31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

Promise of Embedded System with GPU in Artificial Leg Control:
Enabling Time-frequency Feature Extraction from Electromyography

Weijun Xiao, *Student member, IEEE*, He Huang, *Member, IEEE,* Yan Sun, *Member, IEEE, and* Qing Yang, *Senior Member, IEEE*

***Abstract*—Applying electromyographic (EMG) signal pattern recognition to artificial leg control is challenging because leg EMGs are non-stationary. Time-frequency features are suitable for representing non-stationary signals; however, the computational complexity to extract time-frequency features is too high and current embedded systems used for artificial limb control are inadequate for real-time computing. The aim of this study was to quantify the computational speed of a novel embedded system, the Graphic Processor Unit (GPU), on EMG time-frequency feature extraction. The computational time derived from a GPU was compared to that derived from a general purpose CPU. The results indicated that the GPU significantly increased the computational speed. When the size of EMG analysis window was set to 100 ms, the GPU extracted EMG time-frequency features over 50 times faster than the CPU setting. Therefore, high performance GPU shows a great promise for EMG-controlled artificial legs and other medical applications that need high-speed and real-time computation.**

***Index Terms*—Electromyography, control of artificial limbs, embedded system, high performance computing.**

I. Introduction

Electromyographic (EMG) signals recorded from the residual limb of individuals with limb amputations are effective neural control signals for powered artificial limbs [1]. Great success in multifunctional artificial arm control has been realized by pattern recognition (PR) of EMG signals [2-7]. Currently, our research group attempted to apply EMG PR algorithms to artificial leg control. Compared to upper-limb prosthesis control, the difficulties of EMG PR in artificial leg control are two-fold. (1) The EMG signals recorded from the leg are non-stationary and (2) fast system response is required to ensure the user's safety for prosthesis use. To address these challenges, we developed a new, phase-dependent EMG PR strategy to classify the user's locomotion modes [8]. To achieve a prompt time response, four time-domain features [9] in each of 11 EMG channels and a linear discriminant analysis (LDA)-based classifier were applied due to their computational simplicity. The offline testing results [8] showed over 90% classification accuracy, which was promising. However, this accuracy was still not sufficient for the safe use of artificial legs because any error might lead to a fall. A previous study [4] compared the classification performance among different classifiers and EMG features on non-stationary EMG signals offline. The result showed that time-frequency features outperformed time-domain features, while the type of classifier did not significantly influence the classification performance. Therefore, time-frequency features, in lieu of time-domain features, should be applied to non-stationary leg EMGs in order to accurately identify user locomotion mode for neural control of artificial legs.

Compared to time-domain features, the implementation of time-frequency features significantly increases computational complexity because it requires additional time-frequency signal transformation and a calculation for feature dimension reduction [3]. For example, given the number of EMG channels is 11 used in [8] and the EMG sampling rate is 1000 Hz, the total number of multiplication operations is $O(10^5)$ for computing Cohen's class time-frequency representations [10] of 100 ms EMG data, $O(10^{15})$ for feature (a 1.1×10^5-by-1 vector) dimension reduction during the training of a classifier when using principle component analysis (PCA)[11], and $O(10^8)$ for dimension reduction during real-time testing if 1% of the feature dimension is kept. Obviously, the computational speed of current embedded systems used in C-leg® [12] or Rheo Knee [13] are insufficient to identify the user's intention within tens of milliseconds when time-frequency domain features are applied.

In this study, we explored a high performance embedded controller with a built-in Graphic Processor Unit (GPU)[14] for EMG time-frequency feature extraction. A GPU is a single chip processor originally designed for the computation related to 3D graphic rendering. Recently, many researchers and developers have become interested in leveraging the power of GPUs for general-purpose computing. There are two reasons for the explosive research efforts in general purpose GPU computing. First, GPUs can provide an extraordinary speedup for applications that show inherent data parallelism. For example, NVIDIA GeForce 8800 GTX [15] can have 367 GFLOPS peak performance with 128 cores, which is 20-50 times faster than current high-end microprocessors. Secondly, GPUs have been built for commodity PC graphic cards with large volume productions resulting in very low price. Currently a very powerful graphic card cost only a few hundred dollars. The high performance/cost ratio has attracted many researchers and engineers to adopt the GPU for data intensive general purpose computation. Typically, a GPU is a massively parallel machine equipped with multiple cores for concurrent

This work was supported in part by the National Institute on Disability and Rehabilitation Research, U.S. Department of Education (Grant H133F080006) and RI Science and Technology Advisory Council (RIRA-2009-27).

