Towards Ubiquitous Mobile-Computing-Based Artificial Leg Control

Robert Hernandez, *Senior member, IEEE*, Jason Kane, *Student member, IEEE*, Fan Zhang, *Student member, IEEE*, Xiaorong Zhang, *Student member, IEEE*, He Huang, *Senior member, IEEE*

Abstract—**This paper presents a rapid prototype approach for the development of a real-time capable neural-machine-interface (NMI) for control of artificial legs based on mobile processor technology (Intel Atom^{TM} Z530 Processor.) By effectively exploiting the architectural features of a mobile embedded CPU, we implemented a decision-making algorithm, based on neuromuscular-mechanical fusion and gait phase-dependent support vector machines (SVM) classification to meet the demanding performance constraints. To demonstrate the feasibility of a real-time mobile computing based NMI, real-time experiments were performed on an able bodied subject with window increments of 50ms. The experiments showed that the mobile computing based NMI provided fast and accurate classifications of four major human locomotion tasks (level-ground walking, stair ascent, stair descent, and standing) and a 46X speedup over an equivalent MATLAB implementation. The testing yielded accuracies of 96.31% with low power consumption. An offline analysis showed the accuracy could be increased to 98.87% with minor modifications to the application.**

Keywords—neural machine interface; leg control; support vector machines; mobile CPU

I. Introduction

In 2005, there were approximately 1.6 million people in the United States with some kind of limb loss [1]. By the year 2050, the number is expected to increase to 3.6 million people [1]. Furthermore, in 2005, lower limb loss accounted for almost two-thirds (65.5%) of the 1.6 million [1]. People with lower-limb amputations typically favor their intact limb and therefore provide additional stress upon their intact limb during everyday activities [2]. It has been speculated the additional placed stress upon their intact limb will lead to degenerative diseases [2]. These statistics clearly present the increasing need for technology that restores as much functionality to the large and increasing population of lower limb amputees.

The recent development of powered artificial legs, such as the Power Knee [3] and the Vanderbilt University design [4], provide positive mechanical energy that helps restore the user's locomotion modes [5]. These devices detect the user's intended locomotion mode though the use of echo control or solely though intrinsic mechanical feedback. In particular, the Power Knee [3] utilizes echo control [4] and requires instrumentation of the sound leg in order to detect what locomotion mode the user is currently performing. The system described in [4] utilizes, solely, intrinsic mechanical feedback [6]. In contrast, we have developed a Neural-Machine Interface (NMI) based on neuromuscular-mechanical fusion [7] and phase-dependent pattern recognition (PR) strategy [8]. Our strategy does not require instrumentation of the sound leg and has been shown to provide higher accuracy than the classifiers utilizing only electromyographic (EMG) data or only mechanical data [9]. Our PR strategy can be implemented utilizing either Support Vector Machines (SVM) or Linear Discriminant Analysis (LDA) Classifiers and once properly trained can provide autonomous classifications of the user's intended locomotion at 50ms intervals, thereby providing volitional control of a powered artificial leg. The selection of a Support Vector Machines (SVM) classifier provided improved prediction accuracy performance of our PR strategy when compared to a Linear Discriminant Analysis (LDA) classifier [7]; therefore for this study we will utilize an SVM-based classifier.

In order to make our PR strategy a feasible reality we developed a Cyber Physical System (CPS), designed to test our NMI. This CPS is a unique and complex system consisting of sensors (EMGs and a six degrees of freedom (DOF) Loadcell), which need to integrated with a data acquisition system, a biomedical engineering algorithm, a mechanical prosthesis, and a computational engine. Our objective here was to integrate the various components in our complex system in an optimal fashion using a System-of-Systems approach. During each stage of integration we performed system verification and validation to ensure that the components were behaving as expected. The important performance criteria that we aim to optimize include mainly 1) real-time performance to provide fast control of prosthesis; 2) high accuracy of locomotion prediction; 3) low power consumption; and 4) small size wearable by leg amputees.

With these objectives in mind we investigated commercial off-the-shelf (COTS) computing devices and chose one ubiquitous mobile computing system, the Intel Atom^{TM} Z530 as our computational engine for our first real-time capable design iteration. It is low power (2.2 watts [10]), low cost, and a portable mobile computer that meets our NMI performance requirements. Our preliminary study [11] showed that a mobile processor based NMI had great promise in control of artificial legs [11]. The primary objective of this paper is to determine

This research was supported in part by the Department of the Navy (Naval Undersea Warfare Center, Newport, Rhode Island), the Office of Naval Research (ONR) Independent Applied Research (IAR) initiative, NSF/CPS #0931820, NSF#1149385, NIH #RHD064968A, NSF/CCF #0811333 and NSF/CCF #1017177.

