Proceedings of the 5th International
IEEE EMBS Conference on Neural Engineering
Cancun, Mexico, April 27 - May 1, 2011

Density-based Hardware-oriented Classification for Spike Sorting Microsystems

Li-Fang Cheng[1], Tung-Chien Chen[2], Nai-Fu Chang[2], and Liang-Gee Chen[1,2]

[1]Department of Electrical Engineering and [2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan

***Abstract*— Successful proof-of-concept laboratory experiments on cortically-controlled brain computer interface motivate continued development for neural prosthetic microsystems (NPMs). One of the research directions is to realize realtime spike sorting processors (SSPs) on the NPM. The SSP detects the spikes, extracts the features, and then performs the classification algorithm in realtime in order to differentiate the spikes for the different firing neurons. Several architectures have been designed for the spike detection and feature extraction. However, the classification hardware is missing. To complete the SSP, a density-based hardware-oriented classification algorithm is proposed for hardware implementation. The traditional classification algorithms require a considerable memory space to store all the training features during the processing iteration, which results in a considerable power and area for the hardware. The proposed one is designed based on the density map of the spike features. The density map can be accumulated on-line with the coming of the spike features. Therefore the algorithm can save significant memory space, and is good for efficient hardware implementation.**

I. Introduction

Spike sorting is an important tool for analyzing neural signals in the realm of neuroscience. It aims to sort the detected neural events, or spikes, to the corresponding sources of the firing neurons. With the aid of accurate sorting results, the performance of the cortically-controlled brain-machine interface for paralyzed patients may be improved [1]. One of the current research targets is to design a real-time spike sorting processor on the neural recording microsystems for the long-term experiments [2]. The power consumption and the size of the device are two of the major concerns for the hardware optimization.

The on-chip spike sorting system generally consists of four stages: the spike detection, the filtering and alignment, the feature extraction and the classification. Each stage of the system can be divided into the on-line processing engine and the algorithm training engine. The algorithm training engine collects a considerable amount of neural signals and then extract the algorithm parameters used for the on-line processing engine. For example, an amplitude threshold should be trained in order to accurately detect the spikes in the on-line spike detection engine. Many on-line processing hardware units for spike sorting have been proposed [3], [4]. The principal component analysis, one of the training algorithms for the spike feature extraction, has also been designed in VLSI hardware [5]. However, the hardware design for the training part of classification has not yet been provided in the previous works.

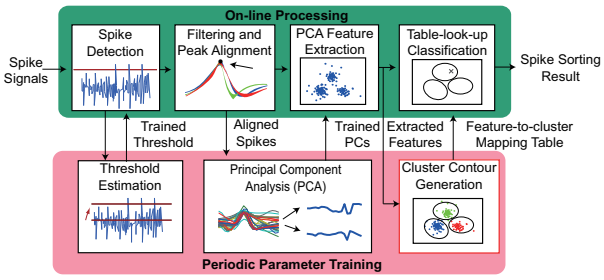


Fig. 1. The hardware structure of spike sorting.

In this paper, we propose a hardware-oriented training algorithm for the classification part of spike sorting. Starting from two of the most often discussed clustering algorithms, the k-means and mean shift algorithms, we try to construct a procedure to improve the accuracy and reduce the amount of computation and memory requirements for the future hardware implementation. The remainder of this paper is organized as follows. Some preliminary knowledge is described in Section II. The k-means and mean shift algorithms are introduced and discussed in Section III. The proposed algorithm for classification is represented in Section IV. Section V shows the simulation results, and Section VI gives the conclusion.

II. Preliminary

A. Spike Sorting

Neurons in the brain communicate through the firing of action potentials, or spikes. These spikes can be recorded extracellularly by the implantation of the micro-electrodes into the brain. Most research in neuroscience relies on the analysis of these measured signals. However, the measured signals are often composed of spikes from a group of close-by neurons, and how to identify the signals from different neurons becomes an important issue. Spike sorting is the process that provides a sequence of procedures to classify the spikes into clusters corresponding to different neurons.

