Towards The Development of a Low-Latency, Biosignal-Controlled Human-Machine Interaction System

Keshav Bimbraw[1], *Member, IEEE*, Mingde Zheng[2], *Member, IEEE*

***Abstract*— With the rise of 5G, IoT, and cybernetic connectivity, the role of human intervention in the digital and physical manipulation of objects, factory automation, and manufacturing has become increasingly critical in meeting unprecedented quality standards. As a result, a reliable Human-centered system construct for enabling seamless and accurate control is urgently needed to interlink human intents with remote targets. In this paper, we report the development of a universal system pipeline, highlighting key enabling modules, and both physical and bioelectrical sensors as input modalities to demonstrate a near-natural motion synchronization between a human and a robotic arm. This effort was exemplified by the formulation of a method to reduce modular and system-level operational latency to achieve congruent human-machine interaction (HMI) through analyzing and simulating common mechanical motions. Furthermore, we explored several efficient machine learning (ML) model that reliably works with a variety of time-series-based biosignals reflective of intents, thus allowing a diversity of sensors to contribute as the system inputs. We believe our system pipeline represents a first step in unveiling otherwise hidden components within Biosignal-Controlled HMI systems and meeting the key challenges will bring us closer to the establishment of a natural, human-intent controlled, remotely operated HMI platform, with applications that extend far beyond major sectors of academia and industry.**

I. INTRODUCTION

Advances in the Internet of Things (IoT), cybernetic and wireless connectivity have led to a rapid increase in digital, physical, and Augmented/Virtual Reality (AR/VR) systems. To effectively interact with them in a rapidly advancing technology landscape, it is vital to develop effective and robust human-machine interaction strategies [1]. Human-machine interaction (HMI) is the design, implementation, and use of technologies to enable humans to interact seamlessly with digital/physical/AR/VR systems. However, as they continue to advance and diversify, their required level of control and manipulation needs have increased drastically. To keep up, we require a new paradigm shift in interfacing with these application domains. And our body provides a natural and flexible means to do so. As an example, two common HMI systems that we use daily are the PC mouse and keyboard. Heralded once as breakthroughs of their days, they have now established themselves as resilient physical interfaces that are difficult to replace and upgrade from. A successful replacement with proper intent-extraction sensors could crown our superior dexterous hands as the ultimate control that translates thoughts to commands, effectively being the robust, reliable, and natural physical interface that lets us manipulate any external applications with utmost freedom and range.

Biosignal-controlled HMI is a rapidly rising new field of research where different sectors (commercial, industrial, military, academia, clinical, etc.) have realized the importance and critical challenges of placing human at the center of their product development processes. This is due to the inference of human intent using the singular or a combination of biosensing modalities. These sensors (ideally non-invasive in nature) ascertain the user's thoughts indirectly and make inferences based on intelligently designed algorithms and machine learning (ML) to translate a thought into a digital command for external control. This is a highly promising area of opportunity for a variety of different applications [2]. Therefore, billions of dollars are being funneled into Biosignal-based HMI research and relevant product development. Input modality flexibility is one of the attractive attributes of its popularity. The input sensors can be based on electrical [2], acoustic [3, 4], optical [5], mechanical [6], biological [7], or physiological [8] changes in the human body to infer the intent, thus facilitating the establishment of a non-verbal-driven HMI. Some examples include haptic technologies such as Meta's AR/VR interfacing prototype [9]; consumer gadgets such as the hand motion detection system in automobiles [10]; surgical robots such as the Davinci Robot as popularly seen in surgical robotic teleoperation [11]; robots for explosive detection and related crisis and military operations [12]; assistive robotic systems in rehabilitation [13, 14] and an extensive and unsorted range of academic research and development devices spanning over a multitude of areas from flying drones via human arm control [15] and beyond. In general, Biosignal-controlled HMI can be classified into four broad categories: a) biopotentials, b) muscle mechanical motion, c) body motion, and d) hybrid signals [16]. Biopotential-driven HMI systems are the most popular amongst them all, thanks to the well-researched surface electromyography (sEMG) sensor in which the signals reflect the myoelectric potential changes and have been extensively used for gestural detection. It is one of the most promising modalities for establishing a natural and intuitive Biosignal-controlled HMI system.

