ALGORITHM AND IMPLEMENTATION OF MULTI-CHANNEL SPIKE SORTING USING GPU IN A HOME-CARE SURVEILLANCE SYSTEM

Yun-Yu Chen, Yi-Min Tsai, and Liang-Gee Chen

DSP/IC Design Lab., Graduate Institute of Electronics Engineering,
National Taiwan University, Taipei, Taiwan
{littlecorn,ymtsai,lgchen}@video.ee.ntu.edu.tw

ABSTRACT

Intensive home-care surveillance programs are associated with a marked decrease in the need for hospitalization. They can improve the functional statuses of elderly patients with severe congestive diseases. The GPU-based home-care surveillance system is effective and has a major impact on health expenditure than traditional surveillance equipments. In this work, we propose a spike sorting technique as a specific case for the GPU-based home surveillance system. Spike sorting is the procedure of classifying spikes corresponding to the firing neurons. In neuroscience research, spike sorting is adopted to analyze neural activities, brain functions and sensation. It is also a key component in cortically-controlled neuroprosthetics for patients. In order to efficiently distinguish different neural spike activities, a robust spike sorting algorithm is required for above applications. To improve accuracy, multi-channel spike sorting is necessary. In addition, real-time monitoring for a home-care system is required. Therefore, we exploit a CUDA implementation using GPU for acceleration.

Index Terms— Spike sorting, home-care surveillance system, CUDA

1. INTRODUCTION

The home-care surveillance system is the current trend of modern health care treatment. Surveillance treatment can replace traditional hospitals or day-care centers. According to [1], it not only results in a significant decrease in recurrent hospitalizations for cardiovascular complications, but also reduces the enormous socioeconomic burden of medical care with these patients. As worldwide healthcare cost-containment escalates, it becomes crucial to develop new strategies that are cost-effective and that improve the care quality. The existing home-care systems mainly consist of vision surveillance equipments. However, some disease symptoms cannot be seen on the video, including pain, the decubitus injury of physical pressure, mental illness, etc.

Fig. 1 shows the operations of a home-care surveillance system. First, the analog front-end recording devices configured by physicians amplify, digitize and record the bio-signal. Second, the collected data are transmitted from the recording devices to the GPU device through a RF transmitter to do data processing and computing. Last, when danger is detected, computer alarms the reaction device and sends the data to hospitals for clinical treatment.

This project was supported by the GUP Programming course hosted by National Taiwan University. The authors would like to thank Prof. Wei-Chao Chen for advising the project and the valuable comments.

In this paper, we exploit a home-care surveillance system via introducing a multi-channel spike sorting technique with a GPU data-computing flow. We propose a spike sorting algorithm for home-care surveillance systems that contact hospitals when the damage is detected from the results of neural decoding, such as perilous action, injured sensation and abnormal neuronal activity.

Spikes are the signals transmitted between neurons in a form of electronic action potential. Through extracellular electrode arrays, each channel records neural signals from multiple neurons. Since the knowledge in the firing pattern of neural signals from individual neuron is highly desirable, the identification of each neuron from the recorded spike trains is necessary. This process is commonly referred as spike sorting. Currently, spike sorting is mainly used in the research of neuroscience and cortical control prosthesis. It is essential for studying neural activities and sensation in neuroscience research. It provides brain functions that communicate with outside world through cortical controlled prosthesis by spinal cord injured patients and those with the Parkinson disease or epilepsy. Once the spikes are accurately classified, neural modeling and decoding can interpret groups of spikes into control commands to prosthesis or actuators. Since the results of the neural decoding are less significant without an accurate spike sorting, robust sorting performance is a critical issue [2]. Besides, home-care surveillance systems require real-time processing capabilities to react to patients' symptoms or motion intention in time.

Computation time and accuracy are the critical issues for off-site spike sorters. Since the classification of spikes depends on the features extracted from the spike waveforms, a better sorting performance usually comes with a higher sampling rate. We introduce cubic spline interpolation to reconstruct waveforms with high sampling rate before feature extraction. However, interpolation increases complexity. We accelerate the sorting procedure through GPU.

Fig. 2 shows a multi-channel spike sorting system. After amplifying and converting the recorded signals from analog to digital, the digital bio-signal processing technique is applied to find and classify the spikes. In this paper, we implement off-site spike sorters with spike detection, interpolation, feature extraction on GPU, and with classification on CPU. A better sorting performance with high signal resolution could be achieved.