The first step of spike sorting is the spike detection. For most neurons, the most prominent way to identify a spike is to detect its amplitude [6]. Therefore, an amplitude threshold related to the local instantaneous energy is often used for the detection. The next step is to interpolate and align the spike waveforms to facilitate the extraction of the spike features. The principal component analysis (PCA) is one of the widely used tools to extract the spike features and project them onto a finite-dimension feature space. Finally, the spikes on the feature space are classified into clusters corresponding to different neurons after the classification algorithms.

978-1-4244-4141-9/11/$25.00 ©2011 IEEE

B. Training For Classification and Design Requirements

Figure 1 shows a general architecture of hardware operation for spike sorting. The raw neural signals after the amplification and digitization are passed to the spike sorting processor. The processor is consisted of four major stages: the spike detection, the filtering and alignment, the feature extraction, and the classification. Each stage can be further divided into two parts—the on-line processing and the training parts as shown in Fig. 1. The on-line processing engines deal with the sequential input signals in a real time with the programmed algorithm parameters such as the amplitude threshold for the spike detection. The training engines collect a large amount of the neural data and extract the algorithm parameters through the algorithm training. In this paper, we will focus on the training part of the classification stage for the spike sorting processor.

As Fig. 1 shows, the training engine for classification takes the results of feature extraction as the input, and outputs a mapping information such as a feature-to-cluster table. Afterwards the mapping information is returned to the on-line processing engine to classify the following detected spikes in realtime. After the training, the new incoming spikes can thus be classified by simply looking up the feature-to-cluster table. We will discuss the training algorithms of clustering in more detail in the next section.

Power and area are two major concerns for the implementation of the spike sorting processor. The power consumption is often proportional to the operation frequency, and the area is related to the processing memory units. Therefore, the power consumption as well as the chip area of a device can be roughly estimated by the amount of computation and memory requirement on the software-design level. For example, the memory size should be considered for the training data that may need to be stored during the whole training procedure.

III. K-means and Mean Shift Algorithms

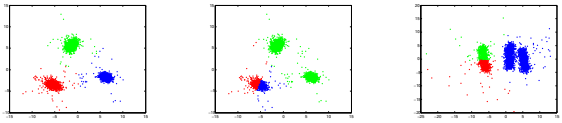
The k-means and mean shift algorithms have been used for the classification in the off-line spike sorting [6], [7]. In this section, we will briefly introduce the two algorithms and discuss the hardware implementation issues based on these two algorithms.

A. K-means Clustering

K-means algorithm divides a feature space into k clusters by minimizing the sum of distances between the feature points and the corresponding cluster center. Given an algorithm parameter of k, the algorithm first sets k initial centers. Then each feature point is assigned to one center with the minimum distance. After the assignment, the new centers are set to the mean values of feature points within the clusters. The feature points are then re-clustered according to the new centers. The iteration is performed until the convergence conditions are met.

B. Mean shift Clustering

Mean shift clustering adopts the concept of local kernel density estimation to cluster the feature space. Given a window



(a) (b) (c)

Fig. 2. Comparisons of k-means clustering.

of size h and its center x, the vector that points toward the local maximum is calculated by

$$m_{h,G}(x) = \frac{\sum_{i=1}^{n} x_i g(\left\| \frac{x-x_i}{h} \right\|^2)}{\sum_{i=1}^{n} g(\left\| \frac{x-x_i}{h} \right\|^2)} - x$$

, which is the so called mean shift vector [8]. The $g(x)$ is the profile function for a radially symmetric kernel $G(x)$ and works as the density estimator [8]. It computes the kernel weights for data points according to their distance to the center. Through calculating the mean shift vector iteratively, the window keeps evolving toward the local maximum. When the local maximum is found, the data points visited by the window during this iteration are grouped into the same cluster [8]. The procedure usually starts from many points of the feature space until most of the feature points are clustered.