R. Hernandez is with the Naval Undersea Warfare Center, Newport, RI, 02842, USA (e-mail: robert.hernandez2@navy.mil).

J. Kane, F. Zhang, X. Zhang and H. Huang are with the Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA (e-mail: jkane@ele.uri.edu; fzhang@ele.uri.edu; zxiaorong@ele.uri.edu; huang@ele.uri.edu).

the viability of mobile technology as a possible architectural solution for use in our 50ms window increment NMI. We chose to utilize 50ms window increments in this study to provide a comparison with our existing MATLAB implementations. Additionally we wish to determine if the Intel Atom based design will allow for further expansion of our NMI algorithm to perform electromyographic (EMG) anomaly detection and perform the prosthesis leg control by sending control signals based on our PR strategy at 10ms intervals, it is desirable to quantitatively evaluate the mobile technology's reserve capability while executing our 50ms window increment NMI.

Existing solutions for prosthesis control have been implemented on MATLAB that cannot satisfy real-time requirements running on the mobile computing device. We have developed an entire software application suite to implement our SVM-based PR strategy in C, targeting a mobile processor. It turns out that porting the software to the mobile computer present several challenges to meet our goals. The first challenge is the time constraint of the NMI to deliver correct control decision in real time. Straight forward implementation is far from satisfactory. We therefore proposed several innovative techniques to exploit the inherent architectural features, which are described in [11].

To meet our research objectives, we designed and developed a real time software interface to a data acquisition system (DAQ) providing the capability to acquire real-time EMG, mechanical force and moment data from human subjects, with no data loss or lag. This newly developed NMI was combined with a Measurement Computing USB-1616HS-BNC DAQ [12] to facilitate the collection of the real-time EMG and 6 degrees-of-freedom (DOF) mechanical data. This final NMI design was utilized to execute and test the real-time performance of our phase dependent SVM based PR algorithm utilizing 50ms window increments on an able bodied human subject.

This paper makes the following contributions:

- Design and implementation of a real-time capable NMI utilizing 50ms window increments for artificial leg control based on a mobile processor;
- Design and implementation of an application suite for a phase-dependent NMI with SVM classifiers tailored specifically to the mobile processor utilizing 50ms window increments;
- A comparison between our new C based NMI embedded application and our equivalent MATLAB based NMI that shows the embedded C application provides a 46X speedup;
- A real time experiment that evaluates the potential use of mobile processors for a 50ms window increment embedded implementation for neural control of powered lower limb prosthesis;
- An analysis that shows the future algorithm expansion capability of this mobile based NMI implementation.

This paper builds upon a prior offline study [11] and an Analysis of Alternatives (AoA) [13]. The rest of this paper is organized as follows. Sections 2, 3, 4, and 5 present our newly designed and developed 50ms window increment real-time system design, software implementation, experimental protocol and performance evaluation. Section 6 presents recommended updates to our new real-time algorithm and updated performance expectation. We conclude our paper in Section 7.

II. Real-Time Capable System Architecture

Based on the offline performance and the results of our AoA [13], it was decided to continue using the AxiomTek eBOX530-820-FL fanless embedded hardware with the Intel Atom^{TM} Processor Z530 [14] (512K cache, 1.6 GHz) as our COTS mobile computing system. During the source selection of a DAQ to combine with AxiomTek embedded hardware, it was clear that the vast majority of COTS DAQ devices with the capability to meet our design requirements (16 analog input channels and simultaneous sampling or a similar capability) only provided drivers for the Windows and Linux operating systems. The NMI design needs to meet real-time constraints and therefore the use of a Real-Time Operating System (RTOS) is preferable. An RTOS performs its functions, including external events in a specified amount of time [15]. Windows and Linux are general purpose operating systems (OSs) and do not meet the criteria of an RTOS. Therefore, as a compromise, it was decided to utilize a general purpose operating system with the understanding that RTOS options were available for both Windows and Linux implementations, such as Windows Compact Embedded (WinCE) [16] and Real-Time Linux (RT Linux) [17]. Furthermore, it would be expected that if real-time constraints can be met with a general purpose OS, then porting the design to an RTOS would provide better system response and make the design more deterministic. The decision to go with the Windows OS vs. Linux was based on the experience and familiarity of the research team with the Microsoft Visual Studio product. This familiarity would facilitate the rapid design, implementation and debugging of the prototype COTS solution.