W. Xiao, H. Huang, Y. Sun, and Q. Yang are with the Department of Electrical, Computer, and Biomedical Engineering, Kingston, RI, 02881 USA (corresponding author: H. Huang: 401-874-2385; fax: 401-782-6422; e-mail: huang@ele.uri.edu).

execution of thousands of independent threads. Each core executes the same code on different data sets. Such a Single Instruction Multiple Data (SIMD) architecture is favorable for high-throughput numerical computation. Many numerical problems, such as matrix computation and linear algebra operations, can be easily parallelized into multiple identical subtasks without data dependency, thus a notable performance speedup is achieved on the GPUs [16, 17]. In signal processing, most analysis tools or algorithms require matrix or vector computation that can be naturally tailored to GPUs to achieve high performance that would not have been possible using general purpose microprocessors. To quantify the computational capability of GPUs applied for time-frequency feature-based EMG pattern recognition, we developed an analytical algorithm for one channel EMG signal and implemented it on both the CPU and GPU architectures. We measured the results and compared the computational time for both cases. The results demonstrated that the GPU implementation can provide much faster response time than the CPU. We have observed up to two orders of magnitude performance gain (100 times) as compared to the general purpose CPU, which is promising for real-time EMG PR based on time-frequency feature sets.

II. METHODS

A. EMG Data Collection

This study was conducted with Institutional Review Board (IRB) approval and the informed consent of the recruited subject. One channel of EMG from the rectus femoris was collected from an able-bodied male subject during his level ground walking. An EMG electrode was placed on the belly of the muscle, and the ground electrode was placed on the bony part of the knee. A myomonitor® wireless EMG system (Delsys Inc, MA, USA) was applied to collect the EMG data at a sampling rate of 1000 Hz.

B. Algorithms for Time-Frequency Feature Extraction

The analytical algorithm included two parts: offline training and real-time testing. The steps for feature extraction are shown in Fig. 1. For both training and testing, the time-frequency features were extracted in every analysis window. Next, time-frequency transformation of the EMG signal was conducted. The Hilbert transform [18] was used to convert the real-value EMG signal into the complex-value signal first. The discrete Hilbert transform of a signal x(k) is defined as

$$H(n)=\begin{cases}\frac{2}{N}\sum_{k\ odd} x(k)\cot\frac{\pi}{N}(n-k); n\ even\\ \frac{2}{N}\sum_{k\ even} x(k)\cot\frac{\pi}{N}(n-k); n\ odd\end{cases} \quad (1)$$

where N is the number of samples in an analysis window.

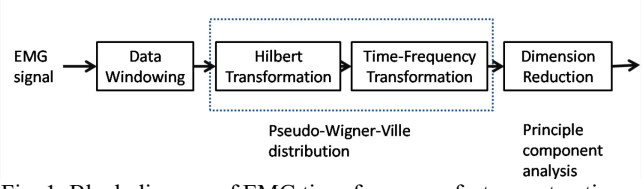

Fig. 1. Block diagram of EMG time-frequency feature extraction.

After obtaining the analytical signal by the Hilbert transform, Smoothed Pseudo-Wigner-Ville Distribution (SPWVD) [19] was used to represent the signal in a time-frequency domain. The calculation formula is defined as

$$SPWVD_x(n,m)=2\sum_{k=-N+1}^{N-1}W_f(k)\times \quad (2)$$
$$[\sum_{p=-M+1}^{M-1}W_t(p)x(n+p+k)x^*(n+p-k)]e^{-j2\pi km/N}$$

where W_f is the smoothing window in the frequency domain with length 2N+1, and W_t is the smoothing window in the time domain with length 2M+1. Then, the resulting time-frequency representation (a matrix) was reorganized into an EMG feature vector (a vector). Due to the high dimensionality of EMG time-frequency feature vectors, a dimension reduction algorithm is necessary to allow efficient classification. Here, we used principle component analysis (PCA) to reduce the dimension of feature vectors. During the offline training, the feature vector in one analysis window was one observation. PCA was conducted on the data matrix, composed of the feature vectors of several observations. The number of principle components was less than 10% of the dimension of feature vectors. The algorithm of PCA involves solving the Eigen values and vectors, which is time-consuming. To enhance the computational speed, we chose the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm to approximate PCA computation [20]. During the testing, the feature dimension was reduced directly by projecting the feature vectors to the principle components obtained in the training procedure. Finally, the feature vectors were fed to a classifier to identify the user's movement intention.

