Towards sequential decoding of intramuscular EMG signals: Parallel computing implementation

Tianyi YU^1, Konstantin AKHMADEEV^2, Eric LE CARPENTIER^1, Yannick AOUSTIN^2

Laboratoire des Sciences du Numérique de Nantes, UMR CNRS 6004
1 rue de la Noë , BP 92101 F-44321 Nantes Cedex 3, France

^{1}Ecole Centrale de Nantes • ^{2}Université de Nantes

(Tianyi.Yu|Konstantin.Akhmadeev|Eric.Le-Carpentier|Yannick.Aoustin)@ls2n.fr

Résumé – La décomposition d'un signal électro-myographique intra-musculaire en composantes élémentaires, les trains d'impulsion des motoneurones, permet de reconstruire le message émis par le système nerveux central pour commander la contraction musculaire. En 2014, un algorithme récursif en temps a été proposé, fondé sur un modèle de Markov dont l'état est estimé par filtrage bayésien. On propose ici une implémentation sur processeur parallèle, en vue d'une décomposition temps-réel.

Abstract – The decomposition of the intramuscular electromyographic signals in some elementary components, the spike trains of motor neurons, allows to rebuild the original order emitted by the spinal cord to activate the muscle contraction. In 2014, a time-recursive algorithm was proposed. It is based on an hidden Markov model whose state is estimated by a Bayes filter. This paper proposes an implementation on a parallel processor, in order to achieve an online decomposition.

1 Introduction

The idea to control active prosthetic devices for amputees through electromyographic (EMG) signals was proposed by N. Wiener in [1]. EMG signals represent the activity of muscle fibers, as driven by the population of spinal motor neurons (MN) innervating the muscle. Recent research works rely on the conjecture that retrieving the activity of individual motor neurons will help to improve the control of prosthetic devices [2–4]. In [5], a potentially online decomposition method of iEMG signals was proposed. The iEMG signals are modeled as a sum of independent convolved spike trains. The sparsity and the regularity of the spike trains is taken into account by a stochastic model. This leads to an Hidden Markov Model (part 2). The decomposition algorithm is a reasonable approximation of the Bayes filter (part 3). Compared with the previous decomposition methods, this one can decode iEMG signals sequentially and analyses perfectly the superimposed MUAPs. It has a parallel structure which makes it suitable to be implemented on a parallel processor such as a Graphics Processor Unit (GPU) to reach a real time implementation (part 4). The results are shown in part 5.

2 Markov model

We consider a model of EMG signal proposed in [6]:

$$Y[n] = \sum_{i=1}^{n_{\text{MN}}} (h_i * U_i)[n] + W[n] \tag{1}$$

where $Y[n]$ is the EMG signal at discrete time index n, n_{MN} is the number of active motor neurons, h_i is the response (motor unit action potential, MUAP) of the motor unit innervated by the *i-th* motor neuron, U_i is a sparse 0-1 sequence representing the spike train emitted by this motor neuron, $W[n]$ is a white gaussian noise, and $*$ denotes convolution.

We introduce the sawtooth sequence $(T_i[n])$ representing the time passed since the last spike [5]:

$$T_i[n] = \begin{cases} 0 & \text{if } U_i[n] = 1 \text{ (the MN fires)} \\ T_i[n-1] + 1 & \text{if } U_i[n] = 0 \end{cases} \tag{2}$$

In the sequel, exponent ^{n} is used for sequences until time index n (*e.g.* $T_i^n = [T_i[1], ..., T_i[n]]$). With the assumption that inter-spike intervals are independent and identically distributed, $(T_i[n])$ is a Markov chain; given T_i^n and Θ_i:

$$T_i[n+1] = \begin{cases} 0 & \text{w.p. } r_{\Theta_i}(T_i[n] + 1) \\ T_i[n] + 1 & \text{w.p. } 1 - r_{\Theta_i}(T_i[n] + 1) \end{cases} \tag{3}$$

where r_{Θ_i} is the parameterized hazard rate [7] of the inter-spike interval probability distribution. As a distribution of inter-spike intervals, we choose the discrete Weibull one [8] which fits well the real process [5] and whose hazard rate expression is simple. Discrete Weibull distribution has three parameters $\mathbf{\Theta_i} = [t_0, \beta, t_R]$: t_0 is a scale parameter, β is a shape one and t_R is a shifting one. The latter represents the minimum interval between two consecutive spikes, which is a physiological parameter with known bounds. We assume that it is bigger than the maximum length ℓ_{IR} of MUAPs, so that consecutive responses of one motor unit do not intermingle.