For our COTS prototype, Measurement Computing's USB-1616HS-BNC DAQ was chosen and integrated with the AxiomTek eBOX530-820-FL fanless embedded hardware to facilitate the integration of the EMG and loadcell sensors to provide the necessary real-time data to make our neuromuscular-mechanical fusion SVM NMI a feasible reality. The Measurement Computing device met all our performance requirement, provided a C-library interface that was capable of interfacing with our prior embedded software design, and was easily interfaced to the AxiomTek embedded hardware via a universal serial bus (USB) port. A block diagram of our system hardware architecture is shown in Fig. 1.

III. Real-Time Capable application suite

All of the initial software architectural and implementation decisions made in our design, such as the use of the C programming language, loop unrolling and inline function expansion were utilized within the real-time implementation. In addition a few other techniques were incorporated to augment and provide further performance enhancement.

A. *Software Architecture*

The use of a general purpose OS in this initial prototype design iteration raised concerns with the embedded software's

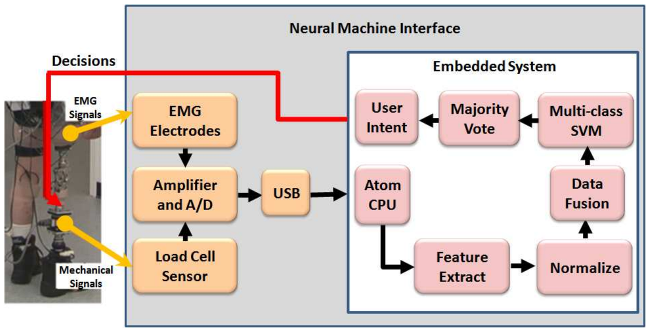

Fig. 1. System hardware architecture block diagram

capability to meet real time constraints. Therefore, to further reduce the impact of the OS on the embedded application, the priorities of the application and thread were increased to a real time critical status. In a Microsoft Windows OS, this is accomplished by setting the priority class to REALTIME_PRIORITY_CLASS and the thread priority to THREAD_PRIORITY_TIME_CRITICAL [18].

The real-time software implementation required that all raw data, phase data, and classification data be logged to allow for performance evaluations. To minimize the impacts of the real-time data logging on the application, a statically allocated and statically defined Random Access Memory (RAM) buffer was implemented that stored all the raw EMG, mechanical, classification and application performance data. The RAM buffer eliminated the need to write to the hard drive during time critical operations. Furthermore, it took advantage of the RAM's superior speed for storage. The real-time data logging for each classification was performed after all time-critical functions were completed (i.e., at the end of each classification). Lastly, The RAM buffer's contents were written to the hard drive for post analysis after the experiment was completed, by which point no further time critical functions were being executed.

Our prior offline analysis utilized LIBSVM [19] for the SVM training and classification. Analysis of the LIBSVM source showed that it could be possible to modify the libraries for real-time use. Therefore, the open source library LIBSVM was used and specifically tailored to our embedded NMI application for real-time SVM classification. Our real-time data acquisition, feature extraction, normalization, data fusion, 5-point majority vote and low pass filter routines were all developed as part of our research and optimized for execution on the mobile CPU.

Lastly, the re-implementation of our prior software optimizations [11] and the newly incorporated additional enhancements resulted in an embedded application specifically designed to minimize pipeline stalls, minimize OS impacts, minimize cost of memory allocation, minimize the impacts of real-time data logging and take advantage of the Intel Atom^{TM} Z530 Processor hardware architecture. These enhancements provided the basis for the performance introduced by this embedded application suite.

B. Real-Time Software Implementation

To implement the real-time Phase-Dependent PR algorithm, four applications were required. Where previously the offline study's data was read in via a file, the real-time study requires a new application to be developed to interface with the DAQ and capture real-time training data. The feature extraction & normalization application, as well as the SVM training application remained unchanged. Finally, the Neuromuscular-Mechanical Fusion PR application had to be modified to acquire real time data testing from the DAQ. The training data capture application acquires data for all of the various human locomotion tasks, segregates the data into each locomotion class, and allows for multiple trials of each locomotion task. The real-time PR application is used during the real-time testing phase. The real-time PR application extracts EMG and mechanical features from the raw testing data acquired in real-time from the DAQ. Similarly to the offline method, the features are then fused and normalized with the provided normalization parameters and formed into a vector. Finally, the application determines the current locomotion gait phase, and forwards the test vector to the respective phase based classifier for determination of user intent. The software implementation data flow is shown in Fig. 2.