C. Testing Setup and Procedure

The analytical algorithm was implemented on both CPU and GPU architectures. The algorithm was ran on a Dell Dimension 8400 desktop PC equipped with a CPU (Intel Pentium 4 with HT technology, 3GHz) and a GPU (NVIDA 9500GT graphic card, Multi-core parallel processor, 1.4GHz)[15]. The computation times for Hilbert transform, SPWVD, and PCA with different window sizes were measured for both the CPU and GPU settings.

III. RESULTS

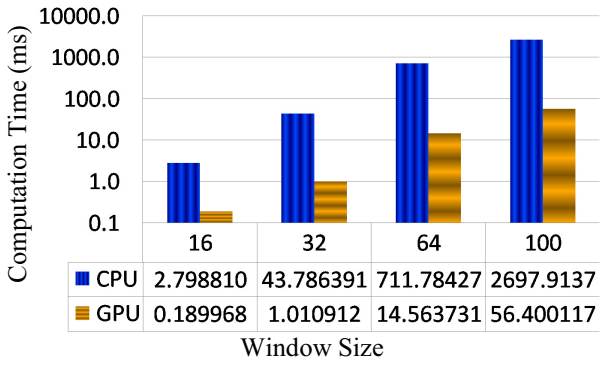

Fig. 2. Computational time of Hilbert transformation.

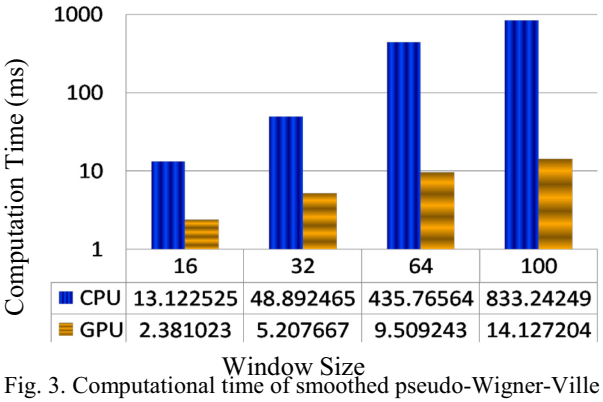

Fig. 3. Computational time of smoothed pseudo-Wigner-Ville distribution.

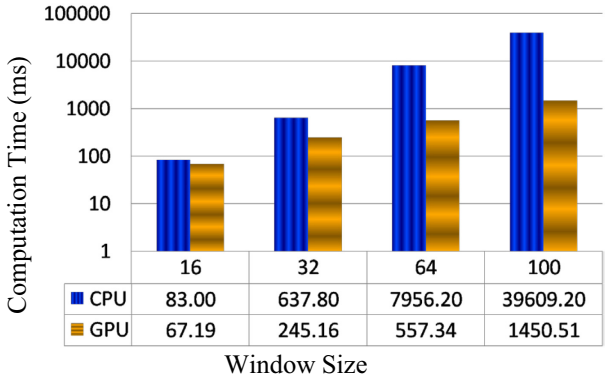

Fig. 4. Computational time of NIPALS to obtain 8 principle components during the training procedure.

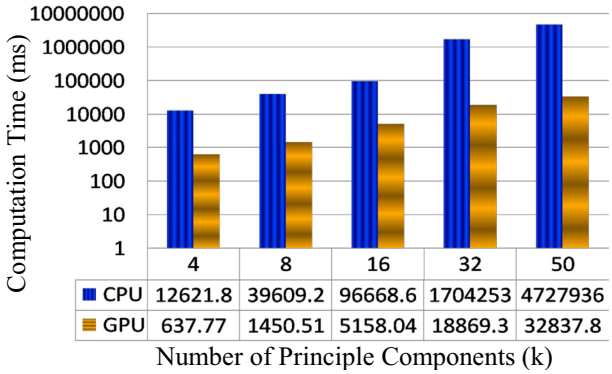

Fig. 5. Computational time of NIPALS when a different number of principle components were selected. The applied window size was 100 ms.

Fig. 2 shows the computational time of the Hilbert transform. The GPU implementation outperformed the CPU for all window sizes. When the applied window was 100 ms, the GPU setting was about 50 times faster than the CPU case.