In terms of system operability, these can be divided into two types of systems: dedicated commercial systems and generic experimental systems. Dedicated commercial systems such as Meta's AR/VR hand glove are highly optimized in terms of latency and have a fixed setup. The

[1]Keshav Bimbraw is with Nokia Bell Labs, New Providence, NJ 07974 USA (Phone: 678-436-9426). keshav.bimbraw@nokia.com

[2]Mingde Zheng is with Nokia Bell Labs, New Providence, NJ 07974, USA. mingde.zheng@nokia-bell-labs.com

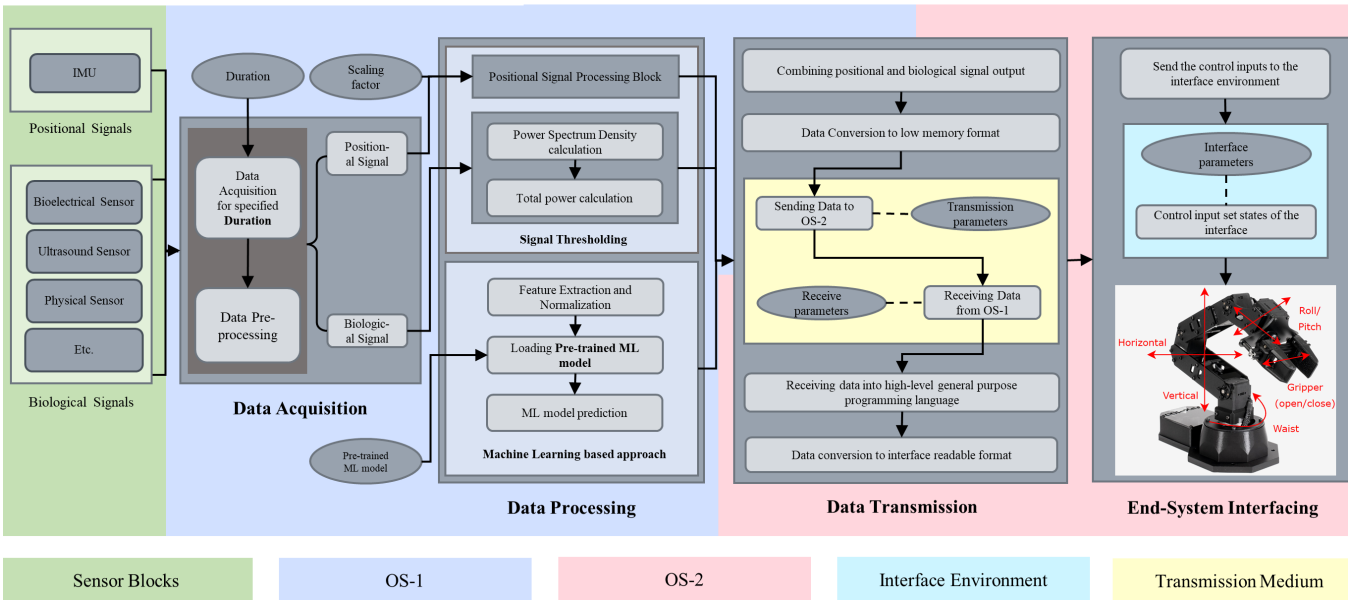


Fig. 1. Biosignal-Controlled HMI system pipeline with key modules and processes in their color-labeled operating platform.

latency in these types of systems usually falls in a range of 100 – 300 ms which makes them suitable for real-time operation. However, these systems are generally expensive, sophisticated, and very specialized. Generic experimental systems such as acoustic signal-based prosthetics and drones are relatively more flexible and are not standardized [3]. The latency of such systems is usually arbitrary and can be greater than 500 ms, making them non-ideal for real-time HMI applications.

Several sEMG-based robotic arm control systems have been developed. Artemiadis, et al. described a system for continuous control of a robotic arm using sEMG signals from the upper limb with a primary focus on the accuracy of estimation of 2-D embeddings based on arm motion in 3-D cartesian space [17]. Cheng, et al. combined sEMG with electroencephalography (EEG) to control a robotic arm with time windows of 2000 ms [18]. Machine learning (ML) and deep learning (DL) based approaches have also been used to estimate hand and wrist movements using sEMG with the primary focus on achieving a high classification accuracy [19, 20, 21]. Dwivedi, et al. combined sEMG with a fiducial marker for a robotic hand control reporting accuracy and algorithmic latency results for telemanipulation [22]. Chen, et al. described a pipeline for controlling a Baxter robot using sEMG [23]. While in the literature, numerous generic sEMG (or a combination of sEMG with other modalities) controlled HMI have been proposed, the focus has primarily been on improving accuracy for gesture classification, or not focusing on overall pipeline latency from start to end for the deployment of such biosignal HMI. Optimizing these generic HMI systems has been challenging due to a lack of guidelines, standardizations, and protocols. Additionally, operational latency has traditionally been viewed as a single systemic issue rather than a series of multi-modular issues that spans all modules of any generic HMI system.

To address this challenge, we present a generalizable structured pipeline with a system-wide latency reduction construct to improve overall system performance. We formulated this construct to examine each module for optimization such as the Data Processing and the Remote Data Transmission module, to demonstrate how low latency must be maintained at the modular and systemic levels. Different data processing techniques were carefully analyzed and selected to work with multi-channel and multi-input sensor modalities with continuous, and real-time signals reflecting live user intent.