C. Discussion

The k-means algorithm has relatively small amount of computations than the mean shift algorithm. However, the performance of k-means algorithm is not robust. One of the factors that may influence the performance is the initial setting of the clusters. Figure 2 (a) and (b) show two different clustering results of the same neural sequence with the k-means algorithm. The result may be trapped into a wrong local maximum without a proper setting of the initial clusters. More, the k-means algorithm may also fail when the clusters are not in the elliptic shapes. Figure 2 (c) shows an example. In contrast, the mean shift clustering often delineates more accurate boundaries because of the adaption of the density estimation, which conceptually coincides with the intuitive way how human distinguishes different clusters on the feature space. Therefore, the density estimation may be a good starting point to design a hardware-oriented classification algorithm in order to have a robust result.

As for the hardware mapping, the memory requirement and the computation complexity are usually estimated on the algorithm level in order to achieve a small and low power design. The k-means and mean shift algorithms keep using the features of the training spikes during the iteration process, and a considerable storage space is required to store all these training data. The large memory generally leads to large area and power consumption. In addition, the mean shift algorithm estimates the local spike density on the feature space and computes point-to-point distance of mean shift vector during the iteration, which results in a significant computation compared with the k-means algorithm. To have a hardware-friendly algorithm, the requirement of the large memory to store all the training data should be released. The computation during the iteration should be simplified as well. Note that the classification accuracy should be maintained after the consideration of all the above hardware issues.

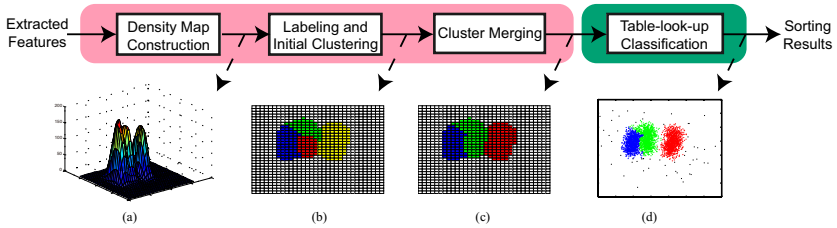


Fig. 3. The block diagram of the proposed procedure.

IV. Proposed Hardware-oriented Algorithm

A. Algorithm Procedure

Figure 3 shows the overall procedure of the proposed density-based hardware-oriented classification algorithm. In order to achieve a good classification performance like the mean shift algorithm, the concept of density estimation is adopted in the proposed algorithm. The first step of the algorithm is to create the density information of the detected spikes based on the extracted features. In the density estimation, each feature dimension is firstly quantized into N different levels, and the d-dimension feature space can thus be divided into N^d units, or cells. The N^d cells are used to store the density information of the detected spikes on the feature space. Similar to the mean shift algorithm, a discrete kernel function is used to formulate the density distribution, and the spike density is estimated by accumulating this kernel on the cells. If a spike is detected, the kernel is added to the specific cells corresponding to the regions of feature space of the detected spikes. After all the training spikes are processed, a density map like Fig. 3 (a) is constructed. Note that the approximated discrete Gaussian kernel is finally used in this paper.

The second step is the labeling and initial clustering of the cells. In this step a label is given to the cells on and around a local maximum of the density. As the result, the entire density map are divided into several clusters corresponding to the local peaks of the density. The detailed operations are described as follows. In one iteration, we starting from an arbitrary unvisited cell and label it with a cluster number. Afterwards we check the neighbor cell according to the shift vector of the current cell. If the next cell is also unvisited, it is labeled to the same cluster number. Otherwise if the next cell is previously labeled with another cluster number, this label is used to replace the label of the previously visited cells in this iteration. The iteration is terminated when the two labels are merged or the local maxima is found. Note that the shift vector of one cell indicates the direction to the adjacent cell with the highest density value. That means the label of the cluster is assigned along the highest density gradient as the mean shift algorithm. Through the iteration process, the feature space is initially clustered after all the cells are visited. Figure 3 (b) shows the result of the initial clustering.