The remainder of this paper is organized as follows. In Section 2, the algorithms used in the off-site spike sorter for each spike sorting steps are introduced. Section 3 states the CUDA implementation in detail. Section 4 demonstrates the improvement on processing time without degrading the performance on a GPU platform. Section 5 concludes this work.

978-1-61284-350-6/11/$26.00 ©2011 IEEE

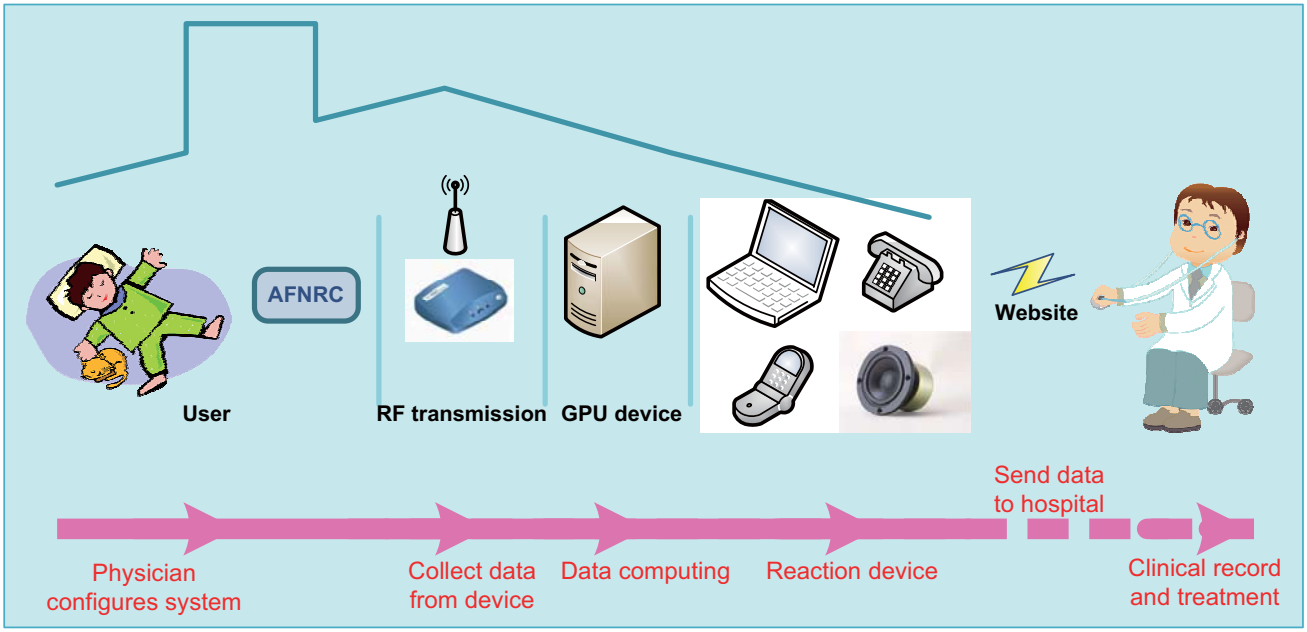


Fig. 1. Home-care surveillance system based on GPU computing.

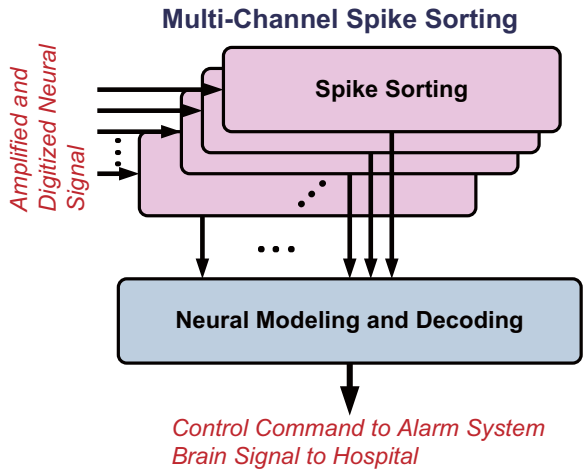


Fig. 2. Multi-channel spike sorting system.

.