In this work, we report the remote and seamless actuation of simple movements of a robotic arm controlled by a human arm using two wireless sensor attachments operating continuously in real-time. Section 2 outlines the methods for system pipeline construction, operational latency construct design, and intent extraction. Section 3 discusses the outcome and performance of the completed end-2-end HMI system, ML with latency performance, and demonstration of the system.

II. Methods

The proposed system pipeline consisted of several key modules based on the author's analysis of other Biosignal-Controlled HMI systems (Fig.1). And unlike other platforms, we generalized them as sub-modules according to their key functions and optimized each module specifically for latency. These included sensor blocks that control all input modalities, and other modules listed as modules for data acquisition, processing, and transfer, user-intent detection via gestural powered ML within the processing module, and end-system interfacing operation module.

A. System Overview

The following section is partitioned according to Fig.1 shown above. Starting with input data from the left-hand side, various sensing modalities are possible. In our experiment, the positional and bioelectrical signals for a predefined time window were acquired from the forearm. These signals were then conditioned using digital signal processing (DSP) and machine learning (ML) based techniques. The processed signals were converted to a low-memory format for low-latency data transfer to the operating system (OS) which controlled the hardware.

1) Data Acquisition: The user intent extraction sensor used for the pipeline development is a multi-channel myoelectric sensor popularly used for hand gestural recognition, and it is known as the Surface electromyography (sEMG) sensor. An inertial measurement unit (IMU) sensor is employed in parallel, taking in both acceleration and orientation information. Collectively, they are embedded in a wearable sleeve format for easy mounting. All programming was implemented in Python. An 8-channel sEMG sensor was employed in our case with a bit-resolution of 16, an amplification factor of 1000, and a sampling frequency of 2000 Hz. The signals were live streamed in multiple channels of the same rate under a timing window between 50 to 500 ms.

2) Data Processing: Following a fixed time window within the system, all acquired input data were conditioned to suppress noises and enhance signal quality (i.e., SNR). The IMU had a positive offset to rectify all the incoming values. The data were then scaled by an arbitrary factor based to be sent as positional commands to the end-system application interface. Myoelectric data were bandpass filtered to lower incoming noises. Following this, two primary methods were explored for extracting intent-specific user motion from the sensors.

a) Signal Thresholding: For rest and motion identification, there were changes in signal intensity which can be extracted from time and frequency domain features derived from the time windows. Instead of relying solely on time domain intensities for identification, frequency domain representation of the signal was more stable. To this end, the power spectrum density (Equation 1) of the signal using a periodogram was acquired for each time window.

$$S\left(\frac{k}{NT}\right) = \left|\sum_{n} x_{_N}[n] \cdot e^{-i2\pi \frac{kn}{N}}\right|^2 \tag{1}$$

where the S is the periodogram for all integers k between 0 and $N-1$ for a parameter T. $x_{_N}$ is a periodic summation defined in equation 2.

$$x_{_N}[n] \triangleq \sum_{m=-\infty}^{\infty} x[n - mN] \tag{2}$$

b) Machine Learning based approach: Numerous ML and DL-based algorithms have been shown in the literature to classify sEMG data. Five different algorithms commonly employed were chosen to build and evaluate our classifiers, namely: AdaBoost-SAMME, nearest neighbor (NNC), linear support vector (L-SVM), gaussian process (GPC), and a multi-layer perceptron neural network (NN).