Furthermore, we assume that the Weibull parameters remain constant along time and that MUAPs are known. Then, vector $\mathbf{T}[n] = [T_1[n], ...T_{n_{\text{MN}}}[n]]$ and vector $\mathbf{\Theta}[n] = [\mathbf{\Theta}_1^T, ...\mathbf{\Theta}_{n_{\text{MN}}}^T]$ are the state vectors of a Markov chain. The Markov model is completed with the observation equation:

$$Y[n] = \underbrace{[\varphi(T_1[n]) \cdots \varphi(T_{n_{\text{MN}}}[n])]}_{\Phi(\mathbf{T}[n])} H + W[n] \qquad (4)$$

$\varphi(T_i[n])$ is a regression vector of ℓ_{IR} size, for all $i \in n_{\text{MN}}$, every position is 0, except the component in the position of $T_i[n] + 1$ is 1, if $0 \leq T_i[n] \leq \ell_{\text{IR}}$. Note that the state vector is composed of both discrete and continuous variables.

3 Bayes filter

The Bayes filter recursively computes the posterior probability of $\mathbf{T}^n$ and Θ given the observed sequence Y^n along time. In [5], an approximated Bayes filter is proposed, in which the expected value of the hazard rate with respect to every possible value of the Weibull parameter is replaced by the hazard rate of the estimated Weibull parameter obtained using a recursive implementation of a quasi-Newton optimization of the maximum likelihood criterion.

It is impossible to process all possible values of $\mathbf{T}^n$, since their number increases exponentially as time index n grows. The n_{path} most probable paths are kept at every time index, where n_{path} is chosen according to the available computation power.

The state vector can be estimated in the following recursion for all $n \geq 1$:

Task 1 Data transmission: Observed signal $Y[n]$

Task 2 Posterior probability of sawtooth sequences $\mathbf{T}^n$:

$$\begin{aligned} \hat{W}_{t^n}^{|n} &= Y[n] - \Phi(t[n])H \\ \text{P}^{|n}(\mathbf{T}^n = t^n) \; &\propto \; \text{P}^{|n-1}(\mathbf{T}^n = t^n) \; g(\hat{W}_{t^n}^{|n}, \hat{V}_{t^{n-1}}^{|n-1}) \end{aligned} \qquad (5)$$

where $^{|n}$ means that Y^n is known, $\text{P}^{|n-1}(\mathbf{T}^n = t^n)$ is the prior probability at time index n-1 and $g(., \hat{V}_{t^{n-1}}^{|n-1})$ is the Gauss probability density function with zero-mean and variance $\hat{V}_{t^n}^{|n-1}$. Selection of the n_{path} most probable paths.

Task 3 Estimation of the variance of the noise:

$$\hat{V}_{t^n}^{|n} = (1 - \frac{1}{n})\hat{V}_{t^{n-1}}^{|n-1} + \frac{1}{n}(\hat{W}_{t^n}^{|n})^2 \qquad (6)$$

Task 4 Update the discrete Weibull distribution parameters [9]:

$$\begin{aligned} \hat{\theta}_{t_i^n} &= \hat{\theta}_{t_i^{n-1}} - \frac{1}{n} \, G_{t_i^n}^{-1} \, Q'_{t_i^n}(\hat{\theta}_{t_i^{n-1}}) \\ G_{t_i^n} &= \frac{1}{n}[Q'_{t_i^n}(\hat{\theta}_{t_i^{n-1}})][Q'_{t_i^n}(\hat{\theta}_{t_i^{n-1}})]^T + (1 - \frac{1}{n}) \, G_{t_i^{n-1}} \end{aligned} \qquad (7)$$

where $Q'_{t_i^n}(\theta)$ is the gradient of $Q_{t_i^n}(\theta)$ with

$$Q_{t_i^n}(\theta) = \begin{cases} -\ln r(t_i[n] + 1, \theta) & \text{if } t_i[n+1] = 0 \\ -\ln (1 - r(t_i[n] + 1, \theta)) & \text{if } t_i[n+1] = t_i[n] + 1 \end{cases}$$

And the $Q'_{t_i^n}(\hat{\theta}_{t_i^{n-1}})$ is the approximate gradient vector and $G_{t_i^n}$ is the approximate Hessian matrix of the maximum likelihood criterion at the current estimate.

