IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 13 July 2023, accepted 5 August 2023, date of publication 11 August 2023, date of current version 21 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3304544

RESEARCH ARTICLE

LibEMG: An Open Source Library to Facilitate the Exploration of Myoelectric Control

ETHAN EDDY[1,2,3], (Member, IEEE), EVAN CAMPBELL[1,2], (Member, IEEE), ANGKOON PHINYOMARK[1,2], (Member, IEEE), SCOTT BATEMAN[3], AND ERIK SCHEME[1,2], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of New Brunswick, Fredericton, NB E3B 5A3, Canada
[2]Institute of Biomedical Engineering, University of New Brunswick, Fredericton, NB E3B 5A3, Canada
[3]Human-Computer Interaction Laboratory, University of New Brunswick, Fredericton, NB E3B 5A3, Canada

Corresponding author: Ethan Eddy (eeddy@unb.ca)

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

ABSTRACT Myoelectric control has been used predominantly in the field of prosthetics, but is an increasingly promising hands-free input modality for emerging consumer markets such as mixed reality. Developing robust machine learning-enabled EMG control systems, however, has historically required substantial domain expertise. This has presented a significant barrier to entry for researchers, impeded progress in EMG-based interaction design, and contributed to the perception that such systems lack the robustness and intuitiveness required for real-world use. To overcome these challenges, we present LibEMG, an open-source Python library for performing offline EMG analyses and developing online EMG-based interactions. By abstracting the challenges and nuances surrounding myoelectric control, including hardware interfacing, data acquisition, feature extraction/selection, classification, post-processing, and evaluation, we eliminate many of the significant barriers limiting the exploration of this technology. Combining expertise from the prosthetics and human-computer interaction communities into a shared library, extensive examples, and documentation, we provide researchers with an accessible tool to accelerate research and improve reproducibility in myoelectric control. In doing so, we aim to facilitate the exploration of this technology, particularly outside prosthesis control, to unlock its potential as a widely applicable hands-free input modality. Documentation: https://libemg.github.io/libemg/.

INDEX TERMS EMG, electromyography, toolkit, library, myoelectric control, gesture recognition.

I. INTRODUCTION

Myoelectric control – the control of a device using the electrical signals generated during muscular contractions – has had a long and successful history in prosthetics for the control of powered prostheses. By placing surface electrodes directly on the skin (i.e., surface electromyography (EMG)), the electrical signals produced during muscular contractions can be recorded, processed, and passed to machine learning algorithms [1]. These algorithms can then differentiate between contractions, effectively turning the human body into a readily-available controller, where muscle inputs are associated with interactive device commands.

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Mei.

Although the first myoelectric control systems were developed in the 1960s for prosthesis control [2], the technology did not garner interest for 'general-purpose applications' until the early 2000s [3], [4], [5]. Notably, in 2014, the landscape of myoelectric control was drastically altered through the release of the Myo Armband, the first widely available and affordable surface EMG wearable band [6]. This commercially available device, which had pre-built software for enabling gesture recognition, paved the way for an influx of EMG research in human-computer interaction (HCI), propelling myoelectric control to new bounds. This can be attributed to the fact that this device lowered the entry barrier and enabled non-domain experts (such as HCI researchers) to explore this technology for various interactive systems. However, with the discontinuation of the Myo Armband,

no 'killer' application on the horizon, and relatively immature gesture recognition capabilities, the progression and interest in consumer applications of myoelectric control dwindled, at least until recently.

Driven by the growing need to compute in rich environments such as mixed reality, myoelectric control has regained interest as a general-purpose input modality [7]. Its potential convenience, intuitiveness, and subtlety make it a particularly attractive solution for readily-available input within ubiquitous computing environments. However, while this technology is potentially more applicable than ever, significant barriers exist to its adoption and exploration, particularly outside prosthetics. For the continued development, progression, and maturation of myoelectric control for general-purpose use, it needs to once again be made accessible to a broader research audience. Correspondingly, we present LibEMG, an open-source toolkit for exploring myoelectric control.

LibEMG is an open-source Python library that aims to facilitate the development of myoelectric control systems, particularly for non-domain experts. To alleviate the barriers hindering the exploration of this technology, LibEMG abstracts many challenges associated with developing myoelectric control systems, including data processing, hardware interfacing, feature extraction/selection, classification, post-processing, and evaluation. Through its creation, LibEMG makes this space accessible to a broader range of researchers and practitioners, regardless of their expertise with EMG, signal processing, and machine learning. Moreover, LibEMG's documentation describes the API and provides explanations, background, and pointers that additionally support expertise acquisition. We also developed LibEMG based on the current state-of-the-art practices from the myoelectric prosthesis literature and enable extended functionality and control for more expert users. Through this flexibility, we hope that LibEMG will serve as a powerful tool that can be adopted by developers, students, and researchers whose experience with EMG ranges from novice to expert.

II. BACKGROUND

A. ELECTROMYOGRAPHY

Electromyography (EMG) is a measurable representation of the electrical activity resulting from the contraction of a muscle. While multiple methods exist to measure these physiological signals, such as through skin-implanted needles and wires [8], surface EMG is the most common due to its convenience and non-invasiveness [1]. By measuring, processing, and passing the signals to machine learning algorithms, gestural intent can be converted to interactive commands for hands-free myoelectric control.

Initial myoelectric control schemes (i.e., conventional control) took a one-muscle one-function approach for device control [1]. By placing electrodes on antagonistic muscle pairs (e.g., the flexor and extensor forearm muscles) and activating a controlled device when the amplitude of the signal passed a predefined threshold, amputees could control one function of a prosthesis (e.g., opening or closing a hand). While these initial systems were simple and reasonably effective, they required complicated mode-switching (i.e., the co-contraction of both antagonistic muscles) to increase the input space (e.g., to switch a prosthesis between wrist rotation and hand-open/close modes). In turn, learning to use conventional control systems effectively was challenging and required extensive training [9]. To increase the intuitiveness and effectiveness of these control schemes, researchers turned to approaches based on "pattern recognition" (i.e., machine learning) [10], [11], [12], [13].

By using multiple electrodes to measure EMG across several muscle sites [8], pattern recognition leverages the synergistic behaviour of muscles when they are contracted. Machine learning algorithms can categorize repeatable and separable muscular inputs into classes (i.e., unique contractions), which can act as input commands to a device. For prosthesis control, these classes have traditionally corresponded to physiologically appropriate inputs (i.e., where the recognized contraction corresponds to a similar device control). For example, the hand open and close classes would correspondingly open and close the prosthetic device. Continuously classifying these contractions based on predefined window size and increment (i.e., update rate) to control a prosthesis in real-time, as proposed by Englehart and Hudgins [10], is the foundation of today's commercially available pattern recognition-based myoelectric control (e.g., Coapt[1] and Ottobock[2]).

Modern day continuous myoelectric control schemes can be summarized in four stages, as highlighted by Scheme and Englehart [1]: data preprocessing, windowing, feature extraction, and classification. First, EMG enters the system, and unwanted noise (i.e., signal contaminants), such as powerline interference and motion artifact, is filtered. Next, due to the stochastic and random nature of EMG, it is split into windows (i.e., a predefined amount of data samples) with specified increments (i.e., the time that elapses before capturing the next window). For prosthesis control, the window size and increment are short [14] (in the order of milliseconds) to enable continuous and constant control over a device, enabling amputees to micro-adjust their prosthesis. In the third step, features are computed from each window to increase the information density of the underlying signal before being passed into a machine learning algorithm that differentiates between them and outputs a class label corresponding to one of the N classes used to train the system. Finally, these outputs are converted into device commands and combined with proportional velocity-based control approaches [15] – where contraction intensity dictates the device speed – enabling amputees to control their prostheses.

[1] https://coaptengineering.com
[2] https://www.ottobock.com/

B. EMG IN HCI

The success of leveraging EMG for prosthesis control ultimately led to its exploration by HCI researchers in the early 2000s [3], [4], [5], [16], [17], [18]. Before this, surface EMG devices were cost-prohibitive, tethered, and primarily used for medical purposes. Additionally, the release of commercially available wearable devices, such as the Myo Armband in 2014 [6], made the exploration of this technology for novel hands-free interactions possible for the broader research community. Since then, myoelectric control has been leveraged for various general-purpose applications such as piano augmentation [19], drone control [20], gaming [9], hands-full input [21], and mixed reality interactions [22]. Simultaneously, the HCI community has also contributed novel work and applied their expertise to inform prosthetics-related research through training tools [9], [23], [24], alleviating phantom limb pain [25], and improved prosthesis design [26]. However, regardless of the uptake and interest in this technology, both commercially [27] and academically, the adoption of myoelectric control for general-purpose use is still limited [7].

From understanding the physiology underlying the stochastic EMG signal, the processing techniques required for its interpretation and the machine learning needed for classifying user intent, myoelectric control is inherently complex. In turn, developing control systems with real-world viability is challenging. This presents a significant barrier, especially for researchers and system developers unfamiliar with this technology, and may even dissuade some from pursuing it completely. Recent work by Eddy et al. [7] supports the notion that a considerable challenge to the adoption of EMG is that its robust exploration is challenging, and designing control systems requires considerable EMG-specific domain knowledge. In turn, they emphasize that to facilitate the eventual adoption of EMG for general-purpose use, its principled and well-informed exploration must be made accessible to the broader research community, including HCI researchers. In particular, they highlight the need for toolkits that enable non-domain experts to explore this technology. This would enable research practices around interactive general-purpose EMG applications to mature, and it would allow researchers outside prosthetics to contribute and apply their expertise to advance prosthetics-focused research in new directions.

C. TOOLKITS

One solution to improve accessibility for complex areas of research is through the development of toolkits. Ledo et al. defines toolkits as "*generative platforms designed to create new interactive artifacts, provide easy access to complex algorithms, enable fast prototyping of software and hardware interfaces, and/or enable creative exploration of design spaces*" [28]. Many toolkit examples exist for facilitating research, such as for exploring electrical muscle stimulation [29], haptics [30], mixed reality development [31], autonomous driving [32], and outlier detection [33]. More specifically, several toolkits have been developed for gesture [34], [35], [36] and activity [37] recognition using IMU sensors and cameras. However, to the best of our knowledge, no toolkits (other than EMBody [38] and BioPatRec [39]) exist for facilitating the development of myoelectric control systems.

Although an excellent starting point, EMBody's contribution stands primarily in its open-access hardware design and goal of enabling HCI researchers to explore EMG as an input modality. However, EMBody's limitation is that it was created with a predefined and rigid myoelectric control system. For example, by default EMBody only supports a single classifier (SVM) and feature (root mean square). We believe that a library for designing and developing myoelectric control schemes should include robust signal processing, feature selection and extraction techniques, algorithmic implementations, and effective evaluation opportunities. Moreover, it should be usable and facilitate the thorough exploration of myoelectric control through shared datasets and hardware integration.

BioPatRec, an EMG toolkit released in 2013, was an early attempt within the prosthetics community to create a shared platform for myoelectric prosthesis control research. Correspondingly, the release of this toolkit and its associated dataset led to many contributions within the prosthetics field, highlighting the impact these tools can have on a research community. However, while this toolkit supports offline and online analysis, various classifiers, and features, its rigid UI-based structure, lack of support for commercial hardware, focus on prosthesis control, and requirement for Matlab licensing mean its adoption outside of prosthesis control research (such as in HCI venues) never occurred.