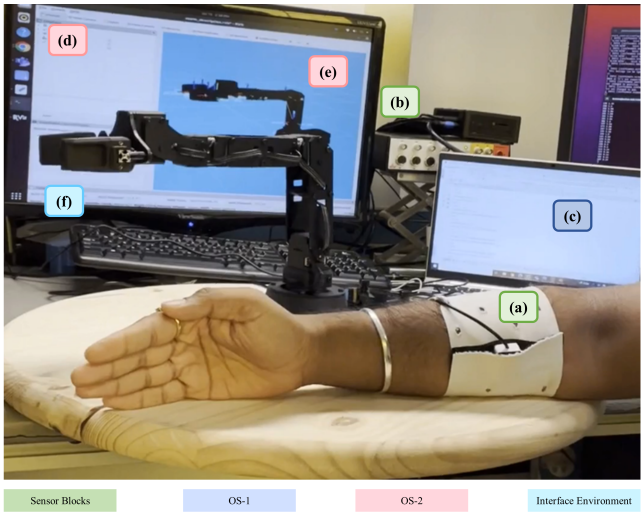


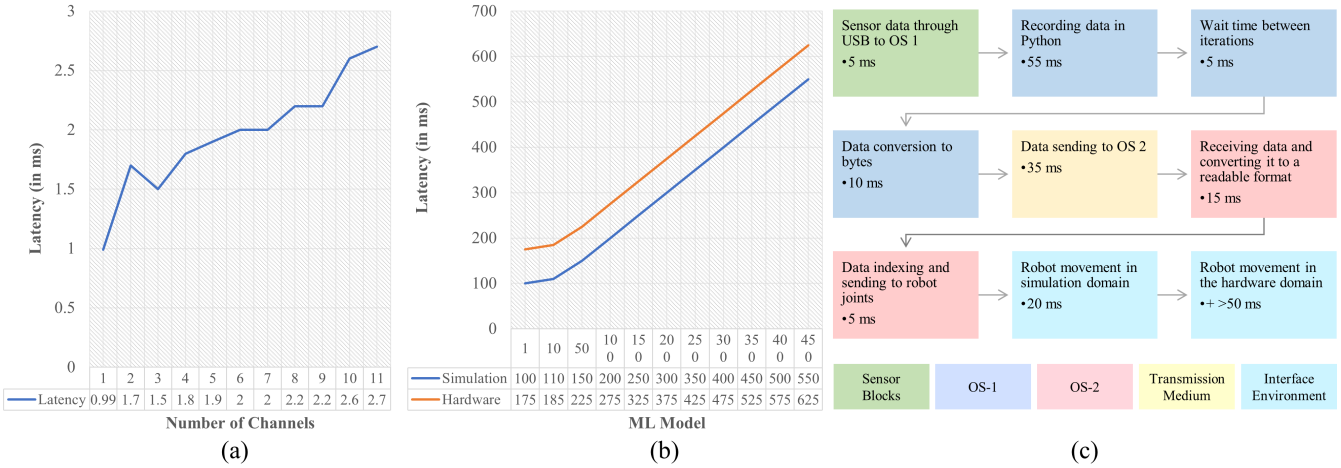
Fig. 2. Different hardware components of the system: (a) Sensors in a custom design armband, (b) The data acquisition system iWorx RA 834, (c) Windows 10 with Python running in PyCharm, (d) Linux Ubuntu 20.04, (e) ROS1 Noetic with RViz real-time robotic simulation, and (f) Pincher X-100 Robotic Arm.

3) Data Transmission: The data obtained from the IMU and the sEMG sensor were combined and subsequently converted into a low-memory data conversion format. The converted data was transferred wirelessly to the operating system containing the robot operating system (ROS) set up for controlling the robotic system.

4) End-System Interfacing: After the low-memory data containing relevant sensor-based movement commands were acquired in the operating system containing the operating system for the end-system interface, it was first converted to a format that was understandable by the interface. For this work, a robotic arm was used as the end-system interfacing application and ROS served as a robotics middleware due to its popularity in the robotics community, low-level device control, and defined message passing between different processes of robot motion. The data received in the system was used to control the robot's position and orientation. This was done by passing the control commands to the joints of the robotic system through ROS nodes. The data sent through the ROS nodes sets the joint parameters of the simulated robotic arm as well as the physical robotic arm.

B. Operational Latency Construct Design

Operational latency is a key challenge that spans all modules in the system pipeline. This is due to the existence of independent and co-dependent processes within each module, therefore regardless of the central processing speed of any OS that the system pipeline was established in, the challenge of targeted low latency hinders the overall performance of the pipeline. We have devised an ingenious solution to resolve this challenging issue.



(a) (b) (c)

Fig. 3. Operational latency results. (a) Latency for data conversion to low-memory format for transfer. (b) Latency for simulation and hardware systems for time windows of lengths 1 ms, 10 ms, 50 ms, etc. (c) For a 50 ms time window, it takes 150 ms for simulated robot control and over 200 ms for physical robot control.

1) Selection of Time Window Size: To acquire IMU and sEMG data, the duration of the time window needed to be optimally chosen. While the positional data from the IMU can give relevant information based on small window sizes of about 10 - 20 ms, the sEMG window size had to be larger to extract the necessary information from the data set. Window sizes from 100 ms - 1000 ms have been reported in the literature [2]. Selection of the right window size was important because increasing it increased the time across all the successive modules of the pipeline. To build a system that operated with the end-system interfacing application seamlessly, it was important to keep the latency of the pipeline less than 300 ms [2]. A window size of 50 ms was hence chosen for the signal thresholding-based method for robot control, and a window size of 500 ms was chosen for the ML-based approach. The prior provided a simplistic approach that one can quickly deploy whereas the latter provides the means for more sophisticated applications.

2) Conversion to a Low-Memory Format: After the relevant data was obtained from both IMU and sEMG sensors, they were concatenated in the form of a list object with two arbitrary values. They were then scaled between 0 and 255 so they could be converted into a single unsigned byte. This was necessary for converting the data from a high-to-low-level fundamental data unit. This low-memory occupancy compared to other data types makes it ideal for real-time data transfer. Following this conversion, the byte array was sent over the user diagram protocol (UDP) to the end-system interfacing application environment containing the robot control operation established through ROS. During a function call for the data, it was then first converted back to a list object with two values in a higher-level format, suitable for conversion to the values interpretable by the end-system interfacing application, which, in our case, is the robotic arm.