Task 5 Data transmission: Discrete Weibull distribution parameters $\hat{\Theta}[n]$ and the most probable sawtooth sequences $\mathbf{T}[n]$

Task 6 Bifurcation of sawtooth sequences and calculation of the priori probability:

$$\text{P}^{|n}(\mathbf{T}^{n+1} = t^{n+1}) = \text{P}^{|n}(\mathbf{T}^n = t^n) \times$$
$$\prod_{i=1}^{n_{\text{MN}}} \begin{cases} r(t_i[n] + 1, \hat{\theta}_{t_i^n}) & \text{if } t_i[n+1] = 0 \\ 1 - r(t_i[n] + 1, \hat{\theta}_{t_i^n}) & \text{if } t_i[n+1] = t_i[n] + 1 \\ 0 & \text{otherwise} \end{cases} \qquad (8)$$

4 Parallel computing model

The estimation of state sequence computed recursively in Bayes filter can be interpreted as a loop-based pattern [10]. The performance of a loop-based pattern implemented in the parallel computing structure varies in terms of the dependencies between loop iterations and the work partition between the available processors. Since the calculation of the posterior probability of sawtooth sequences $\text{P}^{|n}(\mathbf{T}^n = t^n)$ at time index n depends on the results of the prior probability $\text{P}^{|n-1}(\mathbf{T}^n = t^n)$ and the sawtooth sequences $\mathbf{T}[n-1]$ at time index n-1, it is impossible to remove the dependencies between loop iterations. Therefore one must calculate them in strictly sequential manner. In contrary, each iteration can be broken into a number of single tasks executed in parallel. In the following sections of this article we are going to analyse the structure of this algorithm to minimize communication between processors and to maximize the use of on-chip resources.

4.1 Parallelism analysis

Data parallelism is a form of parallelization based on data. It focuses on the distribution of data in the different processors that execute the same operation in parallel [10].

Paths on parallel: Before the bifurcation of sawtooth sequences $\mathbf{T}[n]$, there are n_{path} paths which are mutually independent. After the bifurcation, all new paths remain independent. So calculations in all paths could be implemented in the parallel structure with less communication between them.

Motor units on parallel: According to the hypothesis of the Markov model, there is no dependency between any two motor units. And therefore, in every path, the calculation of all motor units can be executed simultaneously.

Task parallelism is another parallelization contrasting to data parallelism [10]. Rather than simultaneously computing the same function on lots of data elements as data parallelism, task parallelism involves doing two or more completely different tasks in parallel. In the structure of decoding intramuscular EMG signal, the simultaneous execution of tasks limited by the dependences between tasks.

In every iteration, the data transmission takes place two times: data transmission of observed signal $Y[n]$ from host (CPU) to device (GPU) (task 1); data transmission of Weibull parameters $\hat{\Theta}[n]$ and sawtooth sequences $\mathbf{T}[n]$ from device to host (task 5). CUDA supports simultaneous kernel execution and the two types of memory copy. As a result, to overlap the data transmission, both of them are executed simultaneously with kernel functions.

In addition, the Nvidia's GPUs whose micro-architecture is issued after "Fermi" supports concurrent kernel execution [11, 12], where different small kernels of the same application context can be executed at the same time to utilize the whole GPU. Therefore, in every loop, the estimation of the variance of noise (task 3) and the calculation of the prior probability (task 6) are executed at the same time. Scheme illustrating these two applications of task parallelism is figure 1.

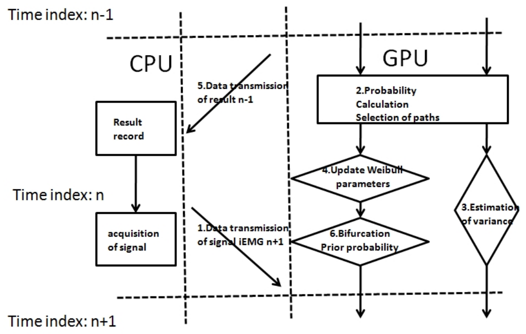
Time index: n-1
CPU
GPU
5.Data transmission of result n-1
2.Probability Calculation Selection of paths
Result record
4.Update Weibull parameters
Time index: n
3.Estimation of variance
acquisition of signal
1.Data transmission of signal iEMG n+1
6.Bifurcation Prior probability
Time index: n+1

Figure 1: Structure of task parallelism

4.2 Task analysis

In every iteration, tasks 1 and 5 are data transmissions. Task 3 (estimation of the variance of the noise) and task 4 (update the discrete Weibull distribution parameters) are ordinary small-size parallel computing problems, whereas task 2 consists of posterior probability calculation of sawtooth sequences $\mathbf{T}[n]$ and selection of the n_{path} most probable paths which is related to a classic parallel sorting problem. Task 6 (bifurcation of sawtooth sequences) which change the size of parallel structure also deserves consideration.