In this work, we lean on EMBody and BioPatRec as conceptual starting points and extend their use in new directions guided by experts from both the prosthetics and HCI fields. We strive to expand upon the foundational work of both tools and provide a new tool for enabling robust EMG-based interactions through a feature-rich library grounded in the latest research in prosthetics and inspired by widely used libraries such as scikit-learn [40] and LIBSVM [41].

III. LibEMG DESIGN

We designed LibEMG with the primary goal of enabling researchers outside prosthetics to explore myoelectric control as a robust and reliable hands-free input modality for general-purpose applications. We also hope that it serves as a new scalable platform upon which ongoing advancements in myoelectric control can be built. By combining the expertise of a prosthetics lab and an HCI lab into a shared library, we have created a common resource for exploring and evaluating EMG-based interactions. After outlining the pipeline architecture, each module was designed, developed, and tested independently. For several modules, including the Feature Extraction, Feature Selection, and Filtering modules, source code used as part of previously published work from our

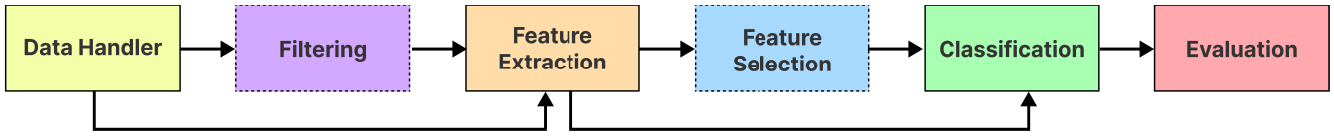


FIGURE 1. The six main steps (modules) of the pipeline for developing EMG-based interactions using LibEMG. The dashed borders indicate that the module is an optional step in the pipeline.

labs was leveraged as a starting point. Additionally, through the design and development of ongoing studies, the set of examples (see Section V), and feedback from others within and outside the lab, the library incrementally evolved until we felt it had the reliability, flexibility, and usability required for deployment. We hope that through the release of this library, myoelectric control can continue to mature to the level required for its widespread adoption within the consumer market.

A. DESIGN GOALS

In this section, we highlight and explore the four design goals that guided the development process of LibEMG: usability, accessibility, reliability, and reproducibility.

1) USABILITY

Several factors, from signal processing to machine learning, make the design and development of EMG-based interactions challenging. LibEMG strives to abstract these complications and reduce the domain knowledge required for the robust exploration of this technology. Moreover, it simultaneously enables the development of more complicated interactions for expert users that may have more familiarity with myoelectric control. With this in mind, the goal was to develop a highly usable library to support a wide spectrum of potential end users (from novices to experts). Correspondingly, a significant usability requirement of LibEMG was a robustly documented API[3] with several working demo systems. We modeled this documentation after scikit-learn [40], a popular Python library for machine learning, and examples from previous works published in IEEE and ACM venues. Moving forward, this documentation (including the user guide, API, and examples) will act as a living extension of this paper. Additionally, we promote usability by having default integration with common EMG devices (initially we support four: the Myo Armband, Delsys Trigno/Avanti, SIFI Bioarmband, and gForcePro+ EMG Armband), and inclusion of an initial collection of pre-recorded datasets. Ultimately, this reduces the overhead typically required for exploring this technology and enables the community to focus on designing interactive systems rather than on the control schemes themselves.

2) ACCESSIBILITY

To ensure an accessible library for all interested researchers, LibEMG is implemented in Python, is fully open source, requires no licenses, is publicly documented, and is operating system agnostic. We chose Python as it is free, easy to use, popular, and has become a common environment for machine learning – a fundamental component of deciphering user intent from EMG. Not only is all the code[4] open-source, but the library itself is available for download via PyPI (i.e., pip install libemg). Additionally, due to the modular nature of this library, we have made each pipeline component accessible to developers. Therefore, the entire pipeline pictured in Figure 1 or any of the individual blocks can be leveraged based on the developer's needs.

3) RELIABILITY

One major limitation of previous work is that control systems have often lacked the perceived robustness and intuitiveness required for consumer-level use. Due to the challenges of processing and handling EMG, developing reliable control systems is inherently challenging. Therefore a goal for this library was to facilitate the design of myoelectric control systems beyond the standards set of other EMG toolkits, like EMBody and BioPatRec. To achieve this goal, LibEMG was designed and developed by HCI researchers experienced with numerous interactive technologies, system evaluation, and toolkit research. Additionally, it was co-developed by domain experts with years of experience implementing myoelectric control systems for powered prostheses. By providing a reliable tool for designing and developing myoelectric control systems, shared across both HCI and prosthetics, we hope this technology can finally mature to a level where it can complement or even compete with other hands-free input modalities such as computer vision or IMU-based inputs.

4) REPRODUCIBILITY

While reproducibility has recently garnered increased attention from the research community [42], [43], it is inherently challenging for EMG-focused research for several reasons. Different hardware, datasets, algorithmic implementations, and signal-processing techniques make the validation, corroboration, and extension of work difficult. This is especially true when the data and code are unavailable, making the reproduction of results challenging or sometimes impossible. Inevitably, this makes building off of previous work difficult and limits the uptake and maturation of this technology. LibEMG strives to bridge this gap and promotes reproducible research in several ways. First, we include several datasets (see Section III-C2) to facilitate the comparison of

[3] https://libemg.github.io/LibEMG/

[4] https://github.com/libemg/libemg

techniques on a standardized baseline. Secondly, the development pipeline acts as a form of documentation for understanding the implementation details of a particular control system. Thirdly, by providing developers with a shared tool for recording data, we can improve consistency among datasets, improving the reproducibility of results. We hope that through the development of a shared library, LibEMG can indirectly promote the advancement toward more reproducible research around myoelectric control.

B. CORE MODULES

As depicted in Figure 1, LibEMG consists of six core modules: Data Handling, Filtering, Feature Extraction, Feature Selection, Classification, and Evaluation. These steps are consistent with those proposed by Eddy et al. in [7] for developing myoelectric control systems for general-purpose applications. In this section, we explore the functionality and purpose of each of these modules. Note that this section is simply an overview of each module, highlighting a subset of their functionality. A more thorough overview of each module, including additional functionality, can be found in LibEMG's API documentation.

1) DATA HANDLER

When developing myoelectric control systems, EMG data are required to train models, make real-time predictions, and run analyses. The job of the DataHandler module is to deal with and process all of this data, which is either stored in offline datasets (through the **OfflineDataHandler**) or streamed in real-time (through the **OnlineDataHandler**).

When performing offline analyses, a significant up-front time investment is writing the code to parse through a dataset, accumulate the contents of each data file, and store them in memory correctly. This is especially true for publicly available EMG datasets, as they often adopt completely different file and folder structures and naming conventions. We have greatly reduced the challenges associated with this process through the **OfflineDataHandler**. Using regular expressions, we simplified the process of accumulating data files and extracting metadata when it is contained in the file path. For example, in *dataset/participant_1/training/R_1/C_1.csv*, we can accumulate all files and then split the data into different participants, training/testing sets, reps, and classes, since all of this metadata exists in the file path. Also built into the OfflineDataHandler is the ability to split data into windows of a specified length and increment, as is typical for myoelectric control. Although this tool is not fully compatible with all data formats, we analyzed and took motivation from several pre-existing datasets. Data in other formats, such as .mat files, can be converted to compatible datasets by converting the files to a compatible (.csv or .txt) format. We took this approach to interface the Nina Pro datasets (i.e., we convert the .mat files to .csv files). Conveniently, when creating datasets with LibEMG, data are saved in a consistent format by default, standardizing the overall data storage process.

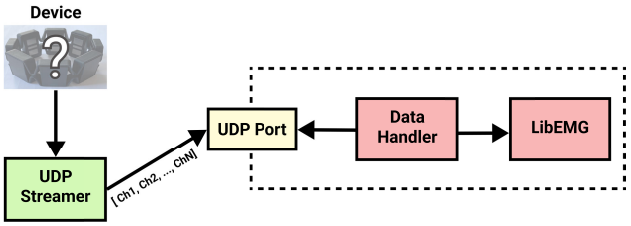


FIGURE 2. Hardware agnostic architecture implemented to abstract the collection of real-time EMG data.

The **OnlineDataHandler** is responsible for collecting, processing, and storing all EMG data streamed from a device in real-time. As depicted in Figure 2, the architecture consists of two main components, a UDP streamer, and a UDP listener. Since these two components are mutually independent and run in different processes (i.e., threads), this architecture is hardware agnostic and acts as a middle layer between the library and the hardware device. Therefore, for any device not integrated into the library already (e.g., a custom hardware setup), a simple UDP streamer (see Section III-C1) can be created to connect a device and leverage the toolkit's functionality. The job of the UDP streamer is to continuously listen to an EMG device for new data. Often this data is streamed over BLE-enabled devices such as the Myo Armband. For every new sample of data (a single reading across a set of channels/electrodes), the UDP streamer writes it to a pre-specified UDP port, and the UDP listener continuously listens on this port and updates an internal buffer with the new data. Ultimately, these data are used for several purposes, such as creating datasets, making predictions, or depicting real-time visualizations. Furthermore, to reduce the burden on developers using this library, we have implemented the UDP streamers for some common hardware (see Section III-C1).

2) FILTERING

The Filtering module of LibEMG is crucial for designing systems with consistent performance across various situations (e.g., during electrode shift or in the presence of powerline interference). As the EMG signal is stochastic, we want to minimize external sources of noise common in real-world use that can overshadow the gesture-specific information required to decipher user intent. For example, EMG measurements are often inundated by powerline interference (50/60 Hz noise) due to proximity to wired electronic devices and the human body acting as an antenna. Further, during movement of the limb and, correspondingly, measurement equipment, the contact characteristics between the electrodes and the skin change, resulting in low frequency (<20 Hz) motion artifacts to the signal. The Filtering module was designed to remove this frequency-localized noise from the signal and improve the downstream classification.

Filtering can be applied as a stand-alone component on data external to the pipeline, can be used directly on datasets through the OfflineDataHandler, or can be passed to an OnlineDataHandler to perform filtering on a live stream of

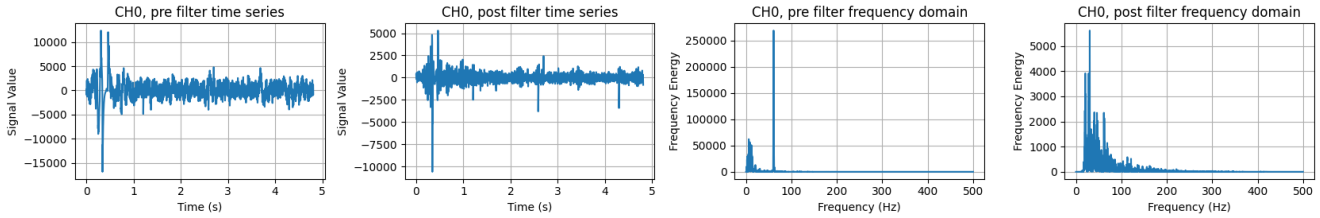


FIGURE 3. A built-in visualization of the effect of a fourth-order notch filter for powerline interference removal on the time series and frequency spectrum of a single EMG channel.

data. The filter component requires first installing the desired filters and provides an easy way of designing the filters through passing in dictionaries with interpretable keys like *name* (e.g., notch, lowpass, highpass, bandpass, or bandstop), *cutoff* (i.e., the boundary of where the filter operates), and *order* (i.e., the rate at which energy is removed from the frequency band). Although LibEMG supports the exploration of a variety of filters, it also includes a method to install a set of filters recommended in prosthetics research to combat common noise sources, such as the aforementioned powerline interference and motion artifact. Figure 3 shows the effects of the filtering module on removing 60 Hz powerline interference in the time domain (i.e., the evolution of the EMG signal over time) and frequency domain (i.e., the energy of the signal across frequencies).