3) Wireless Data Transfer: There are several options using which data can be transferred wirelessly, with a heavy reliance on Wi-Fi-based data transfer approaches. The primary transport-layer protocols for it are UDP and transmission control protocol (TCP). UDP provided an uninterruptible datagram connection between systems and applications. For TCP, a connection is established before the data transmission begins. UDP was chosen for our application due to the speed of transfer, and continuous streaming of data which makes it ideal for real-time teleoperation for robotic systems. The analysis of the UDP packet loss has been thoroughly discussed in Section III.A.

C. Data Processing and Intent Extraction

The IMU data was filtered using a band-pass filter with the low-pass frequency set to 0.5 Hz and the high-pass frequency set to 30 Hz. The ML sub-module of the system pipeline took in pristine sEMG data from the previous DSP module, where the sEMG data windows were filtered using a band-pass filter with the low-pass frequency set to 10 Hz and the high-pass frequency set to 10 kHz. After the filtered data was acquired, features were extracted from the sEMG time windows, and machine learning based algorithms were used to train and evaluate models for classifying hand movement.

1) Feature Extraction: For extracting meaningful information, 14 different features were chosen for the analysis. These features were acquired for each time window. Of these, 12 were time-domain features and 2 were frequency-domain features, which are the signal's total and mean power. The time domain features chosen were zero crossing, waveform length, mean absolute value, root mean square, V-order (orders = 1, 2, 4, and 8), slope sign change, Willison amplitude, variance, and simple square integral. The time and frequency domain features were combined and normalized across each feature. They were then used as an input vectors to the machine learning algorithms.

2) Machine Learning: The data for the hand motions (rest and motion states) was acquired in a supervised learning paradigm with the time windows being labeled automatically as time progressed. Scikit-learn library was used to train and

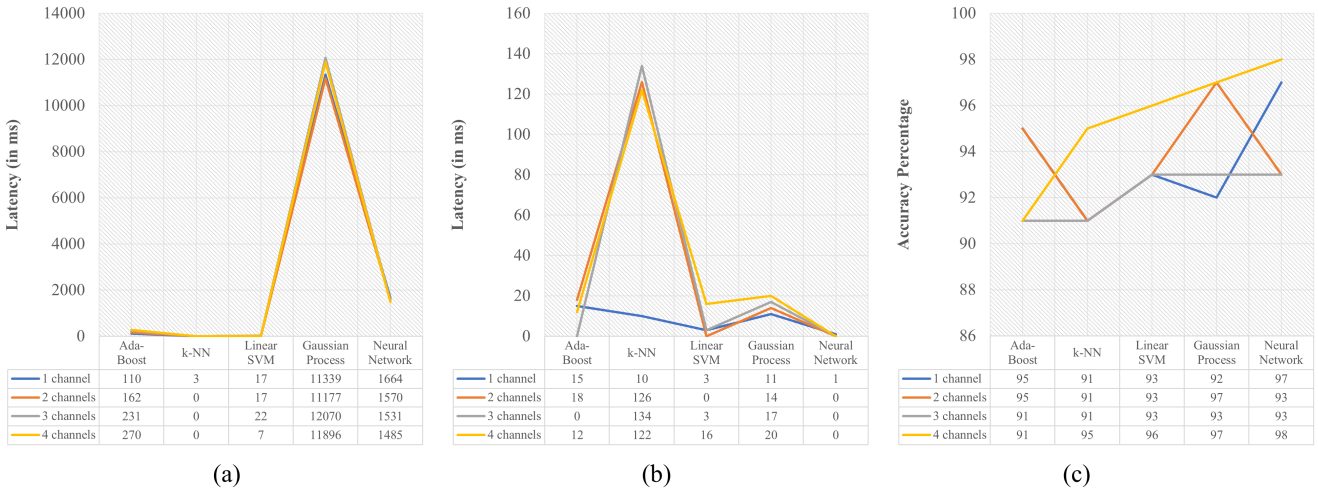


Fig. 4. Machine Learning algorithm comparison in terms of (a) Training time, (b) Inference time, and (c) Accuracy percentage.

evaluate the models in Python [24]. Five different algorithms were analyzed and selected for comparison in our operation: AB, k-NN, L-SVM, GP, and NN-MLP. In terms of model training and evaluation, a total of 12.5 minutes of total data was acquired over 5 trials of 300 at 500 ms, given a total of 150000 ms or 2.5 minutes each. In each trial, there were 500 ms of successive rest and motion states. 6 seconds of data at the start of every file is discarded. The subject had their arm straight up and they were sitting on a chair with their feet firmly on the ground. A total of 12 minutes of 4-channel sEMG data was used to evaluate the systems, with a 75% train-test split leading to 9 minutes as the training set and 3 minutes as the test set.