After all the cells in the feature space are labeled after the initial clustering, the final clusters are constructed by merging the initial clusters according to the boundary conditions of the pairs of the clusters. In this step, a score is given to each boundary to represent the probability that the two clusters beside the boundary are actually from the same firing neuron. Some conditions are used for the evaluation. For example, we may check the density of the cells along the boundary of the two clusters. If the value of the density along the boundary is close to the peak value of two clusters, the two clusters may be merged into one cluster. If the size of one cluster is much smaller, it has a higher probability to merge with other large clusters. Figure 3 (c) shows the cell-to-cluster table which is the final result of the training procedure for classification.

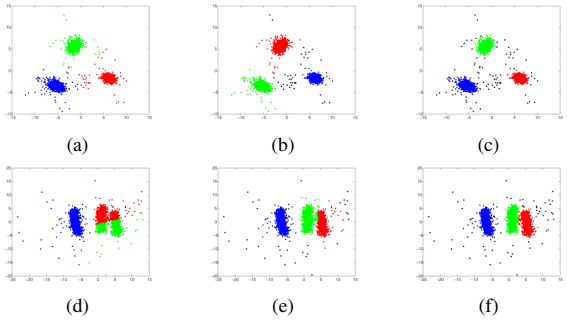


Fig. 4. The comparison of the different classification algorithms for spike sorting. (a) and (d) are for the k-means algorithm. (b) and (e) are for the mean shift algorithm. (c) and (f) are for the proposed hardware-oriented density-based algorithm.

Finally, Fig. 3 (d) represents the re-mapping result by the usage of the table during the on-line processing. After the training, the cell-to-cluster table is sent to the on-line engine for the spike classification. In the on-line engine, the neural signals are processed in a real time, and one spike is classified immediately by the table-look-up according to the extracted features.

B. Discussion about the Memory and Computation Issues

The memory requirement and computation complexity are two primary issues to design a hardware-oriented algorithm. The k-means and mean shift algorithms require to store the features of all the spikes during the entire algorithm training. Significant memory space may thus be required. The proposed algorithm is designed based on the density map of the spikes. After the Gaussian kernel is accumulated on the density map in the first step of Fig. 3, the input features of the spikes can be discarded immediately. The procedure can be operated on-the-fly along with the coming of the spikes. Therefore, only the memory space for the density map is required and should be much smaller than the memory space for the raw data of the spike features.

TABLE I

OBJECTIVE COMPARISON OF CLASSIFICATION ACCURACY

	K-means	Mean shift	Proposed
Case 1	98.33%	98.71%	97.64%
Case 2	75.32%	96.18%	96.97%

TABLE II

COMPARISONS OF MEMORY AND COMPUTATION

	memory (bits)	addition	multiplication
K-means	83,250	69,560	52,170
Mean shift	83,250	3,466,244	1,728,665
Proposed	20,480	178,660	6,697

The mean shift algorithm usually has a better performance but consumes larger computation compared with the k-means algorithm. That is because the mean shift algorithm needs to estimate the local spike density based on the kernel function many times during the iteration. The proposed algorithm estimate the density function globally for only one time in the beginning. During the iteration, the operations is to calculate the shift vectors, and performs simply the comparisons between the densities of adjacent cells. A related smaller computation can be expected. .

C. Accuracy and Memory Tradeoff

The proposed algorithm is based on the density map, and a memory space is required for the cell units. There is a trade-off between the classification accuracy and the chip size regarding this memory. The memory space is related to the number of N. When N becomes smaller, the amount of the memory space is reduced. However, the density map also has a poorer resolution, which may result in the degradation of the classification accuracy. For the hardware optimization, the simulation is required to decide the resolution of the density map and the corresponding memory space.