IV. Real-Time Experimental Protocol

A real-time performance evaluation utilizing a 50ms window increment and an offline performance evaluation utilizing a 50ms window increment were performed as part of the real-time study. The evaluations were performed on the data collected from a male able bodied subject. The collected data included the EMG signals from the subject's thigh muscles and mechanical forces/moments measured by a 6 degree-of-freedom load cell mounted on the prosthetic pylon. The monitored muscles included the sartorius (SAR), rectus femoris (RF), vastus medialis (VM), adductor magnus (ADM), biceps femoris short head (BFS), biceps femoris long head (BFL), and semitendinosus (SEM).

The EMG and mechanical forces/moments were sampled at 1 kHz by the Measurement Computing USB-1616HS-BNC DAQ device. The user intent decisions provided by the embedded hardware were routed via an analog output interface on the DAQ device. The real-time experiments provided real-time gait-phase and user intent decisions to the console screen as a visual cue during the training and testing processes. The 50ms window increment experiments utilized a window increment of 50ms and a window length of 150ms.

For all experiments performed in this study, the prediction time will be defined as the total time to execute feature extraction, normalization, gait phase detection, majority vote (if performed) and classification for a single analysis window.

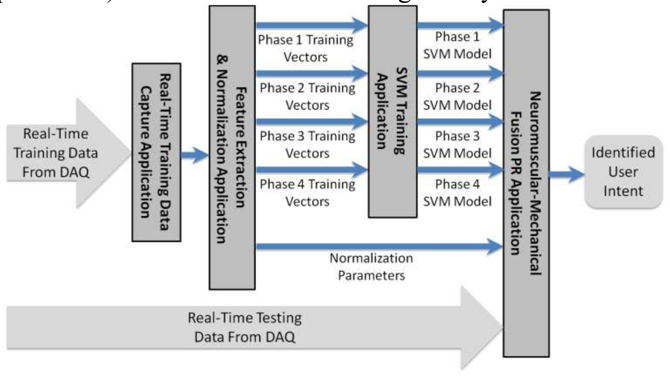

Fig. 2. Real-time software implementation data flow

V. REAL-TIME PERFORMANCE VERIFICATION

For this experiment, four tasks (level-ground walking (W), stair ascent (SA), stair descent (SD), and standing (ST)) were studied and captured for offline analysis. To ensure the subject's safety, the subject was allowed to use hand rails when necessary. To train the gait-phase classifier, the subject was instructed to perform each task for approximately 10 seconds. Two trials of standing data, three trials of walking data, three trials of stair descent and three trials of stair ascent data were accumulated to train the classifier. For the real-time performance evaluation, the subject was instructed to stand and then transition to one of the other tasks ($ST \rightarrow W$, $ST \rightarrow SD$ and $ST \rightarrow SA$). Seven trials of each mode transition were conducted, for a total of 21 trials. To assess the real-time performance of the NMI, the timing and processor loading of the application's execution on the embedded hardware are provided and the raw recognition accuracy of the NMI will be evaluated via the following criteria:

Classification Accuracy in the Static States: For all experiments in this paper, the static state is defined as the state where the subject has completed a transition and is continuously performing the same task (W, SA, SD or ST). The classification accuracy in the static state is the total number of correct classifications observed over the total number of classifications during the static state.

The overall raw classification accuracy of the NMI in the static states for all 21 trials and all tasks (W, SA, SD and ST), when executed on the Intel Atom^{TM} based embedded hardware was 96.31%. A total of 5937 static state predictions were produced by the Intel Atom^{TM} based embedded hardware during the 21 trials. The mean prediction time for all of the predictions performed during the 21 trials was 0.7683ms with a standard deviation of 0.0971ms. The worst case prediction executed in 2.0192ms.

Due to the fact that there is additional loading on the CPU to execute the data logging for post analysis, the CPU loading provided by the operating system may be inaccurate; therefore the mean and maximum values of CPU loading were calculated using Equation (1), which were 1.54% and 4.04% respectively. These results show that the majority of the time, the embedded software design was awaiting new EMG and Loadcell data from the DAQ, as shown in Fig. 3. During this time the processor is idle and can be utilized to execute other additional algorithms to augment our NMI's capability.

