A Versatile User-Oriented Interface for Gesture Recognition Using Electromyographic and Inertial Measurement Unit Data

Jeffrey Yan[1], James Dalton[1], Bianca Doronila[1], Kattia Chang-Kam[1], Victor Melara[1]
Dr. Xiaorong Zhang[2], Ian Donovan[2]
[1]Cañada College
[2]School of Engineering, San Francisco State University

Abstract

Gestures are a natural component of human expression. Research surrounding electromyographic (EMG) signals has shown that gestures can be quantified and classified by using feature extraction and pattern recognition methods. There exists a wide range of applications for which an accurate, real-time gesture recognition interface could prove critically beneficial. Our research was focused on processing EMG and Inertial Measurement Unit (IMU) data to develop such an interface. Our aim was to develop a user-oriented and versatile interface compatible for use with any third-party application that might benefit from gesture recognition. Customizability of the interface allowed it to be optimized for the unique EMG and IMU signals of each individual user. Our interface acquired arm gesture data from a Myo armband (Thalmic Labs), which records the raw EMG signals of its wearer's arm at 200 Hz, and the IMU data of the motion at 50 Hz. The armband's low cost and ease of use was consistent with our project's user-friendly goals. Various algorithmic classifications were performed on the Myo armband input data to yield a gesture decision. This in turn could be passed as input to a third party application. Implementation of our design involved sending gesture decisions from the interface to a virtual reality application that would interpret them as control input.

I. Introduction

In recent years, gesture recognition has gained popularity in several fields, as it provides a potential solution to a variety of problems. Research has been done on hand gesture recognition for a defined sign language library, in order to help the deaf communicate more conveniently [15]. Gesture recognition has also been applied to assist persons with limited mobility, enabling the use of gestures as control inputs for wheelchairs [13] and other assistive devices. Studies involving virtual reality environments have made use of gesture-based control schemes [8], as they allow for a natural, immersive user experience.

Whatever the intended use, gesture recognition requires some means by which to track a subject's movements and position. Previous works have used computer-vision to fill this need, with impressive results. The effectiveness of this approach is however limited by the capabilities of the camera—lighting must be adequate and the motions of interest must be performed within its field of view. Motion-sensing gloves have also been used in hand gesture recognition, however these devices are typically bulky and inconvenient for the subject to wear. A more common approach makes use of biosignals to determine the subject's movements and position. The electrical potential along a person's muscles, commonly known as electromyographic

(EMG) signals, have been shown to be intrinsically tied to the person's intended movements. EMG signals can be captured with relative ease by attaching pairs of electrodes along the subject's muscles.

Figure 1. Myo Armband.

This research project was focused on expanding upon an existing human-machine interface (HMI) for gesture recognition, with the intent of improving its usability and functionality. The interface was developed as an easy to use, open-source software that processes EMG signals from the user's arm to produce gesture decisions. EMG signals were chosen as a means to represent the movement and position of the user. They have shown impressive results in previous gesture recognition studies, and they can be recorded easily without much discomfort to the user. The low-cost commercially available Myo Armband device (Thalmic Labs) shown in Fig. 1 was chosen to provide EMG input to our interface. This device records eight channels of EMG data at 200 Hz, and collects kinematic data at 50 Hz via an inertial measurement unit (IMU). The IMU consists of a 3-axis magnetometer, gyroscope, and accelerometer.

The development of the interface has been a collaborative effort between several contributors, over the span of a year. The first build of the interface by Razi [11] had many of the interface's main functions successfully implemented. It was able to process EMG data from the armband and recognize the user's gesture, then display this gesture decision as a text output on the interface display. An overhaul was later done to improve the design and architecture of the interface. Figure 2 details foundational structure of the interface.