2. PROPOSED ALGORITHM

The main algorithm flow for single spike sorting is shown in Fig. 3. Spike sorting generally consists of four main parts: the spike detection to allocate the spike events, the spike interpolation and alignment to enhance the signal resolution and to improve accuracy, the feature extraction to compute the dominant features and to perform the dimension reduction, and the clustering to classify the detected spikes into groups according to the extracted features.

2.1. Spike Detection

In the previous state-of-the-art designs [3, 4], the spike detection is implemented by simple comparison to a voltage threshold. However, this spike detection algorithm may result in many miss detections or false alarms, because the threshold value is critical and hard to define. Therefore, another algorithm called nonlinear energy operator (NEO), first characterized by Kaiser [5], is suggested by [6]. The NEO is composed of nonlinear energy operation and adaptive thresholding. The first step, nonlinear operation, is processed according to Eq. 1,

$$Y[n] = x[n]^2 - x[n+p] \times x[n-p] \quad (1)$$

where $x[n]$ is the n_{th} input discrete signal sample and $Y[n]$ is the NEO value. p denotes the temporal position offset. If the neural signal varies dramatically, usually occurs around spikes, the NEO value increases. Then a simple comparator-based thresholding is adopted to NEO value. The threshold value depends on data, such as $1.2 \times (standard\ deviation\ of\ input\ signal)$ in our statistical analysis. The NEO algorithm identifies the spikes using the localized instantaneous energy and is robust against the low-frequency noise.

2.2. Spike Interpolation and Alignment

According to [7], sampling skew is one of the main issues that causes the waveform distortion and results in the degradation of the sorting performance. Fig. 4 demonstrates the waveform distortion caused by the sampling skew. Three spikes generated by the same neuron are shown in Fig. 4 (a). However, because of the sampling skew, significant differences between three spikes appear after the waveform alignment along the amplitude peaks (Fig. 4 (b)). The most obvious distortions happens in the neural polarization and depolarization regions (i.e., peak and valley) which are the most significant waveform characteristics used for spike sorting.

Since the variation of spikes is rapid but smooth and continuous, we choose cubic spline interpolation to reconstruct the spike waveforms. Cubic spline interpolation constructs a spike with $n-1$ piecewise third-order cubic polynomials between n sample points

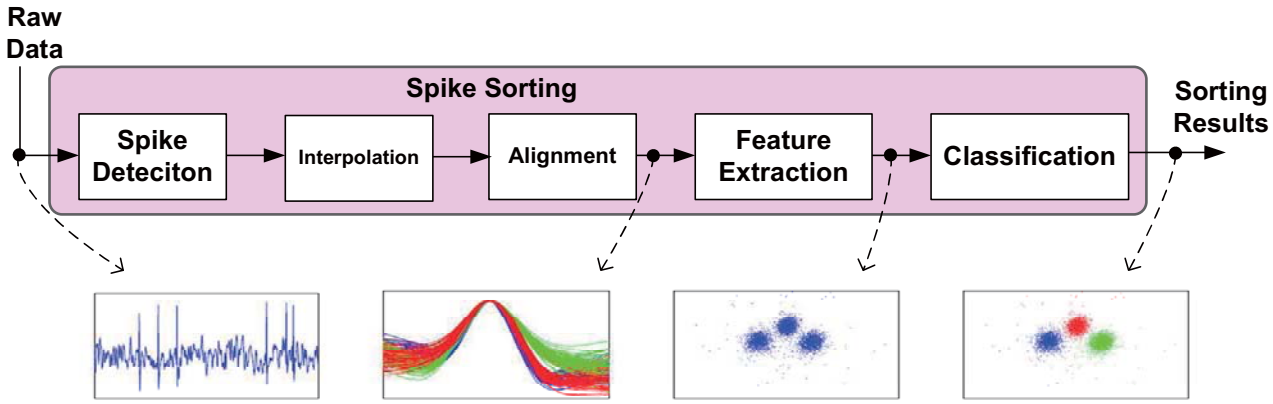


Fig. 3. Single-channel spike sorting algorithm flow.