Although 96.31% accuracy is very good, it fell short of the average 97% accuracy that was achieved by the MATLAB model in the offline analysis. Furthermore, based on the offline analysis, this implementation was expected to perform approximately 1% higher than the MATLAB model due to the use of a different SVM gamma value. Upon further review of [20], it became obvious that this 50ms window increment embedded software design did not incorporate a real-time majority vote method. Upon examination of the raw data, it was apparent that a 5-point majority vote method could have a substantial effect on the overall system accuracy. For example, in Fig. 4 we see a real-time stair ascent trial with 6 misclassifications. We manually post processed the stair ascent data, implementing the 5-point majority vote, which led to the removal of all misclassifications as shown in Fig. 5. The implementation of a majority vote increased the accuracy from

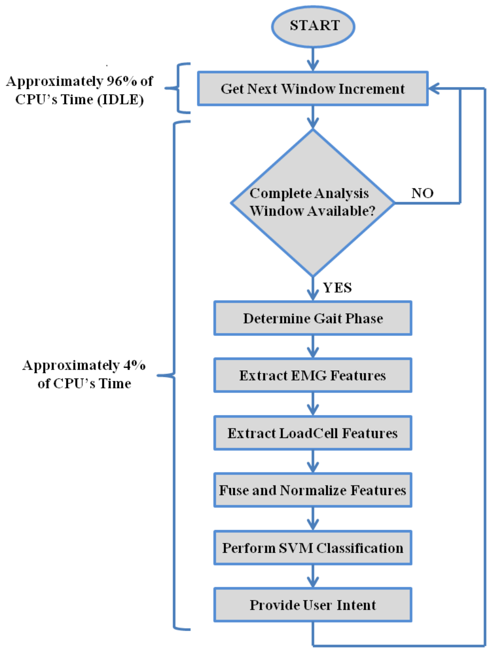
START
Approximately 96% of CPU's Time (IDLE)
Get Next Window Increment
Complete Analysis Window Available?
NO
YES
Approximately 4% of CPU's Time
Determine Gait Phase
Extract EMG Features
Extract LoadCell Features
Fuse and Normalize Features
Perform SVM Classification
Provide User Intent

Fig. 3. Simplified real-time software flowchart and CPU utilization

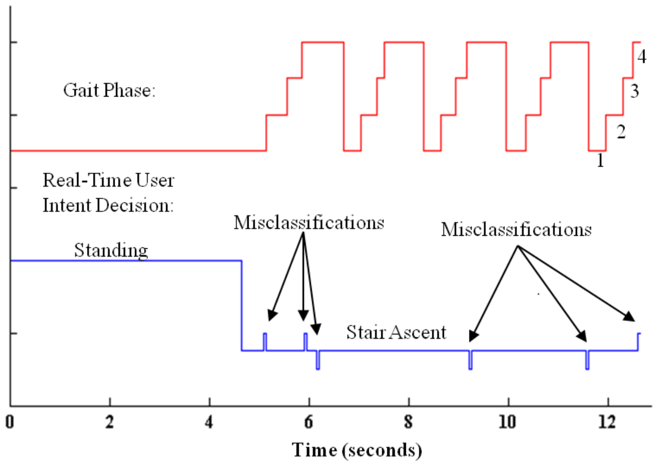

Fig. 4. Real-time stair ascent trial showing misclassification prior to majority vote implementation

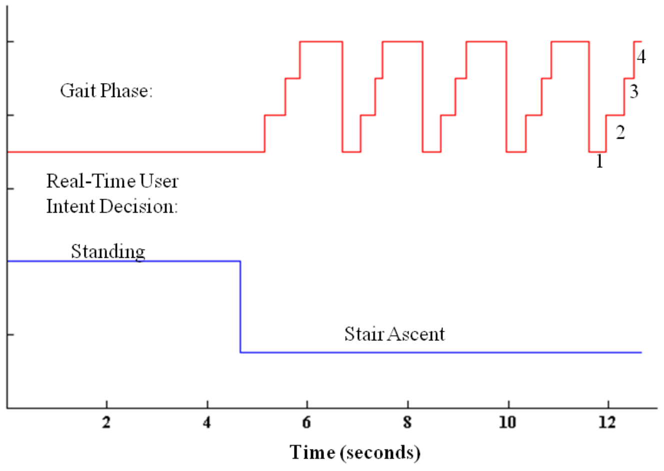

Fig. 5. Stair ascent trial manually post processed with a 5-point majority vote showing no misclassifications

97.9% to 100% for this trial. In order to determine the effect of the 5-point majority vote algorithm to the overall accuracy of the system, it was decided to perform an offline analysis of the data acquired from the real time experiment with the addition of a majority vote implementation to the algorithm.

VI. Modified Algorithm Performance Verification

The 50ms window increment offline evaluation utilized the exact data acquired during the real-time experiment. This allows for an accurate comparison between the original software design and this proposed design.