Another form of signal conditioning supplied through the filter component is channel-wise standardization. For standardization, the mean and standard deviation of each channel is computed from an OfflineDataHandler and stored for later transformations. The standardization can be helpful for deep learning applications where the ideal range of input values is near -1 to 1; however, establishing a uniform range for inputs will also make the default parameters of feature extraction arguments more viable (see Section IV-C).

3) FEATURE EXTRACTION

After collecting, filtering, and segmenting the data, features can be extracted from each window using the Feature Extraction module to increase the information density of the underlying EMG signal. Numerous studies have analyzed features, and their combination (i.e., feature sets), for optimal performance [44], [45], [46], [47]. Understanding the optimal features/groups to use can often be challenging and even daunting. Moreover, after deciding on the feature set, implementation poses a significant programming burden. To alleviate these challenges, we provide developers with a list of 50 features (see Table 1) and 11 feature sets (i.e., combinations of individual features that have been identified as beneficial combinations in previous work; see Table 2). All features included are from the frequency domain, time domain, or time-frequency domain and have been suggested for myoelectric control in prior work. Additionally, each feature's implementation has been thoroughly validated and tested. Since this module stands independently from the rest of the pipeline, it can be leveraged to extract features from any time series, EMG or non-EMG related. Finally, as part of this module, LibEMG includes a function for visualizing a variety of projections (e.g., PCA) of the feature space (see Section IV-A). This tool is useful for reducing the dimensionality of a dataset down to two dimensions to facilitate visualization.

4) FEATURE SELECTION

The Feature Selection module provides an optional assessment after data collection to provide developers with a means to search for an information-dense subset of features (not unlike the aforementioned feature groups) from the features provided in the library. Six metrics for feature group optimization, based on sequential forward selection, are incorporated by default within the library: accuracy, active error, repeatability index, separability index, mean semi-principal axis length, and feature efficiency [68]. The output of this component provides a feature ranking and the associated metric values, which can be used to decide the set of features for classification. Essentially, this module can select a feature group that optimizes one of the aforementioned metrics for a given problem. Although optional, this module is ideal for more complex analyses that require custom-tuned feature sets or feature comparisons.

5) CLASSIFICATION

To enable the classification of a diverse set of muscle-based inputs, researchers have turned to machine learning techniques. Using the features extracted from the Feature Extraction module, algorithms can discriminate between muscle patterns. To facilitate this process, LibEMG's Classification module wraps the popular Python library for machine learning, scikit-learn [40]. This provides access to robust algorithmic implementations for expert users, as they can pass in all the same parameters accessible through scikit-learn, while simultaneously alleviating the challenges for those with less experience by incorporating default parameters. An example of these parameters includes the optimal k-value for a KNN classifier. We also acknowledge that many researchers will have custom algorithm implementations, including deep learning models, that they wish to use within LibEMG's pipeline. To accommodate this, we designed the EMGClassifier module to accept any classifier as long as it follows

TABLE 1. List of features available to be extracted and used as input to the classification module.

[48] Mean Absolute Value (MAV)	[49] Median Frequency (MDF)
[48] Zero Crossings (ZC)	[49] Mean Frequency (MNF)
[44] Slope Sign Changes (SSC)	[50] Mean Power (MNP)
[44] Waveform Length (WL)	[44] Maximum Peaks (MPK)
[46] L-Score (LS)	[51] Discrete Time Fourier Transform Representation (DFTR)
[52] Maximum Fractal Length (MFL)	[53] Skewness (Skew)
[46] Mean Squared Ratio (MSR)	[53] Kurtosis (KURT)
[48] Willison Amplitude (WAMP)	[54] Root Mean Squared Phasor (RMSPHASOR)
[3] Root Mean Square (RMS)	[54] Waveform Length Phasor (WLPHASOR)
[48] Integral Absolute Value (IAV)	[44] Peak Average Power (PAP)
[55] Difference Absolute Standard Deviation Value (DASDV)	[56] Multiplication of Peaks (MZP)
[48] Variance (VAR)	[44] Spectral Moment (SM)
[44] Temporal Moment (TM)	[57] Integral Square Descriptor (ISD)
[58] First Temporal Moment (M0)	[57] Coefficient of Regularization (COR)
[58] Second Temporal Moment (M2)	[48] Log Detector (LD)
[58] Fourth Temporal Moment (M4)	[45] Mean Absolute Value First Difference (MAVFD)
[59] Activation (ACT)	[44] Mean Absolute Value Slope (MAVSLP)
[59] Complexity (COMP)	[60] Fuzzy Entropy (FUZZYEN)
[58] Sparsness (SPARSI)	[61] Sample Entropy (SAMPEN)
[58] Irregularity Factor (IRF)	[57] Mean Difference Derivative (MDIFF)
[58] Waveform Length Factor (WLF)	[57] Mean Logarithm Kernel (MLK).
[62] Autoregressive Coefficient 4 (AR4)	[59] Mobility (MOB)
[62] Autoregressive Coefficient 9 (AR9)	[63] Cepstral Coefficient (CC)
[64] Wavelet Energy (WENG)	[64] Wavelet Variance (WV)
[64] Wavelet Waveform Length (WWL)	[64] Wavelet Entropy (WENT)

TABLE 2. Feature Sets: Predefined groups of features that have been identified in previous work as being useful for classification purposes.

Feature Set	Features
[65] Hudgins' Time Domain (HTD)	MAV, ZC, SSC, WL
[47] Low Sampling Frequency 4 (LS4)	LS, MFL, MSR, WAMP
[47] Low Sampling Frequency 9 (LS9)	LS4, ZC, RMS, IAV, DASDV, VAR
[58] Time Domain Power Spectral Descriptors (TDPSD)	M0, M2, M4, SPARSI, IRF, WLF
[66] Time Domain Autoregressive (TDAR)	MAV, ZC, SSC, WL, AR4
[45] Topologically Selected Time Domain (TSTD)	MAVFD, DASDV, WAMP, ZC, MFL, SAMPEN, TDPSD
[57] Inverse Time Domain Features (ITD)	ISD, COR, MDIFF, MLK
[59] Hjorth Parameters (HJORTH)	ACT, MOB, COMP
[67] Combined (COMB)	WL, SSC, LD, AR9
[51] Discrete Fourier Transform Representation (DFTR)	DFTR
[64] Multi-Signal Wavelet Transform-Based Feature Set (MSWT)	WENG, WV, WWL, WENT

the scikit-learn standard and has a **.fit**(), **.predict**(), and **.predict_proba()** function. Therefore, with minimal effort, custom classifiers can be passed to the pipeline to leverage its functionality.

This module can be split into the **EMGClassifier** and the **OnlineEMGClassifier**. The main difference is that the OnlineEMGClassifier leverages the OnlineDataHandler to read live data and output real-time predictions over a pre-specified UDP or TCP port. Additionally, it handles the buffering of windows, feature extraction, and prediction streaming. This socket-based design enables any application, including mixed reality devices such as the HoloLens, to leverage the output from the library through a UDP or TCP protocol. In contrast, while the EMGClassifier is for running offline analyses and is partially a wrapper for scikit-learn, its true advantage is that it has post-processing built in. The goal of post-processing is to improve the classifier's decision after a prediction by leveraging additional contexts such as confidence (i.e., rejection) or previous predictions (i.e., majority voting). Currently, LibEMG supports rejection (i.e., rejecting decisions with low confidence) [69], majority voting (i.e., applying a majority vote to the past N decisions) [70], and proportional control (i.e., outputs a speed between 0-1 corresponding to the contraction intensity) [15].

6) EVALUATION

Evaluating myoelectric control systems is challenging, and although there is a tendency to favour offline evaluation (i.e., recording contractions and evaluating classification accuracy using previously collected data), recent work suggests that these offline analyses are not necessarily representative of online usability [68], [71], [72]. For myoelectric control system evaluation, this is especially true outside of lab-based settings where several factors, including varying limb position [73], fatigue [74], and electrode shift [67], ultimately lead

FIGURE 4. The supported hardware (at time of submission). From left to right, the (1) Myo Armband, (2) SIFI Labs Bioarmband, (3) Delsys Trigno and Avanti, and (4) gForcePro+ EMG Armband.

to varied EMG signals and system degradation. Regardless, offline analysis is commonplace in prosthetics research due to ease and accessibility, the expense of prosthesis fittings, challenging hardware setups, and difficulties recruiting amputee participants. This has influenced the evaluation of general-purpose applications, since running online studies adds additional complexities to studies. As a result, there is currently an over-reliance on offline metrics, such as classification accuracy, for evaluating these systems. With this in mind, the **Evaluation** module provides developers with a common set of offline evaluation metrics for running offline analyses. While these metrics may not fully predict the real-world viability of a control system, they are valuable for control system tuning (i.e., testing different parameters such as the window size/increment, classifier, or rejection) before deploying the system. This enables developers to optimize their control systems using offline data before running online usability studies.

Although this module focuses on the offline evaluation of control systems, researchers should use it with its limitations in mind. Moving forward, there needs to be more of a focus on evaluating the online usability (i.e., how a system performs when the user gets to interact with it) of a designed control system. As this technology continues to progress toward its use for general-purpose applications, system evaluations will inevitably become application specific. For example, a control system designed to navigate a menu should be evaluated for that particular use case. Generally, the best method to evaluate control schemes is to deploy the system and evaluate its use with user-in-the-loop feedback (i.e., where the user gets feedback and makes decisions based on the control system's output). LibEMG provides all of the functionality to do this through the OnlineEMGClassifier. Additionally, we provide a full testing environment in a version of the ISO 9241-9 (pointing device assessment – Fitts' Law test) [75] (see Section V-D), which has been adopted in prosthetics research for evaluating the online performance of myoelectric control systems (by mapping particular muscle contractions to the movement of an onscreen cursor).

C. SUB MODULES

To provide developers with access to convenient functionality that is not part of the core pipeline, we present a group of sub-modules: streamers, datasets, and data collection. We explore the functionality of each of these sub-modules below.

1) STREAMERS

With the current state of myoelectric control for general-purpose use and the commercial discontinuation of the Myo Armband, a range of different sensors are used across research labs. Some have custom hardware setups, others use expensive biomedical-grade equipment, a few have access to un-released commercial devices, and many continue to use the discontinued Myo Armband. Developing an interface for all these different technologies would be unfeasible, so instead, we designed LibEMG to be hardware agnostic. However, to facilitate the development process and improve usability, LibEMG includes a streamers sub-module (see Figure 4), to interface with commonly used hardware including the Myo armband, SIFI Labs Bioarmband,[5] the gForcePro+ EMG Armband,[6] and the Delsys[7] Trigno and Avanti. Therefore, anyone interested in using these devices can interface with them directly through the library's API with a single function call. These streamers are simply predefined code for streaming data over UDP, adhering to the architecture defined in Section III-B1. As new EMG devices are released to the public and become widely adopted, the library will be updated to support them.