Similar results for computing smoothed pseudo-Wigner-Ville distribution as those for the Hilbert transform calculation were observed (Fig. 3). Note that we used the latest CUFFT 2.1[21] and FFTW 3.2.1[22] libraries to calculate SPWVD, but we did not optimize the Hilbert transform. Therefore, SPWVD had faster response time than the Hilbert transform in some parameter settings.

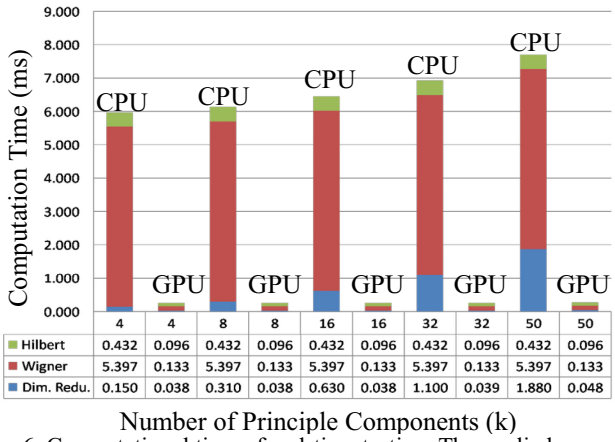

Fig. 6. Computational time of real-time testing. The applied window size was 100 ms.

Fig. 4 shows the time response of the PCA computation when 8 principle components were selected. The GPU produced much faster response than the CPU with up to 30 times performance improvements. Another observation is that the performance speedup of the GPU over the CPU increased as the size of analysis window increased. That is to say, the larger the window size, the more speedup can be obtained using the GPU. This result can be attributed to the better parallelism with a larger data set. Fig. 5 shows the computational time for selecting six different numbers of principle components when the window size was 100 ms. Compared to the CPU, the performance speedup of the GPU was 144 times faster.

Finally, the computational time to extract a feature vector from an analysis window of 100 ms in real-time testing is demonstrated in Fig. 6. The CPU took 7.7 milliseconds to extract time-frequency features, while the GPU setting only needed less than 0.5 ms.

IV. Discussion

From the measured outcomes, we found that the most time was spent on the PCA computation for both the CPU and GPU architectures during the training. Therefore, PCA is the bottleneck of our analytical algorithm. Currently we used the NIPALS method to approximate PCA. This algorithm works well for a small size matrix; however, if the matrix is large, other more efficient algorithms should be considered. One possible optimization is the divide-and-conquer strategy that partitions the single big matrix into multiple smaller matrices. PCA is conducted on each block matrix. Then the global principle components are the combination of all the "block" principle components. In computational terms, it is called data/loop blocking that can significantly improve computational performance due to increased data locality, parallelism, and communication efficiency. Since there is no data dependence among different block matrices, we can perform the PCA for each block matrix simultaneously on the GPU architecture. Our theoretical analysis and preliminary experiment showed that the block PCA has the potential to improve the performance of the PCA. We will address PCA optimization in our future research.

When using feature vectors with 10,000 dimension (derived from 100ms window size) and 16 principle components, the training procedure took around 1.5 seconds for GPU but over 100 seconds for CPU. In the real-time testing, the GPU can extract the features from an analysis window of 100 ms within 0.5 ms, while the CPU needed 7.7 ms to achieve the same computation. The outcomes indicate that if only one EMG channel is used, both CPU and GPU are sufficient for real-time EMG pattern recognition. However, in practice, multiple channels will be considered, which further increases the feature dimensionality and computational complexity. Because the GPU architecture demonstrates up to two orders of magnitude speedup compared to CPU, the GPU has much more potential to handle high-dimension features for real-time control of artificial legs than the CPU.

The limitation of this work is that only one EMG channel was used. Higher computational efforts will be required to perform time-frequency feature extraction when multiple channels are considered. Since there is no data dependency among different channels, the GPU architecture should be more favorable for efficiently computing the analytical algorithm than the CPU. We will address the computational efficiency of the GPU for multi-channel EMG feature extraction in our future work. In addition, the selected analytic algorithm was relatively complicated; other simple methods should be investigated to improve the computational efficiency without deteriorating the classification performance. Also, some prior knowledge of EMG signals can be useful to further reduce the feature dimension before computing PCA.