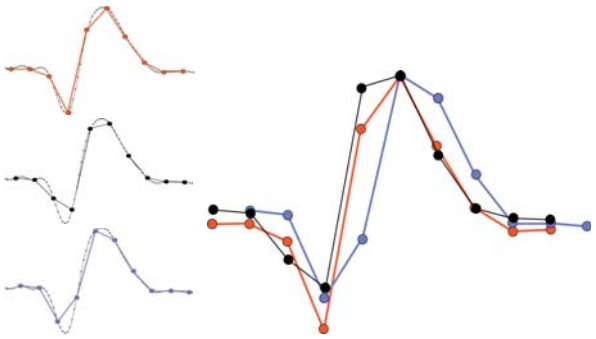


Fig. 4. The waveform distortion caused by the sampling skew. The background waveforms are sampled at the rate of 100K sample per second (sps), while the foregrounds are sampled at 12.5Ksps. (a) Three spikes generated by the same neuron with different sampling time. (b) Significant differences can be seen between three spikes at 12.5Ksps after peak alignment.

$(y_1, y_2, ..., y_n)$. The polynomial curves are represented by,

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \tag{2}$$

where $t \in [0, 1]$ and $i = 1, 2, ..., n-1$. The properties of cubic interpolation are described by Eq. 3,

$$\begin{array}{c} Y_{i-1}(1) = y_i \\ Y_i(0) = y_i \\ Y'_{i-1}(1) = Y'_i(0) \\ Y''_{i-1}(1) = Y''_i(0) \end{array} \tag{3}$$

where Y'_i and Y''_i are the first and second differentiation of i_{th} cubic polynomials. That is, the first and the second differentiation are continuous on source data points. Commonly, the first and second derivative is set to zero as boundary conditions at the endpoints. The differential equation has been simplified into tri-diagonal matrix computation in Eq. 4. The coefficient of i_{th} third-order polynomial can be expressed via Eq. 5. D_i indicates the function's first derivative (i.e., $Y'_i(0)$). Once the polynomial is solved, we interpolate spikes by substituting the equal-division points into Eq. 2. Afterward, the peak alignment is executed for the up-sampled spikes.

$$\begin{bmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & 1 & 4 & 1 & \\ & & & & 1 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \\ \vdots \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} 3(y_1 - y_0) \\ 3(y_2 - y_0) \\ 3(y_3 - y_1) \\ \vdots \\ 3(y_{n-1} - y_{n-3}) \\ 3(y_n - y_{n-2}) \\ 3(y_n - y_{n-1}) \end{bmatrix} \tag{4}$$

$$\begin{array}{c} a_i = y_i \\ b_i = D_i \\ c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1} \\ d_i = 2(y_i - y_{i+1}) + D_i + D_{i+1} \end{array} \tag{5}$$

2.3. Feature Extraction and Classification

As for feature extraction, the most intuitive method would be to take basic characteristics of waveform shapes, such as peak-to-peak amplitudes and timing intervals as features. This simple algorithm has little computation time, but is poor in distinguishing spikes of different clusters especially when the SNR is low. Other algorithms based on principal component analysis (PCA) [8] and discrete Haar wavelet transformation (DWT) [9] are demonstrated with better sorting performance. Haar wavelet transformation is formulated according to Eq. 6 and Eq. 7,

$$W(m) = \frac{1}{(\sqrt{m})} \sum x[t]\psi(t/m) \tag{6}$$

$$\psi(t) = \{ \begin{array}{c} 1, 0 \le t < \frac{1}{2} \\ -1, \frac{1}{2} \le t < 1 \end{array} \tag{7}$$

Fig. 5 shows the example of Haar DWT operation for sequence x. A wavelet-PCA-based feature extraction scheme is then pro-

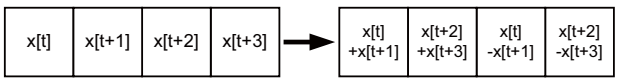


Fig. 5. Haar DWT operation for seq x.

.