D. Hardware Interfacing

Fig. 2 shows the different physical components of the system. For signal acquisition, iWorx RA 834 biosignal research workstation was employed [25]. A multi-channel iWire-BIO8 external module was used for sEMG signal amplification, and an iWire-IMSx was used for IMU signal conditioning. The sensors were attached to the forearm using a wearable sleeve. PincherX-100 research-grade robot arm was used as the end-system interfacing [26]. As part of the Interbotix X-series, it uses dynamixel X-series smart servo motors. Two different operating systems were used, a Windows 10 system and an Ubuntu Linux 20.04 system. The robot operation was implemented using Interbotix's robot control API and it runs on ROS1 Noetic.

III. Results & Discussion

The establishment of a universal system pipeline for Biosignal-controlled HMI was to enable transparent development and standardization of methods so a reliable robotic system under flexible human control. There are numerous technical challenges depending on the end-system interfacing applications, however, we believe many key challenges arise from common modules shared by all these systems. As an outstanding demonstration shown here, we identified operational latency as the primary obstacle across several key co-dependent modules.

A. Operational Latency Reduction Outcome

In the system pipeline shown in Fig. 1, there were several components that led to time lags. The latencies in these components were optimized by (a) Intelligently splitting the operating systems, (b) Selecting the right window size, (c) Choosing a memory-efficient route of data compression and transfer, and (d) Using a low-latency data transfer protocol for a natural, real-time operation.

Fig. 3(a) shows the latency for an increasing number of channels for data conversion to bytes. For an increasing number of channels from 1 to 11, the latency increased from 1 ms to 3 ms. Thus, the low-memory data conversion does not take more than 3 ms even for 11-channel data, thereby making it suitable for data compression in wireless transfer. For wireless transfer with UDP, there were issues with data packet loss, but because of low-latency transfer, it was the preferred choice. Based on experiments with a small number of data samples (102 - 104 samples of low memory byte arrays), it was found that 70% of data is received, with a 30% packet loss. For a larger number of data samples (greater than 106 samples of low memory byte arrays), 90% of data is received with a 10% packet loss. To reduce the data loss even further, a small wait time (1 ms) was added between successive samples for data transfer. This improved the data transfer significantly with over 99% of data received for both small and large data samples. This communication was established over Wi-Fi with an average speed of 1000 Mbps. With other Wi-Fi speeds, the results might be different.

Fig. 3(b) shows the system latency for robot simulation and hardware control under real-time teleoperation. It was found that system pipeline latency increased linearly with the size of the data acquisition time window. It takes more time for the hardware to respond because of the robot's physical