To perform this evaluation, the initial software was modified to utilize the raw DAQ data logged during the real-time testing. The algorithm was further modified to provide a five-point majority vote algorithm as in [20]. For this experiment, the same four tasks (W, SA, SD, and ST) were examined. Since the intent of this study is to determine the mobile CPU's capability to execute our PR algorithm, initially it was determined that examining the *Classification Accuracy in the Static States* should suffice. However, since slight modification to the software would enable mode transition performance evaluations that initiate from a standing position and all of the raw data was recorded during the real-time trials, we were also able to examine the performance during the three mode transitions (ST→W, ST→SD and ST→SA), therefore the analysis was performed and the results have been included within this paper. Additionally, included in the offline evaluation is a speedup assessment of the C based embedded application to the MATLAB based application. The performance of the NMI will be evaluated using the following criteria:

Classification Accuracy in the Static States: As previously defined in the real-time 50ms experiment.

The Number of Missed Mode Transitions: For this experiment, the mode transition period starts from the beginning of gait phase 2 (single limb stance) and terminates at the beginning of gait phase 4 (swing). A mode transition is declared to have been missed, if no correct transition decision is made during this defined period.

Mode Transition Prediction Time: The mode transition prediction time in this experiment is defined as the amount of time prior to the critical timing, during which the classifier user intent decision has stabilized and is no longer changing, such that safe switching of the prosthesis device is made possible. For this experiment, the critical timing is defined as the termination of the mode transition (i.e. - just prior to the start of the swing gait phase).

A. *Speed Up Provided by the C Embedded Application*

A self-contained version of the PR embedded application was built with raw test data resident within the application itself. Timing analysis software was added to verify the performance of the embedded software design and implementation. To provide an accurate comparison between the MATLAB based NMI and our C based embedded application, our application was executed on the MATLAB system for a determination of average prediction time. The MATLAB system is composed of Core 2 Duo E7500 CPU clocked at 2.93 GHz with 3GB of RAM and executes the Windows XP operating system. A total of 1002 classifications were performed by the PR embedded application on the MATLAB system and completed in 472.53ms. This results in an average of 472 microseconds per classification. The average classification time of the MATLAB model executed on the same system was 21.9ms. Based on this experiment, the C based embedded application provides a 46X speedup over the MATLAB model.

Although this is obviously not an entirely fair comparison (i.e. - MATLAB vs. C), this does provide critical information that is useful to systems engineers; it allows them to understand what speedup can be achieved and/or expected by simply porting a MATLAB algorithm to an optimized C-based application. We would like to have provided a comparison of our PR algorithm to other embedded architectures, but this is our first SVM-based embedded implementation, therefore no other comparison is available.

B. *Recognition Accuracy of NMI*

The overall classification accuracy of the NMI in the static states for all 21 trials and all tasks (W, SA, SD and ST) was 98.87%. No missed mode transitions were observed during the defined mode transition period. The mean mode transition prediction time for ST→SA was 871.4ms with a standard deviation of 197.6ms. The mean mode transition prediction time for ST→W was 528.6ms with a standard deviation of 107.5ms. The mean mode transition prediction time for ST→SD was 314.3ms with a standard deviation of 94.5ms. The mode transition performance implies that user intent classification during transitions can be accurately determined, on the average, 314.3ms prior to the critical timing and be used for safe switching and control of the prosthesis. Representative trials, depicting the user intent classifications prior and during the ST→SA, ST→W and ST→SD mode transitions are provided in Figures 6, 7, and 8, respectively. As can be seen in Figs. 6 thru 8, there were a few misclassifications during the ST→W and ST→SD transitions, but it can be seen that the transitions were correctly predicted prior to the critical timing and the static state accuracy was 100% during these three trials.

This revision to the algorithm provided an additional 2.5% accuracy in static states, while still meeting all of its other performance requirements. This clearly showed that the majority vote method is a critical component of the algorithm and must be included in future implementations and/or expansion of the algorithm.