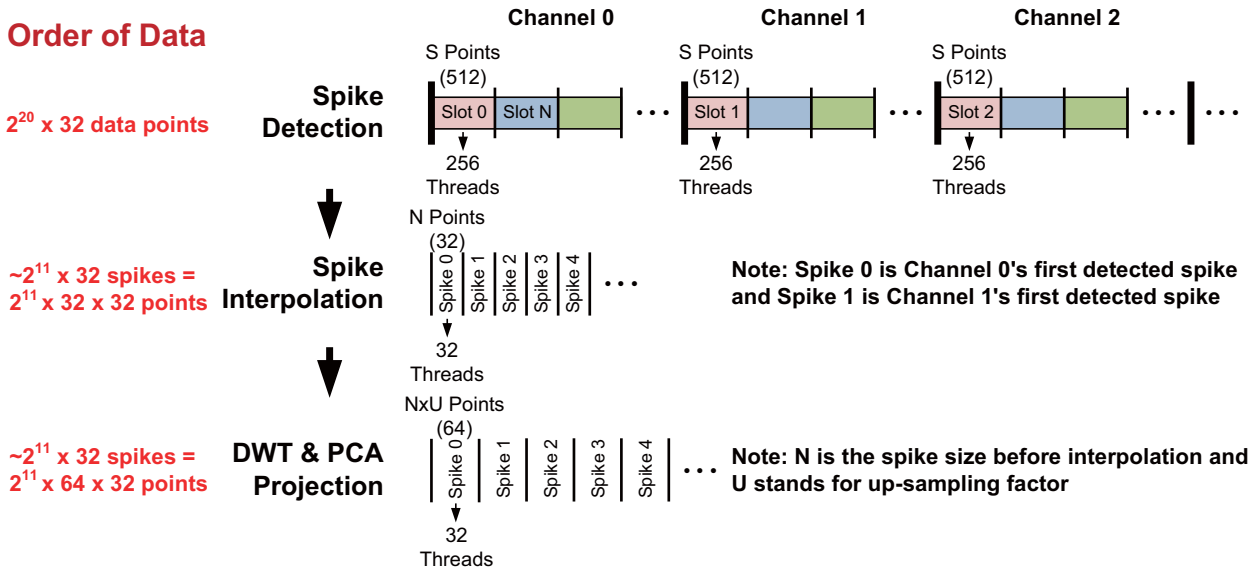


Fig. 6. Resource mapping flow for GPU.

posed. PCA involves the calculation of the eigenvalue decomposition of a data covariance matrix or the singular value decomposition of a data matrix. It is performed after the discrete Haar wavelet transformation to reduce dimensions and to do coefficient selection.

The classification based on the Watershed segmentation algorithm is adopted to sort the spikes in the reduced feature space. As described in [10], Watershed transformation is used to find the catchment basin (i.e., local minima) in the gradient feature space image using the flooding procedure. The catchment basins founded are labeled as different clusters. Since Watershed outperforms other algorithms especially in overlapped clusters and non-circle clusters, it is applied to the spike classification.

In summary, the NEO, cubic spline interpolation and PCA with DWT are chosen because of their robustness against noise and benefits to accuracy. For the spike detection, the NEO operator is first performed. Then, the spike events are detected when the NEO value exceeds a threshold at the peaks of the convex curves of the original neural waveforms. Signal resolution is increased by cubic spline interpolation in order to raise sorting accuracy. For the feature extraction, the DWT is followed by PCA. Finally, we use Watershed segmentation algorithm to classify the clusters.

3. CUDA IMPLEMENTATION

In this section, a GPU-based multi-channel spike sorting design is investigated. We develop a 32 channel spike sorting algorithm on a G80 GPU with CUDA implementation and CPU co-design. Considering the complexity and data parallelism, the Watershed clustering is not suitable for CUDA implementation and therefore is implemented using CPU. We benchmarked the processing time on a Duo-Core 2.8GHz CPU for four modules in our spike sorting algorithm (Fig. 7). The result shows that the spike detection and the spike interpolation costs 27.5% and 57.9% computation time, respectively. For feature extraction, it costs 8.8% computation time for Wavelet calculation and PCA projection. Since the eigenvalues are similar for spike data, they are pre-calculated to reduce complexity. For alignment, only 5.8% computation power is needed. In our experiments, the correct rate of classification drops slightly without alignment module for multi-channel spike sorting. We therefore simplify the overall algorithm. The spike interpolation is directly followed by feature extraction. In conclusion, only NEO spike detection, cubic spline interpolation and two-dimensional PCA projection with Haar wavelet calculation are implemented with CUDA.