Task 2: selection of the n_{path} most probable paths After the bifurcation of sawtooth sequences, with respect to the transition distribution presented in (2), there are usually $n_{\text{path}} \times 2^{n_{\text{MN}}}$ paths in maximum where n_{MN} is the number of motor units. The size of sorting problem in parallel varies from n_{path} to $n_{\text{path}} \times 2^{n_{\text{MN}}}$. In our parallel model, it assumed that there are at most 4 motor units in the iEMG signal. Therefore, it's a small size sorting problem. For small sequences, bitonic sort is usually considered as the fastest traditional sort [13]. The work complexity of bitonic sorting is $O(n \ \log_2^2 n)$. In the parallel environment, the time complexity of bitonic sorting network is $O(\log_2^2 n)$. The most important operation of the bitonic sorting is the arrangement of a bitonic sequence into a sorted sequence which implements perfectly in the parallel computing.

Task 6: bifurcation of sawtooth sequences Path $\mathbf{T}[n]$ bifurcates in at most $2^{n_{\text{MN}}}$ different ways giving an overall number of $n_{\text{path}} \times 2^{n_{\text{MN}}}$ of new paths. To avoid the memory initialisation and allocation of each bifurcation originated from one path, indexing is used. We note that each old path occupies a separate memory block N_b and all its bifurcations receive a separate thread. Here is an example for two active motor neurons, which gives a two-dimensional vector $\mathbf{T}[n]$ and four possible bifurcations (used values are arbitrary):

$$\text{if } \mathbf{T}[n] = \begin{bmatrix} 450 \\ 635 \end{bmatrix}, \quad \mathbf{T}[n{+}1] \in \left\{ \begin{bmatrix} 451 \\ 636 \end{bmatrix}, \begin{bmatrix} 0 \\ 636 \end{bmatrix}, \begin{bmatrix} 451 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} \tag{9}$$

Each i-th motor unit can either not fire at time $n+1$ ($T_i[n+1] = T_i[n]+1$) or fire if ready ($T_i[n+1] = 0$). Therefore, to each configuration in $\boldsymbol{T}$, a binary code can be associated.

$$\mathbf{T}[n+1] \mapsto \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}; \tag{10}$$

This code is unique for each bifurcation and it covers all serial numbers of thread N_t from 0 to $2^{n_{\text{MN}}} - 1$. These numbers and the serial numbers of block N_b are then used as indexes for threads dedicated to each bifurcation. Its state $T[n+1]$ can be decoded using $\mathbf{T}[n]$ in N_bth path, for each motor neuron i:

$$T_i[n+1] = (T_i[n]+1) \times \begin{cases} 1 & \text{if } \frac{N_t}{2^i} \bmod 2 = 0 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Single-precision floating-point operations were used on GPU. Because the difference of resultats calculated by the type of float and double is not distinguishable by eyes. And single-precision floating-point saves space of shared memory and register to accelrate the programme. To avoid the round-off error accumulating in sums of many elements, Kahans Summation formula [14] was used.

5 Results

The algorithm was evaluated on simulated iEMG signal, generated using the Markov model described earlier, with sampling frequency of 10 kHz and duration 3 s. MUAP shapes were taken from a real iEMG signal decomposed by hand. There were four active motor units. The value of the shifting parameter of Weibull distribution was chosen to be 30 ms; the location parameter t_0 of inter-spike interval ranged from 60 ms to 90 ms; and the concentration parameter β ranged from 1 to 10. The SNR (Signal to Noise Ratio) was set to 20 dB.

Simulated iEMG signal was decomposed using parallel implementation of the algorithm with 64 paths. As shown in figure 2, the reconstructed signal without noise obtained by Markov model coincides well with the simulated signal. The performance of estimation of four spike trains is 100% correct. And in the figure 3, it shows the discharge rate and the mean value of four firing sources over 3 seconds. The discharge rate depends on the Weibull parameters t_0 and β that is sequentially estimated in the decoding model. This value gradually stabilises within one second after beginning of the decomposition for all active motor neurons.