V. SIMULATION RESULTS

Figure 4 shows the visual comparison between the k-means, mean shift, and the proposed algorithms. For the first neural sequence in Fig. 4 (a)–(c), all the three algorithms work successfully. In the second case in Fig. 4 (d)–(f), the k-means algorithm fails to give a reasonable result as shown in Fig. 4 (d). The proposed algorithm works as well as the mean shift algorithm but has smaller computation and memory requirement as shown in Fig. 4 (e) and (f). Table I summarizes the corresponding objective comparison of the classification accuracy with the golden results. The case 1 is referred to the test case shown in Fig. 4 (a)–(c) while the case 2 is for Fig. 4 (d)–(f).

For the hardware optimization, the memory requirement and the computation complexity are needed to be considered during the algorithm development. Table II summarizes the estimation of the required memory and computation for the three algorithms. The number of the spikes used in the algorithm training is about 3300, and the density map with 32×32 cells is used in this comparison. We can observe that the storage space is significantly reduced for the proposed algorithm. The amount of the computation is much smaller in comparison with the mean shift algorithm. An efficient implementation of VLSI hardware can be achieved with the smaller power and area consumption with the proposed algorithm.

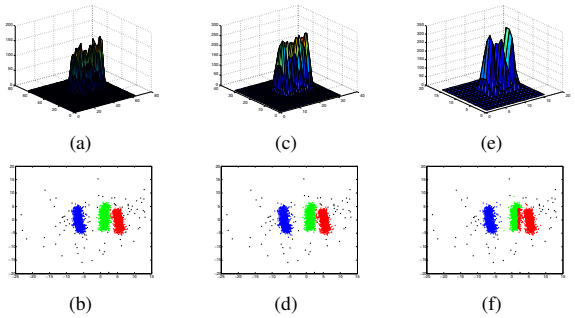


Fig. 5. Simulation results with different resolutions of the density map.

There is a trade-off between the accuracy and the chip area regarding the memory size of the density map as we discussed in Section IV-C. Figure 5 represents the performance comparisons of the proposed algorithm with the different resolutions of the density map. Figure 5 (a) and (b) are tested with the resolution of 64×64, and the feature space is divided into 4096 cells for the density construction. Figure 5 (c) and (d) are tested with the resolution of 32×32 and Fig. 5 (e) and (f) are with 16×16. As Fig. 5 (a)–(d) shows, either 64×64 or 32×32 works successfully, but the 16×16 division fails to give a proper boundary as shown in Fig. 5 (f).

VI. CONCLUSION

In this paper, a density-based hardware-oriented algorithm is proposed for spike sorting microsystems. By the quantization of the feature space and the construction of the density map, the memory space and computation complexity for the training iteration are greatly reduced in comparison with the k-means and mean shift algorithms. The classification accuracy is as robust as the mean shift algorithm according to the simulation. The algorithm is thus feasible for the efficient hardware implementation in the future.

REFERENCES

[1] M.D. Linderman and et al., "Signal processing challenges for neural prostheses," *IEEE Signal Proc. Magazine*, vol. 25, no. 1, pp. 18–28, 2008.

[2] Z. Zumsteg and et al., "Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems," *IEEE Tran. on Neural Syst. and Rehab. Eng.*, vol. 13, no. 3, pp. 272–279, 2005.

[3] M. Chae and et al., "A 128-channel 6mw wireless neural recording ic with on-the-fly spike sorting and uwb transmitter," in *ISSCC Dig. Tech. Papers*, Feb 2008, pp. 146–603.

[4] V. Karkare and et al., "A 130-gw, 64-channel spike-sorting dsp chip," in *ASSCC Dig. Tech. Papers*, 2009, pp. 289–292.

[5] T. C. Chen and et al., "A biomedical multiprocessor soc for closed-loop neuroprosthetic applications," in *ISSCC Dig. Tech. Papers*, Feb. 2009, vol. 25, pp. 434–435.

[6] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Comput. Neural Syst.*, pp. R53–R78, 1998.

[7] Q. Zhao and et al., "Evolving mean shift with adaptive bandwidth: A fast and noise robust approach," in *Asian Conf. on Computer Vision*, Sept. 2009, vol. 5994, pp. 258–268.

[8] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.