For NEO spike detection kernel, it loads S data points, which is called a slot, once at a time from the global memory to the shared memory for processing. Besides, it costs 4 bytes (float type) for each data point. The total number of slots per channel is equal to the total data points per channel divided by S. Hence, the number of blocks depends on these slots. The first slot of each channel is firstly processed. In our case, the one-dimensional block with the size ranging from 8 to 64 is considered. The number of threads is decided considering data point parallelism. T one-dimensional threads are declared to process T data points in parallel. Noting that S should not be less than T. In our case, S is set to 512 and T is set to 256. However, NEO needs to access value of the previous and the next data point. The access of shared memory may cause bank conflicts. Therefore, for a data point at time position K, we also store value of data point $K-p$ and $K+p$ in the shared memory in order to prevent bank conflict problems. If p equals to 1 and S equals to 512, $512\times3\times4=$ 6 KB are required for the shared memory for handling 512 data points. The same idea is also applied to spike interpolation. After spike detection, the order of spike number is much smaller than that of original data points of each channel. The number of blocks depends on the number of spikes detected. We decide the number of threads according to data points of each spike needed to be interpolated. For instance, we interpolate each original spike from 32 points to 64 points, which means one-dimensional thread number is set to 32 for the kernel. The cubic spline interpolation has no data dependency in the interpolation procedure and is suitable for processing in parallel. The intermediate values for interpolation are stored in local memory. As for feature extraction, the one-dimensional thread number is equal to 32 which is identical to the original spike size due to further wavelet calculation. The number of blocks is the same as that in the interpolation procedure. For wavelet calculation, both summa-

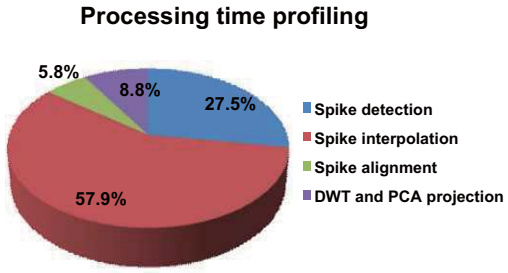


Fig. 7. Time profiling for the proposed spike sorting on a CPU platform.

tion and subtraction operations are executed for two data points set by set in a spike window. The results are reordered to form a Haar wavelet feature. All wavelet waveforms are then reduced to two-dimensional feature vectors using PCA projection, which is enough to maintain classification accuracy. Fig. 6 illustrates the resource mapping flow and the order of data processed in each stage.

4. EXPERIMENTAL RESULTS

The simulation neural signal data comes from the database in [11] from recordings the neocortex of mice. The raw data is recorded with 25K Hz sampling rate and there are about 2^{20} data points per channel originally. After spike detection, around 2^{11} spikes are detected for each channel. For each detected spike, the corresponding spike window is determined as input to the feature extraction module. A spike window consists of $N \times U$ data points according to the original spike size N and the interpolation up-sampling factor U. Hence, about $2^{11} \times N \times U$ data points per channel are processed for Wavelet calculation and PCA projection. In our experiments, we set N to 32 and U to 2. As to Watershed clustering, for each channel, there are 3 kinds of spikes, which suggests 3 classes should be annotated for spike data in the feature domain.

Fig. 8 shows the time comparison between CPU and GPU. We use 32 channels for profiling. For CPU, it needs 3942.8 ms for spike interpolation. However, it costs only 83.8 for the CUDA version and speeds up to 44 times compared to the CPU version. Totally, with CUDA implementation, it speeds up to about 8 times. In addition, we also test another signal data in the order of 2^{19} per channel. The higher the order of signal data is, the more impacts the CUDA-based implementation has.

In the following experiments, we demonstrate the results of each module via CUDA implementation. Two data sets are used for testing (Dataset1 and Dataset2). Fig. 9 shows the NEO spike detection results. If a spike is detected, we set a value one to the corresponding position index. For instance, Fig. 9(a) shows the original signal data and the corresponding detected spikes in the position indices 1331, 1742, 1988 and 2057 with value one. Compared to the CPU version, the detected results are identical but less computational power is needed for the GPU version. The results of cubic spline interpolation using CUDA implementation are shown in Fig. 10. After interpolation, the spike waves become much smoother. Therefore, the interpolated spike contains 64 data points with U being equal to 2. Fig. 11 shows the improvement of neuron cluster separation after interpolation. The original neural signals are sampled at 25Ksps (left). The spike waveforms are re-aligned after up-sampling to 50Ksps through cubic spline interpolation. As observed, the sorting performance significantly increases after interpolation. The PCA

