance change is improvement, the new adjustment is the same as the previous adjustment, and if is degradation, the new adjustment is the opposite of the previous adjustment. The feedback command is ordered per 100 spikes for every channel.
The number of training data for generating covariance matrix is related to the consumption time for waiting the valid output. For instance, if the number of training data is 1000 and the firing rate of a set of neural signals is 20 Hz, then the consumption time of each channel for getting the valid output of the neural signal processor is about 50 sec. However, it seems not necessary to use such large amount of training data to derive covariance matrix if less one could get the same objective. Thus, the determination of the training number is evaluated by the correlation coefficient between the first principal component (PC1) derived from each different training number and PC1 derived by the training number 1000 as shown in Fig. 5. The training number for the simulation ranges from 10 to 1000 increasing every 10 steps and four different data sets at noise level 0.2 are utilized. The simulated data for each training data number has ten sets derived from corresponding data set. Note that the upper and lower error bar is calculated by the measurement of standard deviation of ten simulated data sets. The number of training data could be chosen from 110 to 170 or more from the result. The less training number indicates the fewer consumption time and the training number with order of 2 is beneficial to digital system. Therefore, the training number for covariance matrix is chosen as 128 in this work.

4. CONCLUSION

In this study, a framework of automatic and real-time spike sorting for multichannel neuronal activity recording based on hardware-software co-design is presented. The spike detection and feature extraction of the neural signal processor on this system is implemented in hardware design, and the clustering and classification of the neural spike sorter is designed in software. The two-stage spike detection, combining the benefit of threshold method and nonlinear energy operator, is presented as the initial step of the spike sorting process. The adaptive threshold and the feedback rule are utilized for
adapтивно регулируя порог и дискриминационный фактор с флуктуирующими окружающим. Результаты показывают, что метод двухступенчатого детектирования синаптических импульсов обеспечивает высокую точность при уровне помех менее 0.2 и позволяет эффективно снизить уровень ложных срабатываний по сравнению с классическим методом.

Как показано на рис. 4, метод двухступенчатого детектирования мог бы выделить все синаптические импульсы для всех наборов данных при уровне шума 0.05, 0.1, и 0.15, а точность детектирования бессильна быть выше чем 90% в среднем для этих случаев. Относительная ошибка детектирования (реальный импульс, но не зарегистрированный) зависит от плотности шума и может быть пропущенным при низком значении порога. Однако, если пороги были установлены слишком низки для вычисления влияния шума на ближайшее окружение, и некоторый импульс с низкой амплитудой мог бы быть пропущен при первой ступени детектирования.

В целом, метод двухступенчатого детектирования может быть использован для реального времени сортировки импульсов и управления параметрами системы на основе изменений окружающей среды.

**REFERENCES**