2) DATASETS

While EMG dataset repositories currently exist, such as Kaggle,[8] Physionet,[9] and IEEE Dataport,[10] no previous work has integrated them directly into a library that automatically downloads, extracts, and loads them into a pipeline for eventual classification. Ultimately, this alleviates many of the challenges associated with interfacing datasets, including the significant up-front time investment. LibEMG provides four datasets by default, with plans to incorporate more in future releases. This dataset integration was a crucial design consideration, as the overhead required for parsing and downloading

[5]https://sifilabs.com/bioarmband/
[6]http://www.oymotion.com/en/product32/149
[7]https://delsys.com/
[8]https://www.kaggle.com/
[9]https://physionet.org/
[10]https://ieee-dataport.org/

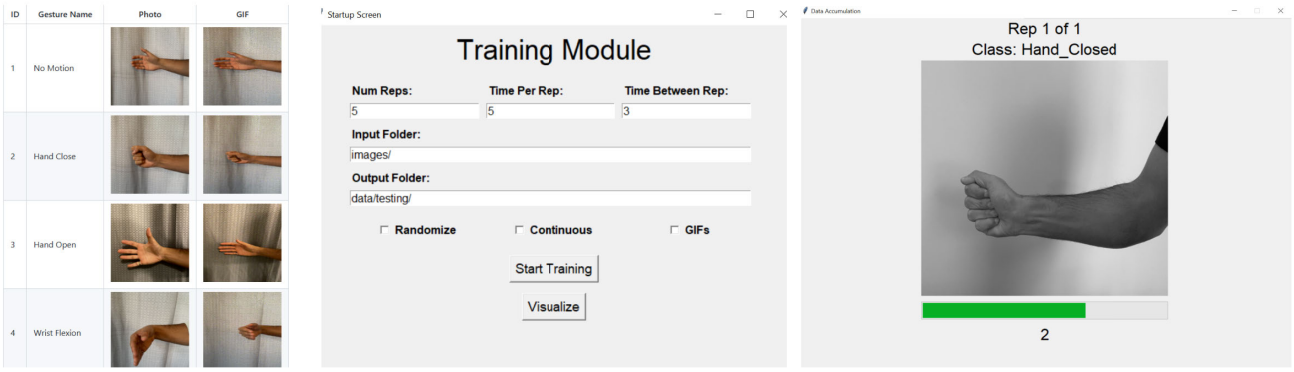


FIGURE 5. The three main components of the data collection module. From left to right: (1) the gesture library consisting of 35 distinct HCI gestures, (2) the main menu of the data collection screen, (3) the data collection screen.

data from different locations is a significant burden. Including a shared repository of datasets for all researchers to use serves two purposes: (1) it makes accessible the exploration of this technology for researchers that do not have the time or means required to acquire or interface datasets, and (2) it serves as a common set of data (or baseline) for researchers to compare their results, improving transparency and reproducibility in this field. The datasets currently supported in LibEMG are summarized in the API documentation. To comply with data usage permissions, the Nina Pro datasets must be downloaded manually from their website.

3) DATA COLLECTION

Data Collection is used to acquire data (i.e., datasets) for future offline analysis or to train a model for real-time interactions. Usually, this occurs through screen-guided training, where users are prompted with a particular image representing a contraction that they elicit for a predefined amount of time. Typically these prompts last between two to five seconds and are repeated for several repetitions. In general, more repetitions lead to more variability and, in turn, better models, but they come at the cost of tedious training. LibEMG enables this data collection through a simple user interface pictured in Figure 5. In this data collection UI, developers can customize the number of reps, time per rep, and time between contractions. Once these values are selected, this module will record the live data from the OnlineDataHandler and save them in csv files for future processing.

Additionally, with the release of LibEMG, we provide a standardized gesture repository,[11] consisting of 35 gestures, including both static (i.e., photos) and dynamic (i.e., videos) gesture representations. For example, the wrist-flexion class is represented in its static form as wrist flexion through an image and its dynamic form (i.e., a wrist flick) through a video. This ensures consistency among studies reliant on data collection and reduces the requirement for manually acquiring these images or videos. These gestures can be downloaded directly from the GitHub repository or through the API by specifying a set of indices (where each gesture is associated with a unique index). Additionally, LibEMG supports the use of custom images for added flexibility and convenience. In turn, this provides developers with all the tools required for simple screen-guided training to acquire EMG data.

[11] https://github.com/libemg/LibEMGGestures

IV. OFFLINE DEMONSTRATIONS

While LibEMG strives to facilitate the development of online interactive systems, offline analysis is nevertheless crucial. Inevitably, the creation of robust interactive systems starts at the algorithmic level before funneling down to the online control system. This process of testing different algorithms, features, and filtering strategies on offline data before deployment is referred to as control system tuning. In this section, we present four offline demonstrations: (1) a simple introduction to offline analysis, (2) a more thorough review of LibEMG's features and feature sets on a Nina Pro database, (3) a method for extracting optimal feature parameters, and (4) a deep learning example. These demonstrations serve as a starting point to highlight some of LibEMG's functionality for running robust offline analyses and system optimization. A more thorough walkthrough of each example, including step-by-step design and code, can be found in the API documentation:

(A) LibEMG_OneSubject_Showcase
(B) LibEMG_Feature_Showcase
(C) LibEMG_FeatureOptimization_Showcase
(D) LibEMG_DeepLearning_Showcase

A. SIMPLE OFFLINE ANALYSIS

One goal of LibEMG is to enable the initial exploration of myoelectric control systems with minimal setup time. In this example, we demonstrate the core functionality of LibEMG through a straightforward offline analysis of a single participant by leveraging the OneSubjectMyoDataset. We compare the classification accuracy, active error, and instability of five

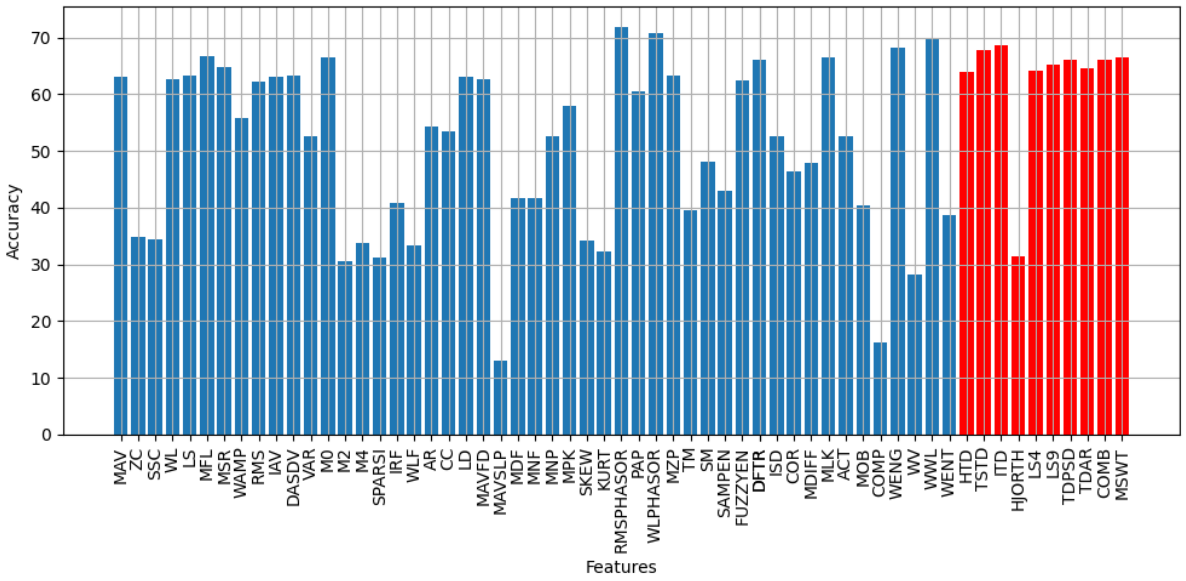


FIGURE 6. Offline accuracy of the individual features and feature sets included in LibEMG computed for a single subject in the NinaPro DB8 dataset. Blue bars represent individual features and red bars represent feature sets.

classifiers (LDA, SVM, KNN, RF, and QDA) on a five-class problem: hand open/close, wrist flexion/extension, and rest. Six reps (i.e., three trials) are used for training and six reps for testing. The classification accuracies achieved using the HTD feature set were 98.5%, 97.6%, 97.7%, 98.0%, and 98.9% respectively for the LDA, SVM, KNN, RF, and QDA classifiers. Additionally, we demonstrate some of the visualization functionality of the library by plotting the PCA feature space, exemplified in Figure 7. This is a common approach used in machine learning research to visualize a dataset's feature space by reducing its dimensionality to 2D (i.e., two principal components).

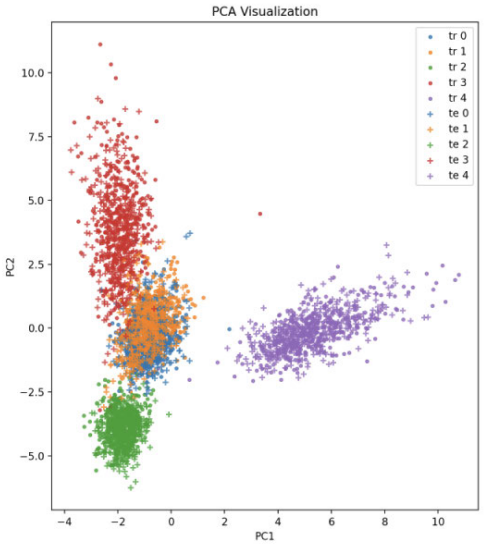


FIGURE 7. The PCA feature space of the OneSubjectMyoDataset representing the training (.) and testing data (+).

B. FEATURE AND FEATURE SET EVALUATION

A major benefit of LibEMG is the robust library of validated features and feature sets. When designing a control system, choosing the best subset of features for a given problem is challenging because of the large number and combination of options. Only a small subset of features should be used at run-time to minimize computational complexity and redundancy. While this subset can be determined automatically using the feature selection module (section III-B4), designers can also manually assess which of the individual features and feature sets are beneficial as a first step during the control system tuning stage. For an example of such a process, a popular finger movement dataset (NinaPro DB8) was used to demonstrate a simple offline analysis of the accuracy of individual features and feature sets. Although classifying the different motions within this dataset is challenging due to the many classes required to represent both flexion and extension of the digits, Figure 6 shows the accuracies obtained using rather naive models based on single features and the included feature sets. Although this initial offline performance would not necessarily be viable for a real-world control system, it could be improved by filtering, constructing a more robust feature set, optimizing the parameters of the feature extraction module, or implementing more advanced classification models. Note that this example can be easily extended to different datasets, as highlighted by the analysis of Nina Pro DB2 in the same repository.

C. DETERMINING OPTIMAL FEATURE PARAMETERS

Although myoelectric potentials typically range from the micro to milli-volt range, different devices amplify raw potentials to different ranges. Inevitably, this impacts the

feature extraction module as it relies on predefined parameters (such as the threshold for slope sign changes). Since the feature extraction module was optimized for the Delsys Trigno and Myo Armband systems, some features reliant on these thresholds may not be optimal on different hardware. We recommend tuning these default arguments used by the feature extraction methods to get the best performance possible. In this example, we leverage the 3DCDataset to demonstrate the feature parameter tuning process. The Willison's amplitude (WAMP) feature is commonly used in myoelectric control and corresponds to the total number of times the magnitude of the derivative exceeds some specified threshold within a window. Selecting an appropriate threshold for this feature can drastically influence a system's performance. So, in this example, we provide the foundational process for optimizing feature parameters, a useful strategy in general for feature parameter selection. To do this, we iteratively test a range of thresholds, and through a visual inspection, we can make an informed selection of the ideal value. From Figure 8, the optimal WAMP threshold of 92 resulted in an accuracy of 89% on the validation set and 84% on the test set which was drastically higher than other valid (non-error returning) potential threshold values.