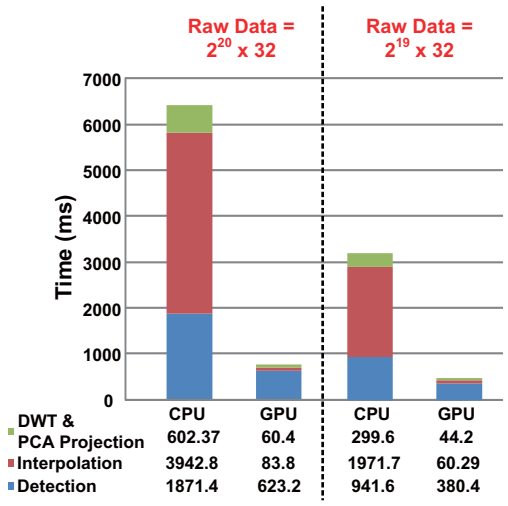
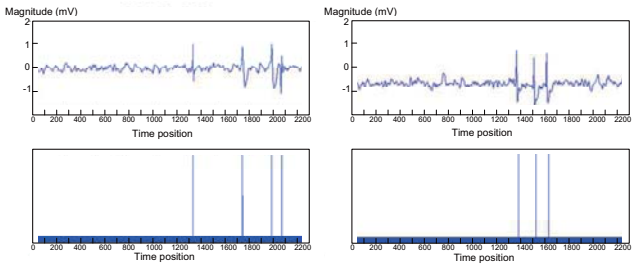


Fig. 8. Time comparison between CPU and GPU implementation. Raw data order equals 2^{20} and 2^{19} with total 32 channels.



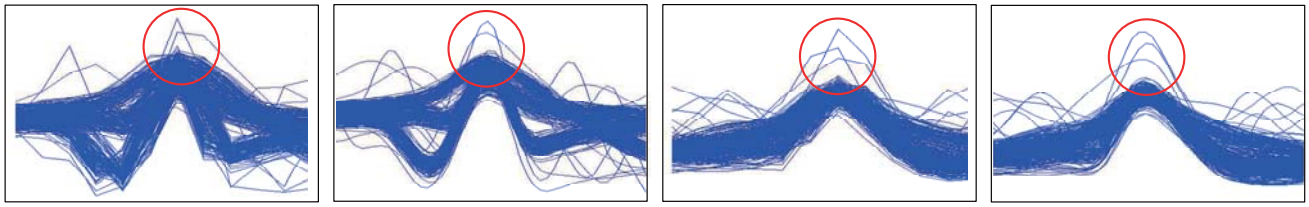
(a) (b)

Fig. 9. NEO spike detection results using GPU (a) Dataset1. (b) Dataset2. The upper row is the original signal and the lower row shows the corresponding detected spikes.

projection is performed after the wavelet transformation. 64 data points are projected to the two principal dimensions by multiplying the corresponding eigen-vectors. Fig. 12(a)(b) show the projected interpolated spikes in the feature space. Finally, we apply CPU-based Watershed clustering on the feature space for classifying the spikes into 3 categories (Fig. 12(c)(d)).

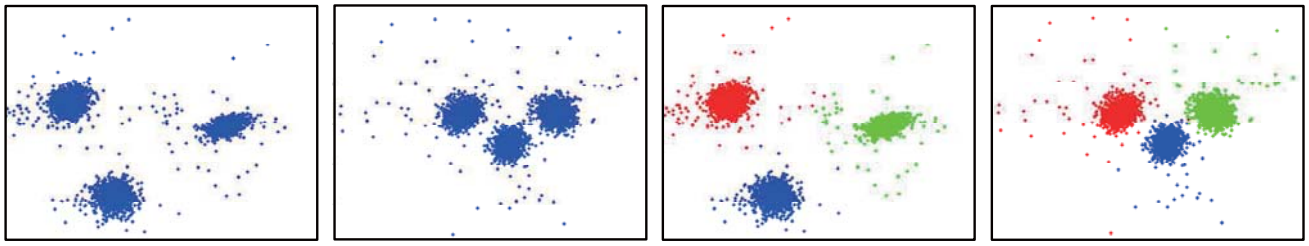
5. CONCLUSION

In this paper, we introduce a spike sorting technique to monitor the non-action symptom as an example for the future home-care surveillance system. Robust performance and real-time monitoring is essential for the proposed system. A multi-channel spike sorting system and the corresponding algorithm flow are proposed. The spike detection, spike interpolation and feature extraction are implemented using GPU in order to accelerate the processing speed. The high system performance is achieved through GPU-CPU co-design. In the future, GPU can be combined into an off-site home-care center system that processes not only for spike sorting but also for other bio-signal processing such as EEG or heart signal. It is obvious that GPU can support powerful computation capability to handle huge amount of data. It is therefore proper to provide real-time monitoring and signal visualization.