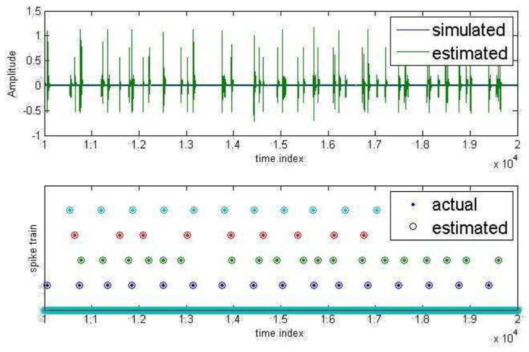

Figure 2: Reconstruction of the simulated signal of the 2nd second and Estimation of spike trains of four sources

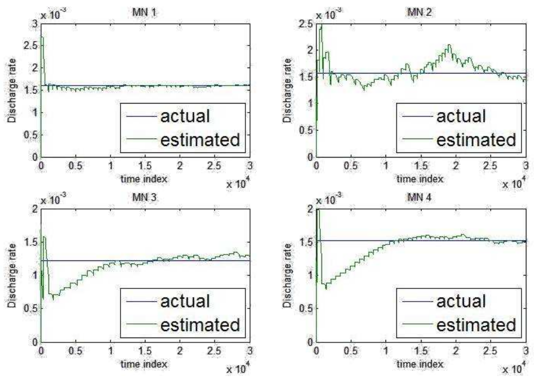

Figure 3: Discharge rate and the mean value of four firing sources over 3 seconds

The execution time of the algorithm in serial structure is tested in the software MATLAB 2014b installed on a PC system with Central Processing Unit: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and RAM 8 Go. The parallel computing is implemented in the software VISUAL STUDIO 2013 and CUDA 7.5 installed on a laptop with Central Processing Unit: Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz and Memory Installed RAM 16.00 Go; Graphics Processing Unit: Intel(R) HD Graphics 4600 and NVIDIA GeForce GTX 780M.

The execution times of parallel and serial implementations were respectively 57 ± 3 s and 5214 ± 133 s. The parallel signal decomposition is 91.5 times faster than the serial computing of decoding signal.

6 Conclusion

The implementation of sequential decomposition of intramuscular EMG signals via estimation of a Markov model in the parallel environment has shown a considerable acceleration of the calculations. The high velocity derives from the data parallelism and task parallelism in every recurrence of state vector estimation and the work partition between the available processors. Though the proposed implementation is still not a real-time one, the computation time is significantly reduced, making the online implementation likely.

References

[1] N. Wiener. *Cybernetics or control and communication in the animal and the machine*. MIT Press, 1948.

[2] Dario Farina et al. The extraction of neural information from the surface emg for the control of upper-limb prostheses: Emerging avenues and challenges. *IEEE Trans. on neural systems and rehabilitation engineering*, 22(4):797–809, 2014.

[3] Francesco Negro, Silvia Muceli, Margherita Castronovo, and Dario Farina. Multi-channel intramuscular and surface emg decomposition by convolutive blind source separation. *Journal of Neural Engineering*, 13(2), 2016.

[4] Dario Farina et al. Man/machine interface based on the discharge timings of spinal motor neurons after targeted muscle reinnervation. *Nature biomedical engineering*, 1, 2017.

[5] J. Monsifrot, E. Le Carpentier, Y. Aoustin, and D. Farina. Sequential decoding of intramuscular emg signals via estimation of a markov model. *IEEE Trans. on neural systems and rehabilitation engineering*, 22(5):1030–1040, 2014.

[6] Dario Farina, A. Crosetti, and R. Merletti. A model for the generation of synthetic intramuscular EMG signals to test decomposition algorithms. *IEEE Trans. on Biomedical Engineering*, 48(1):66–77, 2001.

[7] V. Barbu and N. Limnios. Reliability theory for discrete-time semi-markov systems. In *In semi-Markov Chains and Hidden Semi-Markov Mdels toward Applications*, volume 191, pages 1–30, 2008.

[8] T. Nakagawa and S. Osaki. The discrete weibull distribution. *IEEE Trans. on Reliability*, R-24(5):300–301, 1975.

[9] J. Monsifrot, É. Le Carpentier, and Y. Aoustin. Modélisation de trains d'impulsions à l'aide d'une loi de Weibull discrète. Estimation hors-ligne et séquentielle des paramètres. In *24e colloque GRETSI sur le traitement du signal et des images*, Brest, France, September 2013.

[10] Shane Cook. *CUDA Programming A Developer's Guide to Parallel Computing with GPUs*. Morgan Kaufmann, 2013.

[11] NVIDIA Corporation. *Whitepaper NVIDIA's Next Generation CUDA Compute Architecture: Fermi*. NVIDIA Corporation, 2009.

[12] Jason Sanders and Kandrot. *CUDA by example: an introduction to General-purpose GPU Programming*. Addison-Wesley Professional, 2010.

[13] A. C. Dusseau et al. Fast parallel sorting under logp: Experience with the cm-5. *IEEE Trans. Parallel Distrib. Syst.*, 7(8):791–805, 1996.

[14] Kahan William. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.