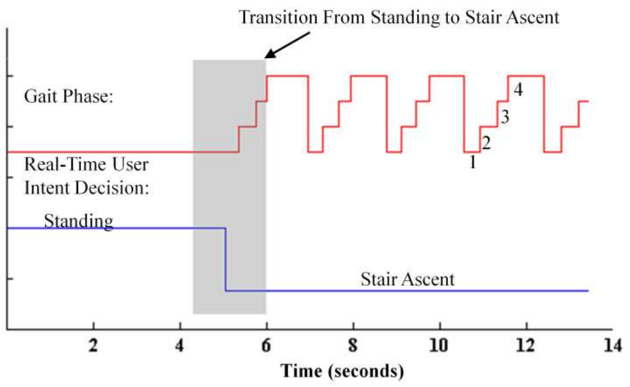

Fig. 6. Offline performance of a standing to stair ascent trial

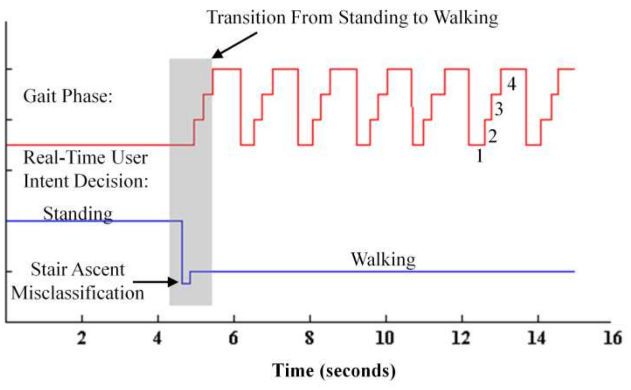

Fig. 7. Offline performance of a standing to walking trial

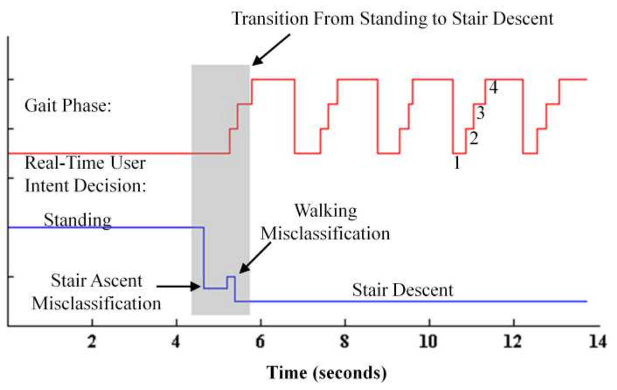

Fig. 8. Offline performance of a standing to stair descent trial

VII. CONCLUSIONS

This paper presented the design and implementation of a mobile CPU based neural machine interface for artificial legs. We utilized a rapid prototype approach, making use of COTS sensors and embedded hardware to build a cyber physical system to host our NMI algorithm. The designed NMI prototype was tested on an able-bodied subject for classification of various locomotion modes (level-ground walking, stair ascent, stair descent and standing) in real-time. The 50ms window increment real-time testing yielded 96.31% classification accuracy in static states. An additional 50ms window increment offline analysis was performed with the inclusion of a 5-point majority vote algorithm, which yielded 98.87% classification accuracy in static states, while utilizing less than 4.04% of the Intel Atom^{TM} CPU. Lastly, we provided an analysis that showed the 50ms embedded application provided a 46X speedup over an equivalent MATLAB implementation.

The experiments showed fast response time for predicting the mode transitions. Lastly, this mobile CPU based design utilizes less power than other systems designed for similar applications [11], while still providing nearly 96% reserve to provide additional expansion capability of our NMI. The results demonstrated the feasibility of a mobile CPU based real-time NMI for control of artificial legs.

Our future work includes utilizing the reserve capacity provided by this efficient implementation to provide real-time impedance based leg control [21, 22], real-time EMG motion artifact detection [23, 24], real-time EMG signal trust assessments [23, 24] and the development of a 20ms window increment NMI.

REFERENCES

[1] Ziegler-Graham K, MacKenzie EJ, Ephraim PL, Travison TG, Brookmeyer R. Estimating the prevalence of limb loss in the United States: 2005 to 2050. Arch Phys Med Rehabil. 2008;10:422–429.

[2] Gailey, Robert, Kerry Allen, Julie Castles, Jennifer Kucharik, and Mariah Roeder, Review of secondary physical conditions associated with lower-limb amputation and long-term prosthesis use. Journal of Rehabilitation Research & Development. 2008.

[3] OSSUR, The POWER KNEE. [Online]. Available: http://www.ossur.com/prosthetic-solutions/bionic-technology/power-knee

[4] F. Sup, A. Bohara, and M. Goldfarb, "Design and control of a powered transfemoral prosthesis," *In. J. Rob. Res.*, vol. 27, no. 2, pp. 263-273, Feb. 1, 2008.

[5] Hargrove, Levi J., Ann M. Simon, Robert Lipschutz, Suzanne B. Finucane, and Todd A. Kuiken. Non-weight-bearing neural control of a powered transfemoral prosthesis. Journal of Neuroengineering and Rehabilitation. 2013.