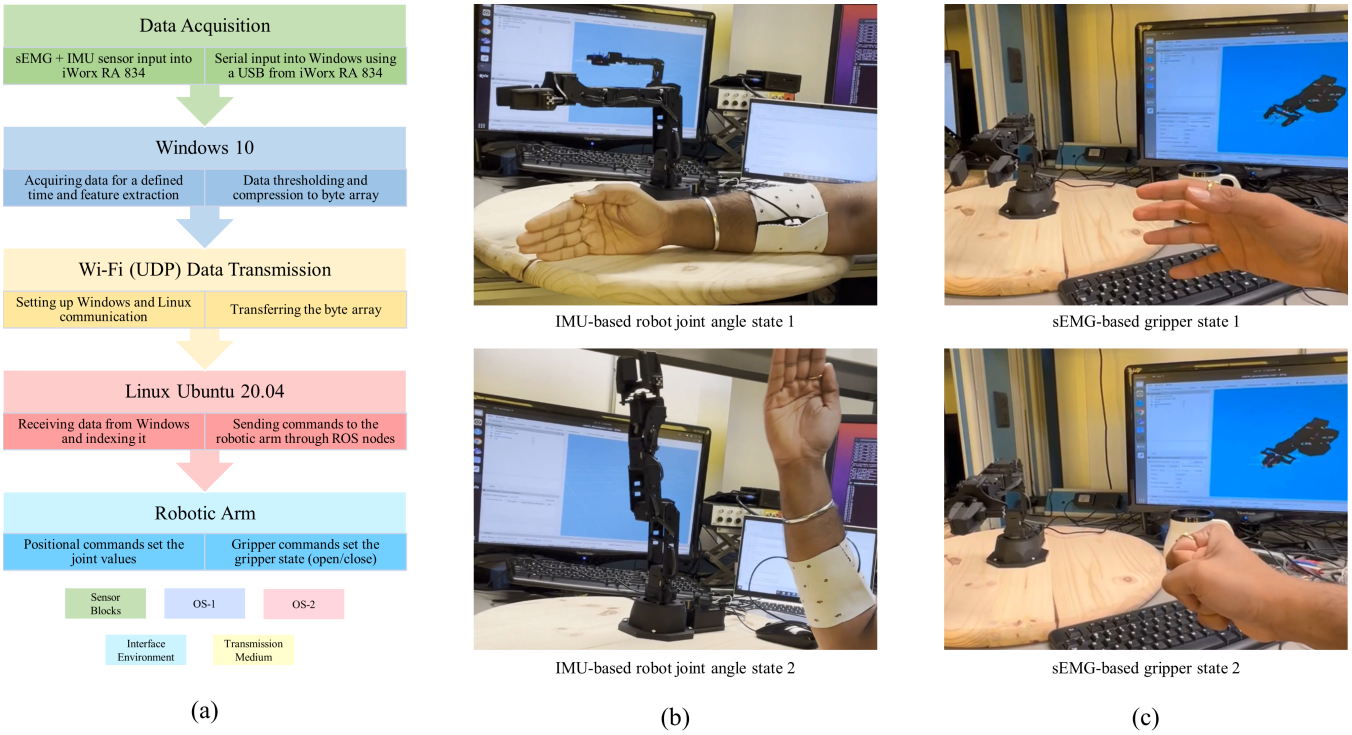


Fig. 5. Figure for demonstration: (a) Data flow chart for demonstration with the color legends at the bottom, (b) IMU-based joint angle control, and (c) sEMG-based robotic gripper control.

limitation to reach from one point to the other. Fig. 3(c) In terms of total latency under each module of the pipeline, we devised a meticulous calculation 'crawler' by navigating and calculating the value through each system pipeline module as shown in the flowchart of Fig. 3(c). The data acquisition time of 55 ms is based on a 50 ms time window with 5 ms of data processing. The wait time between iterations of 5 ms was introduced by the software wait time of 1 ms and an additional acquisition reading time that it took for the next data window to be read. The byte conversion at most took 10 ms inclusive of it being a multi-dimensional byte array. For our Wi-Fi speed, the data transfer, reception, and conversion into a readable format take no more than 50 ms. Because all the robot communication was established over ROS, it took 5 ms to send commands to the robot joints. It took 20 ms for the robot to move in the simulation. Because the hardware had its own movement limitations, it took 50 - 75 ms for the robot to move from one position to the next assuming the joint movement was less than 0.1 radians. Overall, for a 50 ms time window, it took 150 ms for the robot to move in the simulation domain and 200 - 225 ms for the robot to move in the hardware domain.

To demonstrate the latency enhancement on one of the co-dependent ML modules, we performed an analysis specifically to show the time taken for training different machine learning algorithms. In Fig. 4(a), we found that k-NN took the least time while GP took the longest. As the number of channels increased, NN-MLP and L-SVM's training time decreased, while for AB it clearly increased. Fig. 4(b) shows the inference time which is the time required for a final decision after an input feature vector is loaded into a trained model for classification. As can be seen from the figure, k-NN took the longest time for prediction, while NN-MLP took the least inference time. As the number of channels increased, inference time increased for GP and L-SVM. No clear trend was observed for AB and k-NN. From these results, NN-MLP stood out in terms of the low latency of inference without having the lowest training time.

B. Data Processing and Intent Extraction outcome

Fig. 4(c) shows the accuracy percentage for 5 different ML algorithms for 4-channel sEMG, with the performance evaluated over a previously unseen test set of 3 minutes using models trained on 9 minutes of training data. For 4-channel data, NN-MLP outperformed other algorithms with a 98% accuracy. GP performed the second best with an accuracy of 97%. To have a low-latency system, it was important to consider inference latency and accuracy percentage. NN-MLP performed the best with a 4-channel latency of less than 1 ms and an accuracy of 98% over the test set.

C. Performance Demonstration

Fig. 5 shows the different aspects of the performance demonstration. Four different performance demonstrations in form of a video have been attached to the paper. For setting the basis for baseline comparison, a PlayStation 4 controller was used to send basic joint movement commands to the robot arm. This established the basic proof of concept

Linux-only control pipeline setup. Then, the robot joints were enacted in a low-latency fashion using the IMU values. The IMU values were mapped to different joints of the robotic arm and performed actuation. Next, using the power spectrum density based total power thresholding approach, the sEMG-based robot gripper control was achieved. Finally, the IMU and sEMG-based simultaneous control of robot joints and gripper was achieved with a total system latency of 200 ms, allowing us to realize the potential of our latency reduction construct for real-time use, and for enabling applications ranging from basic pick-&-place to advanced dexterous tasks.

IV. CONCLUSIONS

Human intervention in the digital space will undoubtedly be the central topic of discussion in the transition toward next-generation IoT, cybernetics, and 6G development. The ability to augment our physical body to have the capability to interact with the digital world seamlessly and accurately would be the key to realizing the essential role humans play in the future. To that end, we believe the collective and transparent development of a robust system pipeline to enable a control mechanism for the teleoperation of any end-system interfaces will enhance our physical capabilities. The development of this low-latency system pipeline moved us closer to that goal, and it helps researchers and industry professionals to develop remotely operated HMI platforms capable of natural and seamless interaction.

ACKNOWLEDGMENT

The authors would like to thank the researchers in the Artificial Intelligence Research Lab at Nokia Bell Labs for their support.

REFERENCES

[1] Marcus K Weldon. *The future X network: a Bell Labs perspective*. CRC press, 2016.