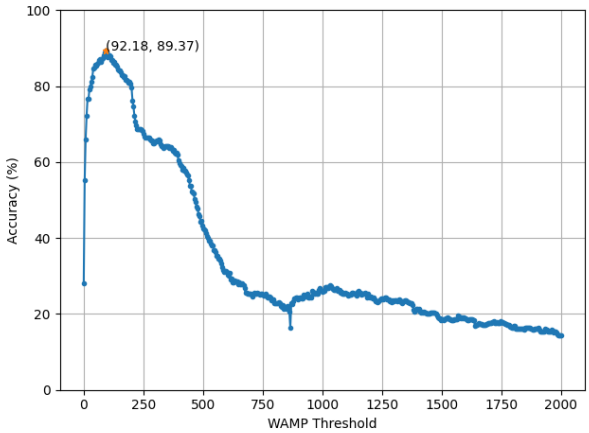


FIGURE 8. The accuracy of different WAMP threshold values.

D. CUSTOM CLASSIFIERS: DEEP LEARNING

In the past offline examples, the library relied on classifiers from the scikit-learn package to perform gesture recognition; however, a custom classifier can also be integrated into the pipeline if it has a fit, predict, and predict_proba method. As an example of integrating a custom classifier, a PyTorch convolutional neural network (CNN) that operates directly on the raw windowed signals, as opposed to hand-crafted features, was used within the EMGClassifier object. The first four trials of the OneSubjectMyoDataset were used to train the model weights, and the fifth trial was used as a validation set. The performance on the held-out sixth trial was 95% accuracy, 5% active error, and 2% instability. While this was a custom classifier, it could still use the post-processing techniques provided by the EMGClassifier object, such as obtaining a proportional control output or improving performance metrics using rejection and majority vote. By adding a rejection threshold of 90% confidence and a majority vote window of three samples, the performance was improved to 97% accuracy, 3% active error, and 0% instability while rejecting 14% of the inputs.

This simple deep learning example can be extended to more challenging problems, like cross-user gesture recognition which tries to maintain the accuracy achieved by within-subject models while not requiring the end-user to provide their own data. Prominent strategies like adaptive domain adversarial neural networks [76], [77], subject-specific normalization [78], and few-shot supervised domain adaptation [79] have shown early success in this area. Moreover, with the flexibility to update model weights online, related research areas like adaptive supervised strategies [80] or reinforcement learning [81] can be explored.

V. ONLINE DEMONSTRATIONS

In this section, we present five online (i.e., real-world) demonstrations of LibEMG: (1) a simple continuous control scheme to play Snake, (2) cross-platform use for a myoelectric training game in Unity, (3) proportional control to interface a mouse and play an online game, (4) an ISO Fitts' evaluation tool for assessing the online performance of different classifiers, and (5) controlling a menu in mixed reality. The goal of these demonstrations is to highlight LibEMG's functionality and provide a starting point for future work. An in-depth walk-through of each example, including code and design decisions, can be found in the API documentation:

(A) LibEMG_Snake_Showcase
(B) LibEMG_Unity_Showcase
(C) LibEMG_Cursor_Showcase
(D) LibEMG_Isofitts_Showcase
(E) LibEMG_MixedReality_Showcase

A. SIMPLE CONTINUOUS CONTROL: SNAKE

Continuous control [10], where decisions are generated based on a predefined parameter such as a time increment, is the current commercial standard approach to prosthesis control. These decisions are governed by the window size (i.e., the number of samples) and the increment size (i.e., the number of samples advanced before extracting a new window). For prosthesis control, a classifier makes a prediction for each window of data, typically on the order of 150-200 milliseconds, to provide amputees the responsiveness required to micro-adjust their prostheses. In this example, we leverage this type of control for a common HCI application – gaming. To do this, we created an adapted version of the traditional snake game using Pygame, a game development library for Python. The player can move the snake up, down, left, and right using hand open/close and wrist flexion/extension muscular inputs. The goal of the game is to navigate the

environment without the snake bumping into the wall or itself, growing the snake as large as possible by eating food.

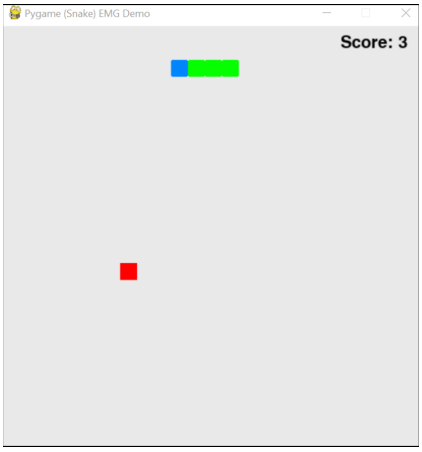


FIGURE 9. Leveraging continuous control to play a simple Pygame rendition of Snake.

B. CROSS PLATFORM USE: UNITY

It may be desirable to use technology platforms other than Python to develop applications that leverage myoelectric control. For example, applications might be developed in another programming language, such as C#, or on a completely different device, such as a mixed-reality headset or a microcontroller. In this example, we interface LibEMG with Unity, a 3D game development environment that is widely used for VR/AR applications. To accomplish this, the OnlineEMGClassifier communicates over UDP or TCP, meaning any environment or programming language that can open a standard socket interface can communicate with LibEMG. This example shows how to leverage the library in a simple Unity game (The Falling of Momo [9] – borrowed from the work done by Tabor et al.) to create a myoelectric training game for amputees. In this fall-down style game, players can move the character left or right using wrist flexion/extension and jump over barriers by making a fist.

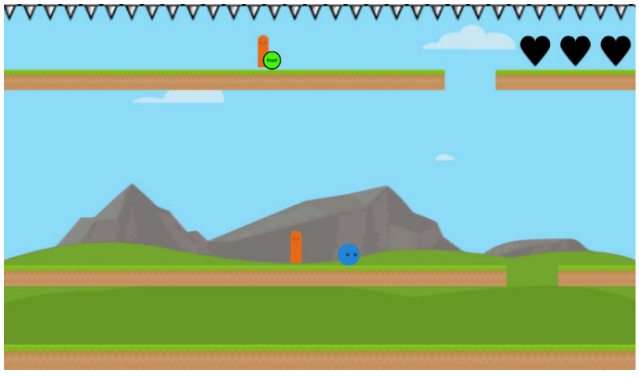


FIGURE 10. Interfacing a myoelectric training game (The Falling of Momo) in Unity.

C. PROPORTIONAL CONTROL: MOUSE INTERFACE

Combined with continuous control, proportional velocity-based control systems control the speed of an action based on the contraction intensity. For example, an amputee using this type of system can open their prosthetic hand very slowly by performing a soft hand-open contraction. When it comes to continuous movements, such as prosthesis control, or in this example, cursor control, proportional control can drastically improve throughput through speed modulation [15]. In this example, we use a proportional control system to create a cursor interface for a desktop computer. Ultimately, users can control the direction of the cursor through a set of contractions (flexion/extension - left/right and hand open/close - up/down) and its speed through contraction intensity (a number between 0-1). Leveraging Pyautogui to create mouse inputs, we are able to replace cursor movement with muscle contractions and demonstrate its use by playing snake.io, a third-party[12] online game that is not unlike our snake game. This example could be extended to any cursor-based application and with minor modifications, could recognize discrete inputs (e.g., finger taps) for left and right mouse clicks.

FIGURE 11. Using proportional control to control a mouse and play an online game.

D. ISO 9241-9 FITTS' LAW EVALUATION ENVIRONMENT

Highlighted by previous work [68], [71], [72], offline evaluation metrics, such as classification accuracy, are not necessarily representative of the online usability of a control system for various reasons such as fatigue [82], electrode shift [83], varying contraction intensities [51], and closed-loop user behaviours. As a result, online evaluation must become a core part of interaction design studies to properly assess the viability of a myoelectric control system, especially in HCI. In this example, we introduce an ISO 9241-9 inspired Fitts' test environment to evaluate the online performance of a myoelectric control system with two degrees of freedom (up/down and left/right). Fitts' law evaluations are common for evaluating myoelectric control systems in prosthetics research, as it provides an indication of real-world system viability without the requirement of expensive prosthetics hardware. In this example, we designed a mini experiment to compare

[12]https://snake.io/

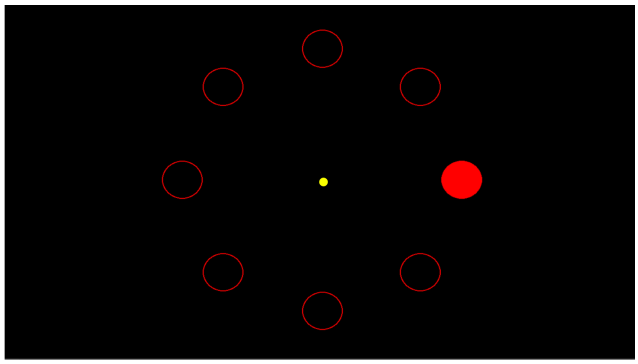


FIGURE 12. An online ISO 9241-9 inspired Fitts' Law assessment tool for myoelectric control.

FIGURE 13. Mixed reality input on the HoloLens 2.

the online performance of four classifiers (LDA, SVM, RF, and KNN) using the metrics of Throughput, Efficiency, and Overshoots proposed in [15]. Although not a formal evaluation, this demonstration is a potential starting point for usability studies, in both HCI and prosthetics to build on moving forward.

E. MIXED REALITY: MENU NAVIGATION

One of the main reasons for the renewed interest in EMG as a general-purpose input modality stems from its applicability to ubiquitous computing and mixed reality. With the continued advancement of mixed reality technology, myoelectric control may become desirable for several reasons. First, EMG can be built directly into wearable devices such as watches and bracelets making it a form of always-available input. Second, myoelectric control can be used in situations where computer vision might not be available, such as in poor lighting conditions or when the hands are outside of the view of cameras (e.g., in gloves). Finally, EMG can enable more subtle gestures compared to movement-based input modalities such as inertial measurement units or camera-based approaches. This could make its use ideal in situations where dynamic gestures are less socially acceptable (e.g., quickly dismissing a phone call in public). Regardless of its promise, however, there has been little research on its use in mixed-reality environments outside of private industry-related work. This example demonstrates how LibEMG can be leveraged to navigate a menu on the HoloLens 2. To communicate with the headset, we connect over TCP and enable users to open/close the menu (hand open), scroll up and down (flexion/extension), and make selections (finger tap). Although simple, future work could extend this example to enable readily-available hands-free input for any mixed reality application (e.g., controlling a music application or responding to messages).

VI. DISCUSSION

In this work, we have presented LibEMG, a library for facilitating the development of robust myoelectric control systems for "general-purpose applications" (i.e., the use of EMG as input to interactive systems outside of prosthesis control). We hope that with the release of this library, we can move toward more collaborative and reproducible research, inevitably improving the robustness and reliability of EMG technology as is required for its widespread adoption. With the growing popularity of ubiquitous computing and the inherent need for readily available inputs, particularly within mixed-reality environments, an exciting opportunity has arisen for this technology to be widely applied to many new applications. In turn, we believe that this work and library has come at an opportune time to enable researchers, particularly outside the traditional field of prosthetics, to apply their expertise to help mature this technology to a level that is competitive with or complementary to other input modalities such as computer vision or IMU-based input. Regardless, there is still a long way to go, both in this field and with this library, and so in this section, we critique our own work to better position its contributions.