[6] H.A. Varol, F. Sup, and M. Goldfarb, "Multiclass real-time intent recognition of a powered lower limb prosthesis," *IEEE Trans. Biomed.* Eng., vol. 57, no. 3, pp. 542-51, Mar. 2010.

[7] H. Huang, F. Zhang, L. J. Hargrove, Z. Dou, D. R. Rogers, and K. B. Englehart, "Continuous locomotion-mode identification for prosthetic legs based on neuromuscular-mechanical fusion," IEEE Trans Biomed Eng, vol 58, pp. 2867-75, 2011.

[8] H. Huang, T. A. Kuiken, and R. D. Lipshutz, "A strategy for identifying locomotion mode using surface electromyography," IEEE Trans Biomed Eng, vol 56, pp. 67-73, 2009.

[9] F. Zhang, H. Huang, "Source Selection for Real-Time User Intent Recognition Toward Volitional Control of Artificial Legs," IEEE Journal of Biomedical and Health Informatics, vol. 17, 2013

[10] Intel Corporation. (2010, June). "*Intel® Atom™ Processor Z5xx Series Datasheet*" [online]. Available: http://www.intel.com/content/www/us/en/processors/atom/atom-z540-z530-z520-z510-z500-45-nm-technology-datasheet.html [March 19, 2012].

[11] R. Hernandez, F. Zhang, X. Zhang, H. Huang and Q. Yang, "Promise of a Low Power Mobile CPU based Embedded System in Artificial Leg Control," *Conf Proc IEEE Engineering in Medicine and Biology (EMBC) 2012*.

[12] Measurement Computing Corporation. (2008). "*USB-1616HS-BNC User's Guide*" [online]. Available: http://www.microdaq.com/measurement_computing/documents/usb-1616hs-bnc-user-manual.pdf [May 21, 2012].

[13] R. Hernandez, and J. Faella, "Towards Policy and Guildelines for the Selection of Computational Engines," *Conf Proc IEEE Systems Conference 2013,* Orlando, FL, April 2013.

[14] AxiomTek Corporation. (2012). "*Fanless Embedded System with Intel® Atom™ Processor*" [online]. Available: http://axiomtek.com/Download/Spec/ebox530-820-fl.pdf [March 19, 2012].

[15] B. Furht, D. Grostick, et. al., "Real-time UNIX systems design and application guide", Kluwer Academic Publishers Group, Norwell, MA, USA

[16] Microsoft Corporation. Deveolpment Tools | Windows Embedded CE Tools for Developers. [online] Available: http://www.microsoft.com/windowsembedded/en-us/develop/windows-embedded-ce-6-for-developrs.aspx [March 26, 2013]

[17] M. Barabanov, "A Linux-Based Real-Time Operating System", M.Comp.Sci. Thesis, New Mexico Institue of Mining and Technology, Socorro, New Mexico, June 1, 1997

[18] Microsoft Corporation. (2012, February 7). Scheduling Priorities. [online] Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ms685100(v=vs.85).aspx [March 19, 2012]

[19] C. C. Chang, C. J. Lin, “LIBSVM: a library for support vector machnies,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2 issue 3, pp. 27:1-27:27, 2011.

[20] F. Zhang, H. Huang, “Real-Time Recognition of User Intent for Neural Control of Artificial Legs,” MEC’11, New Brunswick, Federicton, NB Canada, August 2011.

[21] F. Zhang, M. Liu and H. Huang, “Preliminary Study of the Effect of User Intent Recognition Errors on Volitional Control of Powered Lower Limb Prostheses,” *Conf Proc IEEE Engineering in Medicine and Biology (EMBC) 2012*.

[22] M. Liu, P. Datseris, and H. Huang, "A prototype for smart prosthetic legs: analysis and mechanical design," *Conf Proc Proceeding of International Conference on Control, Robotics and Cybernetics*. New Delhi, India: IEEE, March 21-23, Vol. 1. pp. 139-143, 2011

[23] Y. Liu, F. Zhang, Y. Sun, H. Huang, "Trust Sensor Interface for Improving Reliability of EMG-based User Intent Recognition", *Conf Proc. IEEE Engineering in Medicine and Biology Society (EMBC) 2011*, Boston, MA, Aug-Sept, 2011.

[24] X. Zhang, Y. Liu, F. Zhang, J. Ren, Y. Sun, Q. Yang, and H. Huang, "On Design and Implementation of Neural-Machine Interface for Artificial Legs," *Industrial Informatics, IEEE Transactions on* , vol.8, no.2, pp.418,429, May 2012.