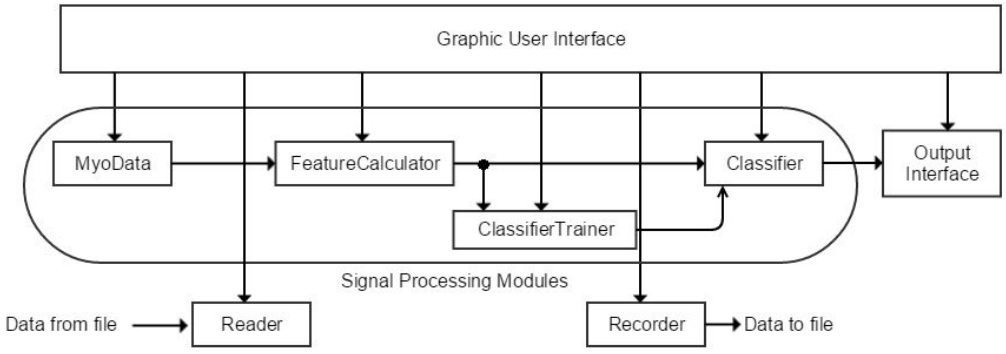

Figure 2. Previous version of HMI structure

A. MyoData

The Myo armband provides raw EMG data as an input for the HMI. This unprocessed data tends to include a lot of noise, either from external interference or motion of the sensors against the surface of the user's skin. In this state, the signal is not useful for gesture recognition. To prepare

the signal to be better interpreted by processor, the signal is segmented into several small windows of discrete samples.

B. FeatureCalculator

Data Windowing

Hudgins states that to preserve the structural information of the signals, a set of features pertaining to the signal can be extracted and analyzed for better accuracy during gesture classification [5]. To extract these features, the signals are segmented into windows. There are two techniques: adjacent and overlapping. For the first one, the window size and the increment lengths are the same. In contrast, the increment in overlapping windowing is smaller than the window size. The latter method is used in our HMI as shown in Fig. 3. According to Englehart, its advantage over the other procedure is that it allows for larger data blocks without the compromise of increased latency [3]. It also enables the processor to interpret the next window w_{i+1} without having to wait for the current window w_i to completely fill.

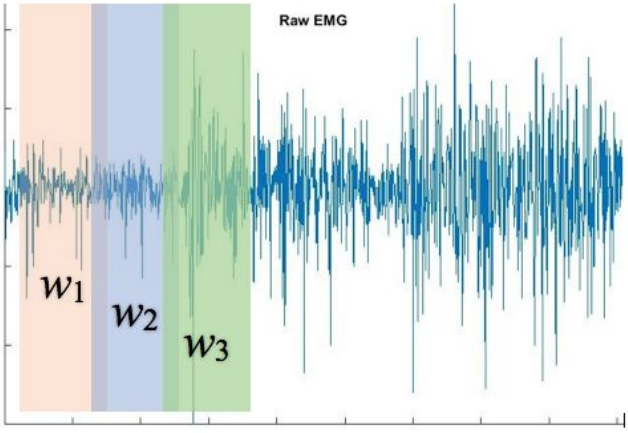

Figure 3. Overlapped Data Windows.

Feature Extraction

The HMI is able to extract the following time-domain features: mean absolute value (*MAV*), waveform length (*WAVE*), zero crossings (*ZERO*), and slope change sign (*TURN*).
Figure 4 shows the extraction of the *MAV* and *ZERO* features from two channels of the armband for three different gestures.