A. EVALUATION BY DEMONSTRATION AND CONTINUED LOCAL USE

Library (or toolkit) evaluation, as captured by Ledo et al. in [28], is crucial for all toolkit-related research. Correspondingly, one of LibEMG's main contributions is its evaluation by demonstration, where we showcase its use through four offline and five online examples. The contribution of these demonstrations is not the examples themselves, but that they act as mini-projects for future work to build upon. Additionally, the fidelity of the library was evaluated and validated by analyzing its performance compared to previously published work and code wherever possible. For example, the benchmarks set in Section IV-B for Nina Pro DB8 and DB2 were validated against those achieved on the same dataset in [84].

LibEMG is the culmination of our experiences in conducting EMG research in the prosthetics field over many years. While LibEMG may be new, we have been using its constituent tools (e.g., scikit-learn) and the included datasets daily as core parts of our daily research for some time. Now that we have completed the first full version of LibEMG, our lab (typically over a dozen researchers at any one time) and other collaborating labs have begun to adopt the library. Correspondingly, we expect to receive continued feedback

and feature requests, leading to the continued maturation of this library. Most importantly, however, we believe that the true evaluation of LibEMG will occur after its public release through continued feedback and constant iteration from an even larger community.

B. A LIBRARY FOR DEVELOPING EMG-BASED INTERACTIONS

From the very name of LibEMG, we have intentionally positioned this work as a library (which we define as being comprehensive, well-documented, and well-maintained), as opposed to a toolkit (which does not have such requirements), with the hope that it will become adopted by the greater research community, similar to LIBSVM [41] and scikit-learn [40]. With this proposition, LibEMG offers a set of resources outside the scope of toolkits and enables the development of interactions beyond simple prototypes. Firstly, it is a more general set of tools that can be applied to develop a wide range of interactive systems. Secondly, it includes a publicly documented API with a detailed user guide that supports learning key EMG concepts and both offline and online demonstrations. Moving forward, this living documentation, which highlights functionality beyond what was explored in this paper, will continue to serve as a critical resource for developers interested in developing myoelectric control systems. Thirdly, it will be available for download via PyPI, making its integration into Python environments trivial. Finally, since LibEMG is a tool that has been adopted as a lab initiative to promote reproducibility and transparency, we are committed to its maintenance moving forward. Ultimately, we hope that this library can enable researchers to begin focusing on the interactive aspects of myoelectric control without having to worry about the design and implementation of robust control systems. Ideally, this will unlock a realm of potential future myoelectric control research across various research fields.

C. ENABLING CROSS-FERTILIZATION BETWEEN HCI AND PROSTHETICS

With the commercialization of machine learning-enabled EMG prosthesis controllers (e.g., Coapt,[13] Infinite Biomedical,[14] and Ottobock[15]), the growing popularity of mixed reality technologies, and a need for readily available hands-free input, myoelectric control may be more popular and desirable than ever before. However, its use for general-purpose applications is still far away, partly because, until now, the technology and techniques needed for building robust EMG control have been largely inaccessible to researchers outside prosthetics. LibEMG changes this by providing the tools required to begin asking and answering the questions needed for maturing this technology. For example, what gesture sets will work best in mixed-reality environments? Moreover, how can we design user interfaces to better support users employing myoelectric control? We also hope this library opens up research avenues for non-proshthetic researchers (e.g., HCI researchers) to contribute their expertise to advancing and maturing prosthetics research. For example, how can embodiment be improved between amputees and their prostheses? Or, how can we create more robust and engaging training tools for amputees? Ultimately, we hope that LibEMG can open up previously inaccessible research opportunities to enable the continued progression of this technology outside of and within prosthetics.

[13] https://coaptengineering.com
[14] https://www.i-biomed.com/
[15] https://www.ottobock.com/

D. CREATING HCI-SPECIFIC EMG CONTROL

LibEMG's pipeline (i.e., Data Handling, Filtering, Feature Extraction, Feature Selection, Classification, and Evaluation) was directly inspired by the work in [7] for designing myoelectric control systems for general-purpose HCI systems (i.e., rather than for prosthesis control). Nevertheless, continued improvements to this library moving forward will be crucial for its success. This is especially true as myoelectric control in HCI continues to mature and consequently diverges from prosthesis control. For example, LibEMG currently only supports continuous control schemes (i.e., where classifier events are generated based on a predefined periodic parameter, such as a window increment). While discrete inputs, such as a wrist flick, can be recognized within continuous control schemes through post-processing state logic, it is not necessarily optimal [7]. For example, to recognize a wrist flick gesture, the decision stream would likely output a sequence of ''rest'' (or inactive) decisions, followed by some number of wrist flexion decisions, followed again by more rest decisions. This decision stream would then have to be mapped to the discrete event through some sequence of filtering or state logic. Considering that HCI is full of these event-based actions, such as flicks, swipes, gestures, and button clicks, there is a need to incorporate fundamentally different temporally aware classifiers that make sense in the context of general-purpose use cases. Future work will explore the implementation, evaluation, and integration of more robust control schemes for recognizing discrete inputs.

Another area of improvement is that at the time of submission of this work, only four compatible devices and four datasets are built into LibEMG by default. We plan to extend this to a much larger repository of datasets (similar to Nina Pro) with support for numerous EMG devices. However, for the continued growth and improvement of this library, we realize that community-wide adoption and feedback are required. Additionally, enabling cross-user models may be crucial to the future success of EMG technology in consumer applications. Although LibEMG does not currently support this functionality, it will be included in future releases.

E. OTHER USES OF LibEMG BEYOND MYOELECTRIC CONTROL

Although LibEMG was designed using principles established in EMG-based upper-limb prosthesis research, the

customization available within the modules (e.g., filtering, feature extraction, and classification), and the interfaces for connecting to hardware and broadcasting predictions means LibEMG could provide benefits to other uses of EMG and research outside of myoelectric control.

The use of EMG in HCI is not limited to the control of interactive systems through muscle-based input on the forearm and wrist. For example, EMG has been studied in the context of brain-computer interfaces [85]. Further, it has been used for user identification as a biometric in the context of security [86]. Other recent examples have shown compelling examples of EMG used on the legs to control an avatar in VR [87]. In these examples, much, if not all, of what we have presented is applicable. Accordingly, LibEMG can facilitate the principled exploration of exciting new uses of EMG for system input beyond what we demonstrated in this work.

Additionally, many modern systems leverage human intent or state-of-being decoded from different modalities (e.g., inertial measurement units (IMUs), surface electroencephalography (EEG), electrocardiography (ECG), and galvanic skin response (GSR)) using similar modules to those supplied by LibEMG. IMUs are currently being leveraged in similar pipelines, such in the Apple Watch's assistive touch gestures, and is an ongoing area of research [88], [89]. Additionally, a large number of resources provided by the library are used in brain-computer interfaces [90]. For example, the EEG signal often undergoes the same windowing, feature extraction, and classification process as EMG [91]. Likewise, GSR or ECG use similar pipelines for detecting stress [92] or pain [93] which can be valuable triggers to govern an affect-aware human-computer interaction. We encourage the community to explore these broader applications using LibEMG and improve the library by enriching and extending its use with best practices from other fields.

F. CONTRIBUTIONS

Contributions to improve and expand LibEMG can come in a variety of ways. Firstly, as previously highlighted, we are open to exploring the integration of additional datasets to build a repository of validated datasets. For example, an ideal dataset for a future release could have dynamic gestures such as swipes, flicks, and gestures. Secondly, we are increasingly interested in exploring new technology and are interested in supporting more hardware, especially wearable devices. Finally, we hope to receive continuous feedback from the community to make LibEMG a tool adopted for all EMG research (including for prosthetics). In turn, we hope that LibEMG inspires a more collaborative, transparent, and reproducible research community for all EMG-related work moving forward.

VII. CONCLUSION

This paper presents LibEMG, a library to facilitate the development and evaluation of EMG-based interactions. With the increased interest in novel hands-free input modalities for ubiquitous environments such as mixed reality, myoelectric control has become increasingly desirable. However, its use is still limited, partially because developing control systems from the ground up is inherently challenging, especially for non-domain experts. Through the release of this library, we hope to alleviate the barriers surrounding myoelectric control so that researchers can continue to mature this technology to reach the level of robustness and reliability required for its eventual use as a widely used input modality for a range of interactions and scenarios.

DATA AVAILABILITY

Documentation: Data and code are available online at https://libemg.github.io/libemg.

ACKNOWLEDGMENT

(Ethan Eddy and Evan Campbell are co-first authors.)

REFERENCES

[1] E. J. Scheme and K. Englehart, "Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use," *J. Rehabil. Res. Develop.*, vol. 48, no. 6, pp. 643–659, 2011.

[2] J. Baits, R. Todd, and J. Nightingale, "Paper 10: The feasibility of an adaptive control scheme for artificial prehension," in *Proc. Inst. Mech. Eng. Conf.*, vol. 183. London, U.K.: SAGE, 1968, pp. 54–59.

[3] T. S. Saponas, D. S. Tan, D. Morris, and R. Balakrishnan, "Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces," in *Proc. SIGCHI Conf. Human Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, Apr. 2008, pp. 515–524.

[4] T. S. Saponas, D. S. Tan, D. Morris, J. Turner, and J. A. Landay, "Making muscle-computer interfaces more practical," in *Proc. SIGCHI Conf. Human Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, Apr. 2010, pp. 851–854.

[5] T. S. Saponas, D. S. Tan, D. Morris, R. Balakrishnan, J. Turner, and J. A. Landay, "Enabling always-available input with muscle-computer interfaces," in *Proc. 22nd ACM Annu. Symp. User Interface Softw. Technol. (UIST)*, New York, NY, USA, Oct. 2009, pp. 167–176.

[6] S. Rawat, S. Vats, and P. Kumar, "Evaluating and exploring the MYO ARMBAND," in *Proc. Int. Conf. Syst. Modeling Advancement Res. Trends (SMART)*, Nov. 2016, pp. 115–120.

[7] E. Eddy, E. J. Scheme, and S. Bateman, "A framework and call to action for the future development of EMG-based input in HCI," in *Proc. CHI Conf. Human Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, Apr. 2023, pp. 1–23.

[8] L. J. Hargrove, K. Englehart, and B. Hudgins, "A comparison of surface and intramuscular myoelectric signal classification," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 5, pp. 847–853, May 2007.

[9] A. Tabor, S. Bateman, E. Scheme, D. R. Flatla, and K. Gerling, "Designing game-based myoelectric prosthesis training," in *Proc. CHI Conf. Human Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, May 2017, pp. 1352–1363.

[10] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans. Biomed. Eng.*, vol. 50, no. 7, pp. 848–854, Jul. 2003.

[11] L. Pan, D. Zhang, N. Jiang, X. Sheng, and X. Zhu, "Improving robustness against electrode shift of high density EMG for myoelectric control through common spatial patterns," *J. Neuroeng. Rehabil.*, vol. 12, no. 1, pp. 1–16, Dec. 2015.

[12] A. Phinyomark, E. Campbell, and E. Scheme, "Surface electromyography (EMG) signal processing, classification, and practical considerations," in *Biomedical Signal Processing: Advances in Theory, Algorithms and Applications*. Singapore: Springer, 2020, pp. 3–29.

[13] D. Farina, N. Jiang, H. Rehbaum, A. Holobar, B. Graimann, H. Dietl, and O. C. Aszmann, "The extraction of neural information from the surface EMG for the control of upper-limb prostheses: Emerging avenues and challenges," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 4, pp. 797–809, Jul. 2014.

[14] L. H. Smith, L. J. Hargrove, B. A. Lock, and T. A. Kuiken, "Determining the optimal window length for pattern recognition-based myoelectric control: Balancing the competing effects of classification error and controller delay," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 2, pp. 186–192, Apr. 2011.