V. CONCLUSIONS

In this paper, we have presented a design and implementation of a new set of parallel algorithms for time-frequency feature exaction from EMG. Our algorithms have been tested and measured on both a commodity embedded system with a GPU and a general purpose PC. Because of the inherent parallelism that exists in these computations, we are able to effectively parallelize the algorithm on the GPU. Measurement results have shown dramatic speedup of the computation on the GPU system as compared to the CPU architecture. Up to two orders of magnitude speedup have been observed. With continued advancement in performance and decrease in cost, GPUs show a great promise for real time and high speed computation for EMG pattern recognition and eventual application to neural-controlled artificial legs. We are currently working on further optimization of these algorithms.

REFERENCES

[1] J. Basmajian and C. De Luca, "Muscles alive: their functions revealed by electromyography," 5 ed. Baltimore, MD: Williams and Wilkins, 1985.

[2] S. H. Park and S. P. Lee, "EMG pattern recognition based on artificial intelligence techniques," *IEEE Transactions on Rehabilitation Engineering*, vol. 6, pp. 400-405, 1998.

[3] K. Englehart, B. Hudgins, P. A. Parker, and M. Stevenson, "Classification of the myoelectric signal using time-frequency based representations," *Medical Engineering& Physics*, vol. 21, pp. 431-8, 1999.

[4] K. Englehart, B. Hudgins, and P. A. Parker, "A wavelet-based continuous classification scheme for multifunction myoelectric control," *IEEE Transactions on Biomedical Engineering*, vol. 48, pp. 302-311, 2001.

[5] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Transactions on Biomedical Engineering*, vol. 50, pp. 848-54, 2003.

[6] D. Graupe, A. A. Beex, W. J. Monlux, and I. Magnussen, "A multifunctional prosthesis control system based on time series identification of EMG signals using microprocessors," *Bulletin of Prosthetics Research*, vol. 10, pp. 4-16, 1977.

[7] H. Huang, P. Zhou, G. Li, and T. A. Kuiken, "An analysis of EMG electrode configuration for targeted muscle reinnervation based neural machine interface," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, pp. 37-45, 2008.

[8] H. Huang, T. Kuiken, and R. D. Lipschutz, "A strategy for identifying locomotion modes using surface electromyography," *IEEE Transactions on Biomedical Engineering*, vol. 56, pp. 65-73, 2009.

[9] B. Hudgins, P. Parker, and R. Scott, "A new strategy for multifunction myoelectric control," in *IEEE Transactions on Biomedical Engineering*, vol. 40, 1993, pp. 82-94.

[10] L. Cohen, *Time-frequency analysis*. Englewood Cliffs, N.J: Prentice Hall PTR, 1995.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. New York: Wiley, 2001.

[12] I. Otto Bock Orthopedic Industry, *Manual for the 3c100 Otto Bock C-LEG*. Duderstadt, Germany, 1998.

[13] OSSUR, *RHEO KNEE*: http://bionics.ossur.com/Products/RHEO-KNEE/ACT.

[14] "GPGPU.org," *http://www.gpgpu.org*.

[15] NVIDIA Corporation, "CUDA: compute unified device architecture programming guide," http://*www.nividia.com*.

[16] N. Fujimoto, "Faster matrix-vector multiplication on GeForce 8800 GTX," In *the Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2008.

[17] V. Volkov and J. W. Demmel, "Benchmarking GPUs to tune dense linear algebra," In *the* P*roceedings of the 2008 ACM/IEEE Conference on Supercomputing*, Austin, TX, USA, 2008.

[18] S. Kak, "The discrete finite Hilbert transform," *Indian Journal of Pure and Applied Mathematics*, vol. 8, pp. 1385-1390, 1977.

[19] V. Bernasconi, L. Bollea, A. Breda, P. Daponte, G. Maroncelli, and S. Rapuano, " A TFR-based method for the quality assessment of UMTS signals: an application on the first Italian experimental network," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, pp. 485-492, 2004.

[20] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, pp. 37-52.

[21] "NVIDIA Corporation. CUDA: CUFFT library," *http://developer.download.nvidia.com/compute/cuda/2_1/toolkit/docs/CUFFT_Library_2.1.pdf*, 2008.

[22] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," in *IEEE Proceedings of the Special Issue on Program Generation, Optimization, and Platform Adaptation*, vol. 93, pp. 216-231, 2005.