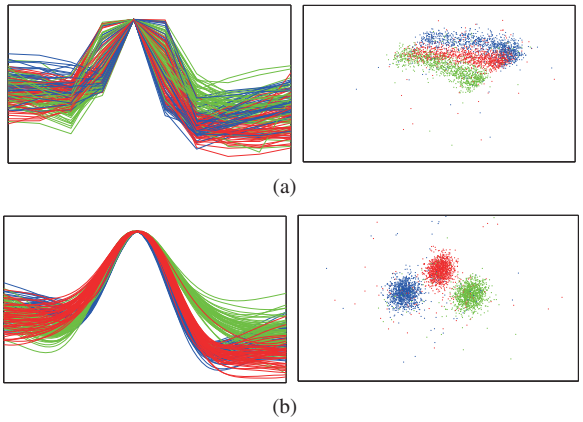
(a) (b) (c) (d)

Fig. 10. Cubic spline interpolation using GPU with interpolation factor 2. (a) Dataset1 before interpolation. (b) Dataset1 after interpolation. (c) Dataset2 before interpolation. (d) Dataset2 after interpolation. The red circles indicate the difference between and after interpolation.



(a) (b) (c) (d)

Fig. 12. Feature extraction using GPU and Watershed classification using CPU. (a) Dataset1 for Wavelet and PCA feature extraction results. (b) Dataset2 for Wavelet and PCA feature extraction results. (c) Dataset1 for clustering. (d) Dataset2 for clustering.



(a)

(b)

Fig. 11. The improvement of neuron cluster separation after the interpolation. The original neural signals are sampled at 25Ksps as shown in (a). For (b), the spike waveforms are re-aligned after upsampling to 50Ksps via interpolation. Since the synthesized neural data from [11] are used, different colors corresponding to different neurons are drawn according to the golden standards (not the classification results).

6. REFERENCES

[1] R. Kornowski, D. Zeeli, M. Averbuch, A. Finkelstein, D. Schwartz, M. Moshkovitz, B. Weinreb, R. Hershkovitz, D. Eyal, and M. Miller, "Intensive home-care surveillance prevents hospitalization and improves morbidity rates among elderly patients with severe congestive heart failure," *Am. Heart J.*, vol. 129, pp. 762–766, Apr. 1995.

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

[3] R. H. Olsson and K. D. Wise, "A three-dimensional neural recording microsystem with implantable data compression circuitry," *IEEE J. Solid State Circuits*, vol. 40, no. 12, pp. 2796V2804, Dec. 2005.

[4] R. R. Harrison and et al., "A low-power integrated circuit for a wireless 100-electrode neural recording system," *IEEE J. Solid State Circuits*, vol. 42, no. 1, pp. 123–133, Feb. 2007.

[5] K. H. Kim and S. J. Kim, "Neural spike sorting under nearly 0-db signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier," *IEEE trans. on Biomed. Eng.*, vol. 47, no. 10, pp. 1406–1411, Oct. 2000.

[6] S. Gibson and et al., "Comparison of spike sorting algorithm for future hardware implementation," *in Proc. Conf. IEEE EMBS*, pp. 5015 –5020, Aug. 2008.

[7] Yun-Yu Chen, Tung-Chien Chen, and Liang-Gee Chen, "Accuracy and power tradeoff in spike sorting microsystems with cubic spline interpolation," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, Jun. 2010, pp. 1508 –1511.

[8] M. Abeles and M. H. Goldstein, "Multispike train analysis," *in Proc. of IEEE*, vol. 65, no. 5, pp. 762V773, May. 1977.

[9] J. C. Letelier and P. P. Weber, "Spike sorting based on discrete wavelet transform coefficients," *J. of Neurosci. Methods*, vol. 101, no. 2, pp. 93V106, Sep. 2000.

[10] D. Wang, "Unsupervised video segmentation based on watersheds and temporal tracking," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, no. 5, pp. 539 –546, sep 1998.

[11] R. Quian Quiroga and et al., "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Comput.*, vol. 16, pp. 1661–1687, Aug. 2004.