[15] E. Scheme, B. Lock, L. Hargrove, W. Hill, U. Kuruganti, and K. Englehart, "Motion normalized proportional control for improved pattern recognition-based myoelectric control," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 1, pp. 149–157, Jan. 2014.

[16] E. Costanza, S. A. Inverso, R. Allen, and P. Maes, "Intimate interfaces in action: Assessing the usability and subtlety of EMG-based motionless gestures," in *Proc. SIGCHI Conf. Human Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, Apr. 2007, pp. 819–828.

[17] E. Costanza, S. A. Inverso, and R. Allen, "Toward subtle intimate interfaces for mobile devices using an EMG controller," in *Proc. SIGCHI Conf. Human Factors Comput. Syst. (CHI)*. New York, NY, USA: Association for Computing Machinery, Apr. 2005, pp. 481–489.

[18] G. R. Naik, D. K. Kumar, V. P. Singh, and M. Palaniswami, "Hand gestures for HCI using ICA of EMG," in *Proc. HCSNet Workshop Vis. Human-Comput. Interact. (VisHCI)*, vol. 56. Docklands, VIC, Australia: Australian Computer Society, Nov. 2006, pp. 67–72.

[19] J. Karolus, A. Kilian, T. Kosch, A. Schmidt, and P. W. Wozniak, "Hit the thumb Jack! Using electromyography to augment the piano keyboard," in *Proc. ACM Designing Interact. Syst. Conf. (DIS)*, New York, NY, USA, Jul. 2020, pp. 429–440.

[20] J. DelPreto and D. Rus, "Plug-and-play gesture control using muscle and motion sensors," in *Proc. 15th ACM/IEEE Int. Conf. Human-Robot Interact. (HRI)*, New York, NY, USA, Mar. 2020, pp. 439–448.

[21] F. Kerber, P. Lessel, and A. Krüger, "Same-side hand interactions with arm-placed devices using EMG," in *Proc. 33rd Annu. ACM Conf. Extended Abstr. Human Factors Comput. Syst. (CHI)*, New York, NY, USA, Apr. 2015, pp. 1367–1372.

[22] M. Wu, Y. Xu, C. Yang, and Y. Feng, "Omnidirectional mobile robot control based on mixed reality and sEMG signals," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2018, pp. 1867–1872.

[23] A. Al-Jumaily and R. A. Olivares, "Electromyogram (EMG) driven system based virtual reality for prosthetic and rehabilitation devices," in *Proc. 11th Int. Conf. Inf. Integr. Web-Based Appl. Services (iiWAS)*. New York, NY, USA: Association for Computing Machinery, Dec. 2009, pp. 582–586.

[24] I. Phelan, M. Arden, M. Matsangidou, A. Carrion-Plaza, and S. Lindley, "Designing a virtual reality myoelectric prosthesis training system for amputees," in *Proc. Extended Abstr. CHI Conf. Human Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–7.

[25] C. Prahm, M. Bressler, K. Eckstein, H. Kuzuoka, A. Daigeler, and J. Kolbenschlag, "Developing a wearable augmented reality for treating phantom limb pain using the Microsoft HoloLens 2," in *Proc. Augmented Humans*. New York, NY, USA: Association for Computing Machinery, Mar. 2022, pp. 309–312.

[26] K. A. Shatilov, D. Chatzopoulos, A. W. T. Hang, and P. Hui, "Using deep learning and mobile offloading to control a 3D-printed prosthetic hand," in *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 3, Sep. 2019, pp. 1–19. [Online]. Available: https://tech.facebook.com/reality-labs/2021/3/inside-facebook-reality-labs-wrist-based-interaction-for-the-next-computing-platform/

[27] L. B. Jaloza, "Inside Facebook reality labs: Wrist-based interaction for the next computing platform," Tech at Meta, Menlo Park, CA, USA, Tech. Rep., Mar. 2021.

[28] D. Ledo, S. Houben, J. Vermeulen, N. Marquardt, L. Oehlberg, and S. Greenberg, "Evaluation strategies for HCI toolkit research," in *Proc. CHI Conf. Human Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 1–17.

[29] M. Kono, Y. Ishiguro, T. Miyaki, and J. Rekimoto, "Design and study of a multi-channel electrical muscle stimulation toolkit for human augmentation," in *Proc. 9th Augmented Human Int. Conf.* New York, NY, USA: Association for Computing Machinery, Feb. 2018, pp. 1–8.

[30] D. Ledo, M. A. Nacenta, N. Marquardt, S. Boring, and S. Greenberg, "The HapticTouch toolkit: Enabling exploration of haptic interactions," in *Proc. 6th Int. Conf. Tangible, Embedded Embodied Interact. (TEI)*. New York, NY, USA: Association for Computing Machinery, Feb. 2012, pp. 115–122.

[31] W. Büschel, A. Lehmann, and R. Dachselt, "MIRIA: A mixed reality toolkit for the in-situ visualization and analysis of spatio-temporal interaction data," in *Proc. CHI Conf. Human Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–15.

[32] S. Seiya, A. Carballo, E. Takeuchi, and K. Takeda, "Deepware: An open-source toolkit for developing and evaluating learning-based and model-based autonomous driving models," *IEEE Access*, vol. 10, pp. 105734–105743, 2022.

[33] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python toolbox for scalable outlier detection," *J. Mach. Learn. Res.*, vol. 20, no. 96, pp. 1–7, Jan. 2019.

[34] T. Westeyn, H. Brashear, A. Atrash, and T. Starner, "Georgia tech gesture toolkit: Supporting experiments in gesture recognition," in *Proc. 5th Int. Conf. Multimodal Interfaces (ICMI)* New York, NY, USA: Association for Computing Machinery, Nov. 2003, pp. 85–92.

[35] N. Gillian and J. A. Paradiso, "The gesture recognition toolkit," *J. Mach. Learn. Res.*, vol. 15, no. 101, pp. 3483–3487, 2014.

[36] S. K. Stigberg, "Mobile hand gesture toolkit: Co-designing mobile interaction interfaces," in *Proc. ACM Conf. Companion Publication Designing Interact. Syst.*, New York, NY, USA, Jun. 2017, pp. 161–166.

[37] J. Haladjian, "The wearables development toolkit: An integrated development environment for activity recognition applications," in *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, Sep. 2020, pp. 1–26.

[38] J. Karolus, F. Kiss, C. Eckerth, N. Viot, F. Bachmann, A. Schmidt, and P. W. Wozniak, "EMBody: A data-centric toolkit for EMG-based interface prototyping and experimentation," in *Proc. ACM Hum.-Comput. Interact.*, vol. 5, May 2021, pp. 1–29.

[39] M. Ortiz-Catalan, R. Brånemark, and B. Håkansson, "BioPatRec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms," *Source Code Biol. Med.*, vol. 8, no. 1, pp. 1–18, Dec. 2013.

[40] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Müller, "Scikit-learn: Machine learning without learning the machinery," *GetMobile, Mobile Comput. Commun.*, vol. 19, no. 1, pp. 29–33, Jun. 2015.

[41] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, May 2011.

[42] M. L. Wilson, W. Mackay, E. Chi, M. Bernstein, D. Russell, and H. Thimbleby, "RepliCHI—CHI should be replicating and validating results more: Discuss," in *Proc. CHI Extended Abstr. Human Factors Comput. Syst. (CHI EA)*. New York, NY, USA: Association for Computing Machinery, May 2011, pp. 463–466.

[43] F. Echtler and M. Häußler, "Open source, open science, and the replication crisis in HCI," in *Proc. Extended Abstr. CHI Conf. Human Factors Comput. Syst.*, New York, NY, USA, Apr. 2018, pp. 1–8.

[44] A. Phinyomark, P. Phukpattaranont, and C. Limsakul, "Feature reduction and selection for EMG signal classification," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7420–7431, Jun. 2012.

[45] A. Phinyomark, R. N. Khushaba, E. Ibáñez-Marcelo, A. Patania, E. Scheme, and G. Petri, "Navigating features: A topologically informed chart of electromyographic features space," *J. Roy. Soc. Interface*, vol. 14, no. 137, Dec. 2017, Art. no. 20170734.

[46] A. Phinyomark, R. N. Khushaba, and E. Scheme, "Feature extraction and selection for myoelectric control based on wearable EMG sensors," *Sensors*, vol. 18, no. 5, p. 1615, May 2018.

[47] A. Phinyomark and E. Scheme, "A feature extraction issue for myoelectric control based on wearable EMG sensors," in *Proc. IEEE Sensors Appl. Symp. (SAS)*, Mar. 2018, pp. 1–6.

[48] M. Zardoshti-Kermani, B. C. Wheeler, K. Badie, and R. M. Hashemi, "EMG feature evaluation for movement control of upper extremity prostheses," *IEEE Trans. Rehabil. Eng.*, vol. 3, no. 4, pp. 324–333, Dec. 1995.

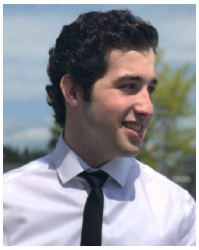
[49] S. Thongpanja, A. Phinyomark, P. Phukpattaranont, and C. Limsakul, "Mean and median frequency of EMG signal to determine muscle force based on time-dependent power spectrum," *Electron. Electr. Eng.*, vol. 19, no. 3, pp. 51–56, Mar. 2013.

[50] S. Du and M. Vuskovic, "Temporal vs. spectral approach to feature extraction from prehensile EMG signals," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, Nov. 2004, pp. 344–350.

[51] J. He, D. Zhang, X. Sheng, S. Li, and X. Zhu, "Invariant surface EMG feature against varying contraction level for myoelectric control based on muscle coordination," *IEEE J. Biomed. Health Informat.*, vol. 19, no. 3, pp. 874–882, May 2015.