[2] Mingde Zheng, Michael S Crouch, and Michael S Eggleston. "Surface Electromyography as a Natural Human-Machine Interface: A Review". In: *IEEE Sensors Journal* (2022).

[3] Keshav Bimbraw et al. "Towards Sonomyography-Based Real-Time Control of Powered Prosthesis Grasp Synergies". In: *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE. 2020, pp. 4753–4757.

[4] Keshav Bimbraw et al. "Prediction of Metacarpophalangeal joint angles and Classification of Hand configurations based on Ultrasound Imaging of the Forearm". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 91–97.

[5] Jingjing Guo et al. "Soft and stretchable polymeric optical waveguide-based sensors for wearable and biomedical applications". In: *Sensors* 19.17 (2019), p. 3771.

[6] Samuel Wilson and Ravi Vaidyanathan. "Upper-limb prosthetic control using wearable multichannel mechanomyography". In: *2017 International Conference on Rehabilitation Robotics (ICORR)*. IEEE. 2017, pp. 1293–1298.

[7] Yang Zhang and Chris Harrison. "Tomo: Wearable, low-cost electrical impedance tomography for hand gesture recognition". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 2015, pp. 167–173.

[8] Dilip Chakravarthy Kavarthapu and Kaushik Mitra. "Hand Gesture Sequence Recognition using Inertial Motion Units (IMUs)". In: *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*. IEEE. 2017, pp. 953–957.

[9] Carlos Bermejo and Pan Hui. "A survey on haptic technologies for mobile augmented reality". In: *ACM Computing Surveys (CSUR)* 54.9 (2021), pp. 1–35.

[10] Eshed Ohn-Bar and Mohan Manubhai Trivedi. "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations". In: *IEEE transactions on intelligent transportation systems* 15.6 (2014), pp. 2368–2377.

[11] C Staub et al. "Human-computer interfaces for interaction with surgical tools in robotic surgery". In: *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE. 2012, pp. 81–86.

[12] Ioannis Kostavelis and Antonios Gasteratos. "Robots in crisis management: A survey". In: *International Conference on Information Systems for Crisis Response and Management in Mediterranean Countries*. Springer. 2017, pp. 43–56.

[13] Abolfazl Mohebbi. "Human-robot interaction in rehabilitation and assistance: a review". In: *Current Robotics Reports* 1.3 (2020), pp. 131–144.

[14] Shrey Pareek et al. "MyoTrack: Tracking subject participation in robotic rehabilitation using sEMG and IMU". In: *2019 International Symposium on Medical Robotics (ISMR)*. IEEE. 2019, pp. 1–7.

[15] Mylena McCoggle et al. "Applying Biosensors' Electromyography Signals through an Artificial Neural Network to Control a Small Unmanned Aerial Vehicle". In: *International Journal of Electrical and Computer Engineering* 16.5 (2022), pp. 77–80.

[16] Daniele Esposito et al. "Biosignal-Based Human–Machine Interfaces for Assistance and Rehabilitation: A Survey". In: *Sensors* 21.20 (2021), p. 6863.

[17] Panagiotis K Artemiadis and Kostas J Kyriakopoulos. "EMG-based control of a robot arm using low-dimensional embeddings". In: *IEEE transactions on robotics* 26.2 (2010), pp. 393–398.

[18] Liwei Cheng et al. "Robotic arm control system based on brain-muscle mixed signals". In: *Biomedical Signal Processing and Control* 77 (2022), p. 103754.

[19] Kaichi Fukano et al. "Deep Learning for Gesture Recognition based on Surface EMG Data". In: *2021 International Conference on Advanced Mechatronic Systems (ICAMechS)*. IEEE. 2021, pp. 41–45.

[20] Dapeng Yang and Hong Liu. "An EMG-based deep learning approach for multi-DOF wrist movement decoding". In: *IEEE Transactions on Industrial Electronics* 69.7 (2021), pp. 7099–7108.

[21] Ulysse Côté Allard et al. "A convolutional neural network for robotic arm guidance using sEMG based frequency-features". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 2464–2470.

[22] Anany Dwivedi et al. "Combining electromyography and fiducial marker based tracking for intuitive telemanipulation with a robot arm hand system". In: *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2019, pp. 1–6.

[23] Minjie Chen and Honghai Liu. "Robot arm control method using forearm EMG signals". In: *MATEC Web of Conferences*. Vol. 309. EDP Sciences. 2020, p. 04007.

[24] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[25] 2022 iWorx Systems Inc. *IX-RA-834 10+ Channel Recorder and Stimulator*. URL: https://iworx.com/products/data-recorders/ix-ra-834-10-channel-recorder-and-stimulator/?v=7516fd43adaa. (accessed: 08.29.2022).

[26] 2022 Trossenrobotics.com. *PincherX 100 Robot Arm - X-Series Robotic Arm*. URL: https://www.trossenrobotics.com/pincherx-100-robot-arm.aspx. (accessed: 08.29.2022).