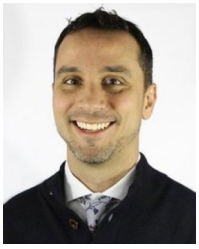
[52] S. Arjunan and D. Kumar, "Decoding subtle forearm flexions using fractal features of surface electromyogram from single and multiple sensors," *J. Neuroeng. Rehabil.*, vol. 7, no. 1, p. 53, 2010.
[53] S. Thongpanja, A. Phinyomark, F. Quaine, Y. Laurillau, C. Limsakul, and P. Phukpattaranont, "Probability density functions of stationary surface EMG signals in noisy environments," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 7, pp. 1547–1557, Jul. 2016.
[54] F. Onay and A. Mert, "Phasor represented EMG feature extraction against varying contraction level of prosthetic control," *Biomed. Signal Process. Control*, vol. 59, May 2020, Art. no. 101881.
[55] K. S. Kim, H. H. Choi, C. S. Moon, and C. W. Mun, "Comparison of k-nearest neighbor, quadratic discriminant and linear discriminant analysis in classification of electromyogram signals based on the wrist-motion directions," *Current Appl. Phys.*, vol. 11, no. 3, pp. 740–745, May 2011.
[56] S. Pancholi and A. M. Joshi, "Time derivative moments based feature extraction approach for recognition of upper limb motions using EMG," *IEEE Sensors Lett.*, vol. 3, no. 4, pp. 1–4, Apr. 2019.
[57] M. G. A. Samuel, O. W. Samuel, Y. Geng, P. O. Idowu, S. Chen, N. Ganesh R, P. Feng, and G. Li, "Enhancing the robustness of EMG-PR based system against the combined influence of force variation and subject mobility," in *Proc. 3rd Asia–Pacific Conf. Intell. Robot Syst. (ACIRS)*, Jul. 2018, pp. 12–17.
[58] R. N. Khushaba, L. Shi, and S. Kodagoda, "Time-dependent spectral features for limb position invariant myoelectric pattern recognition," in *Proc. Int. Symp. Commun. Inf. Technol. (ISCIT)*, Oct. 2012, pp. 1015–1020.
[59] B. Hjorth, "EEG analysis based on time domain properties," *Electroencephalogr. Clin. Neurophysiol.*, vol. 29, no. 3, pp. 306–310, Sep. 1970.
[60] A. Mengarelli, A. Tigrini, S. Fioretti, S. Cardarelli, and F. Verdini, "On the use of fuzzy and permutation entropy in hand gesture characterization from EMG signals: Parameters selection and comparison," *Appl. Sci.*, vol. 10, no. 20, p. 7144, Oct. 2020.
[61] J. S. Richman and J. R. Moorman, "Physiological time-series analysis using approximate entropy and sample entropy," *Amer. J. Physiol.-Heart Circulatory Physiol.*, vol. 278, no. 6, pp. H2039–H2049, Jun. 2000.
[62] S.-H. Park and S.-P. Lee, "EMG pattern recognition based on artificial intelligence techniques," *IEEE Trans. Rehabil. Eng.*, vol. 6, no. 4, pp. 400–405, Dec. 1998.
[63] A. Phinyomark, F. Quaine, S. Charbonnier, C. Serviere, F. Tarpin-Bernard, and Y. Laurillau, "Feature extraction of the first difference of EMG time series for EMG pattern recognition," *Comput. Methods Program. Biomed.*, vol. 117, no. 2, pp. 247–256, Nov. 2014.
[64] R. N. Khushaba, S. Kodagoda, S. Lal, and G. Dissanayake, "Driver drowsiness classification using fuzzy wavelet-packet-based feature-extraction algorithm," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 1, pp. 121–131, Jan. 2011.
[65] B. Hudgins, P. Parker, and R. N. Scott, "A new strategy for multifunction myoelectric control," *IEEE Trans. Biomed. Eng.*, vol. 40, no. 1, pp. 82–94, Jan. 1993.
[66] G.-C. Chang, W.-J. Kang, J.-J. Luh, C.-K. Cheng, J.-S. Lai, J.-J.-J. Chen, and T.-S. Kuo, "Real-time implementation of electromyogram pattern recognition as a control command of man-machine interface," *Med. Eng. Phys.*, vol. 18, no. 7, pp. 529–537, Oct. 1996.
[67] D. Tkach, H. Huang, and T. A. Kuiken, "Study of stability of time-domain features for electromyographic pattern recognition," *J. Neuroeng. Rehabil.*, vol. 7, no. 1, p. 21, May 2010.
[68] J. L. Nawfel, K. B. Englehart, and E. J. Scheme, "A multi-variate approach to predicting myoelectric control usability," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1312–1327, 2021.
[69] E. J. Scheme, B. S. Hudgins, and K. B. Englehart, "Confidence-based rejection for improved pattern recognition myoelectric control," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 6, pp. 1563–1570, Jun. 2013.
[70] M. F. Wahid, R. Tafreshi, and R. Langari, "A multi-window majority voting strategy to improve hand gesture recognition accuracies using electromyography signal," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 2, pp. 427–436, Feb. 2020.
[71] B. Lock, K. Englehart, and B. Hudgins, "Real-time myoelectric control in a virtual environment to relate usability vs. accuracy," in *Proc. MyoElectric Controls/Powered Prosthetics Symp.*, Fredericton, NB, Canada, Jan. 2005, pp. 1–4.
[72] L. Hargrove, Y. Losier, B. Lock, K. Englehart, and B. Hudgins, "A real-time pattern recognition based myoelectric control usability study implemented in a virtual environment," in *Proc. 29th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2007, pp. 4842–4845.
[73] E. Campbell, A. Phinyomark, and E. Scheme, "Current trends and confounding factors in myoelectric control: Limb position and contraction intensity," *Sensors*, vol. 20, no. 6, p. 1613, Mar. 2020.
[74] D. R. Rogers and D. T. MacIsaac, "A comparison of EMG-based muscle fatigue assessments during dynamic contractions," *J. Electromyogr. Kinesiol.*, vol. 23, no. 5, pp. 1004–1011, Oct. 2013.
[75] I. S. MacKenzie and R. J. Teather, "FittsTilt: The application of Fitts' law to tilt-based interaction," in *Proc. 7th Nordic Conf. Human-Comput. Interact., Making Sense Through Design*. New York, NY, USA: Association for Computing Machinery, Oct. 2012, pp. 568–577.
[76] U. Côté-Allard, E. Campbell, A. Phinyomark, F. Laviolette, B. Gosselin, and E. Scheme, "Interpreting deep learning features for myoelectric control: A comparison with handcrafted features," *Frontiers Bioeng. Biotechnol.*, vol. 8, p. 158, Mar. 2020.
[77] E. Campbell, A. Phinyomark, and E. Scheme, "Deep cross-user models reduce the training burden in myoelectric control," *Frontiers Neurosci.*, vol. 15, May 2021, Art. no. 657958.
[78] Y. Lin, R. Palaniappan, P. De Wilde, and L. Li, "A normalisation approach improves the performance of inter-subject sEMG-based hand gesture recognition with a ConvNet," in *Proc. 42nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2020, pp. 649–652.
[79] B. Xue, L. Wu, A. Liu, X. Zhang, X. Chen, and X. Chen, "Reduce the user burden of multiuser myoelectric interface via few-shot domain adaptation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 972–980, 2023.
[80] J. W. Sensinger, B. A. Lock, and T. A. Kuiken, "Adaptive pattern recognition of myoelectric signals: Exploration of conceptual framework and practical algorithms," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 17, no. 3, pp. 270–278, Jun. 2009.
[81] J. Berman, R. Hinson, and H. Huang, "Comparing reinforcement learning agents and supervised learning neural networks for EMG-based decoding of continuous movements," in *Proc. 43rd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Nov. 2021, pp. 6297–6300.
[82] M. A. Oskoei and H. Hu, "Myoelectric control systems—A survey," *Biomed. Signal Process. Control*, vol. 2, no. 4, pp. 275–294, Oct. 2007.
[83] A. Ameri, M. A. Akhaee, E. Scheme, and K. Englehart, "A deep transfer learning approach to reducing the effect of electrode shift in EMG pattern recognition-based control," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 2, pp. 370–379, Feb. 2020.
[84] S. Pizzolato, L. Tagliapietra, M. Cognolato, M. Reggiani, H. Müller, and M. Atzori, "Comparison of six electromyography acquisition setups on hand movement classification tasks," *PLoS ONE*, vol. 12, Oct. 2017, Art. no. e0186132.
[85] R. Leeb, H. Sagha, R. Chavarriaga, and J. D. R. Millán, "A hybrid brain–computer interface based on the fusion of electroencephalographic and electromyographic activities," *J. Neural Eng.*, vol. 8, no. 2, Mar. 2011, Art. no. 025011.
[86] L. Lu, J. Mao, W. Wang, G. Ding, and Z. Zhang, "A study of personal recognition method based on EMG signal," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 4, pp. 681–691, Aug. 2020.
[87] J. Karolus, S. Thanheiser, D. Peterson, N. Viot, T. Kosch, A. Schmidt, and P. W. Wozniak, "Imprecise but fun: Playful interaction using electromyography," in *Proc. ACM Hum.-Comput. Interact.*, vol. 6, Sep. 2022, pp. 1–12.
[88] B. Coffen and Md. S. Mahmud, "TinyDL: Edge computing and deep learning based real-time hand gesture recognition using wearable sensor," in *Proc. IEEE Int. Conf. E-Health Netw., Appl. Services (HEALTHCOM)*, Mar. 2021, pp. 1–6.
[89] M. Kim, J. Cho, S. Lee, and Y. Jung, "IMU sensor-based hand gesture recognition for human-machine interfaces," *Sensors*, vol. 19, no. 18, p. 3827, Sep. 2019.
[90] A. Bablani, D. R. Edla, D. Tripathi, and R. Cheruku, "Survey on brain-computer interface: An emerging computational intelligence paradigm," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–32, Feb. 2019.
[91] X. Zheng, X. Liu, Y. Zhang, L. Cui, and X. Yu, "A portable HCI system-oriented EEG feature extraction and channel selection for emotion recognition," *Int. J. Intell. Syst.*, vol. 36, no. 1, pp. 152–176, Jan. 2021.
[92] J. A. Healey and R. W. Picard, "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 2, pp. 156–166, Jun. 2005.
[93] E. Campbell, A. Phinyomark, and E. Scheme, "Feature extraction and selection for pain recognition using peripheral physiological signals," *Frontiers Neurosci.*, vol. 13, p. 437, May 2019.

ETHAN EDDY (Member, IEEE) received the B.Sc. degree in software engineering from the University of New Brunswick, in 2021, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. He is also a part of the Institute of Biomedical Engineering, Human-Computer Interaction (HCI) Laboratory, and the Spatial Computing Training and Research (SPECTRAL) Laboratory. His research interests include human–computer interaction, ubiquitous computing, mixed reality, machine learning, and myoelectric control.

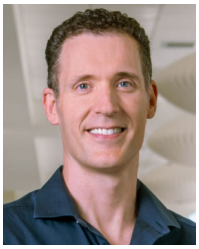


EVAN CAMPBELL (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from the University of New Brunswick, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering. He has been a member of the Institute of Biomedical Engineering, since 2018. His research interests include deep learning, reinforcement learning, and signal processing applied to myoelectric control.

ANGKOON PHINYOMARK (Member, IEEE) received the B.Eng. degree (Hons.) in computer engineering and the Ph.D. degree in electrical engineering from the Prince of Songkla University (PSU), Hat Yai, Thailand, in 2008 and 2012, respectively. From 2012 to 2013, he was a Postdoctoral Research Fellow with GIPSA and LIG Laboratories, Université Grenoble Alpes, Grenoble, France. From 2013 to 2016, he was a Postdoctoral Research Fellow with the Human Performance Laboratory, University of Calgary, Calgary, AB, Canada. From 2016 to 2017, he was a Researcher with the ISI Foundation, Turin, Italy. Since 2017, he has been with the Institute of Biomedical Engineering, University of New Brunswick, Fredericton, NB, Canada, where he is currently a Senior Research Scientist. His research interests include biomedical signal processing and machine learning, wearable sensors, gait biomechanics and biometrics, and neuroscience.



SCOTT BATEMAN received the master's and Ph.D. degrees in computer science from the University of Saskatchewan, in 2008 and 2013, respectively. He was a Researcher with the Microsoft Research, IBM Watson Research, University of Calgary, and the National College of Ireland. He joined UNB, in 2015, after 2.5 years with the University of Prince Edward Island. He is currently an Associate Professor of computer science, the Co-Director of the Human-Computer Interaction Laboratory, and the Director of the Spatial Computing Research (SPECTRAL) Laboratory, University of New Brunswick. His research interests include novel input devices and interactions, mixed reality, computer-based collaboration, and games for learning and health. He was an Associate Editor and the Program Chair for several leading international journals and conferences.



ERIK SCHEME (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the University of New Brunswick (UNB), Fredericton, NB, Canada, in 2003, 2005, and 2013, respectively. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, the Associate Director of the Institute of Biomedical Engineering, and the Director of the Health Technologies Laboratory, UNB. His research interests include application of signal processing and machine learning for human–machine interfaces, mobility and rehabilitation, digital health and diagnostics, and biometrics. He is a Registered Member of the Association of Professional Engineers and Geoscientists of New Brunswick (APEGNB).