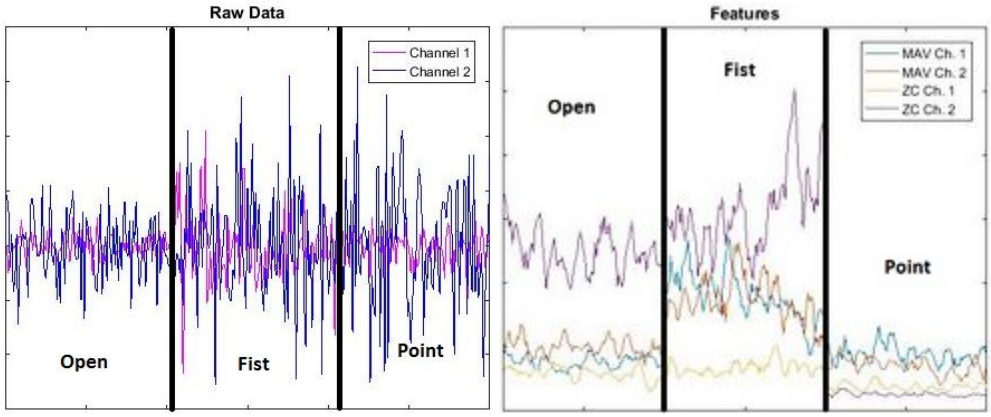

Figure 4. MAV and ZERO (ZC in figure) features from two channels are being extracted from gestures open, fist, and point.

The mean absolute value is calculated for each window. This is the sum of the absolute value of each sample in the window divided by the window length. According to Phinyomark, this feature detects the muscle contraction level [9]. Equation 1 is used to describe *MAV*:

$$MAV_i = \frac{1}{N}\sum_{k=1}^{N}|x_k| \quad for\ i = 1, \ldots, I \quad (1)$$

According to Hudgins, waveform length indicates the total length of the waveform over a period of time. This feature gives information about the amplitude, frequency, and time of the segment [5]. The calculation of this feature is as follows:

$$WAVE = \sum_{k-1}^{N}|x_k - x_{k-1}| \quad (2)$$

Zero crossings calculates the amount of instances; in which, the filtered signal crosses zero. To effectively implement this feature, a threshold must be set up to filter the noise that is present in the signal. The conditions shown in Equation 3 demonstrate how to implement it.

$$(x_k > 0 \ and \ x_{k+1} < 0) \ or \ (x_k < 0 \ and \ x_{k+1} > 0), \ and \ |x_k - x_{k-1}| \geq threshold \quad (3)$$

Slope change sign determines the amount of times the slope of the waveform has changed signs, that means from positive to negative or negative to positive. To implement this calculation, three data points are taken from the signal. If the middle point of this data set is greater than the other two, there is a maximum point (positive to negative); in contrast, if the middle point is less than the other two points, there is a minimum (negative to positive). This feature also needs a threshold to avoid incorrect readings. Equation 4 shows the conditions for this feature.

$$(x_k > x_{k-1} \text{ and } x_k > x_{k+1}) \text{ or } (x_k < x_{k-1} \text{ and } x_k < x_{k+1}), \text{ and}$$
$$|x_k - x_{k-1}| \geq \text{threshold or } |x_k - x_{k+1}| \geq \text{threshold} \quad (4)$$

C. ClassifierTrainer and Classifier

The classifiers used in the Human-Machine Interface were chosen because they have been proven to be an effective way to do classification, moreover each classifier offers a different approach to calculate the difference between the data sets [7][14].

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) was first developed by Ronald Fisher in the 1930s. Fisher's idea is a very simple algorithm. Let's suppose there are two data sets one called $S^1 = \{s_1^1, s_2^1, s_3^1, \ldots, s_{k_1}^1\}$ and $S^2 = \{s_1^2, s_2^2, s_3^2, \ldots, s_{k_1}^2\}$; both of these data samples represent the entire scope of the data sample *S*. Now that we have these data samples we want to maximize a vector, say **v**. In this vector there is a direction associated with the projection of the mean of all the data sample and the different classes. Therefore, once this direction is found the variance between the class must be optimized. This is done by minimizing the variance in the direction, in which the mean of all the classes is project on. Once these two components have been found the significance of the vector **v** comes into play. Vector **v** represents the ratio between the mean over the variance between the classes. This vector then creates a decision boundary between the different classes [7]. Figure 5 shows three different classes being classified by the LDA, therefore each of the classes are being group together based on their individual means. Once these grouping is clear, the LDA creates the dividing linear vector; this is denoted on Fig. 5 by the red and black lines. .

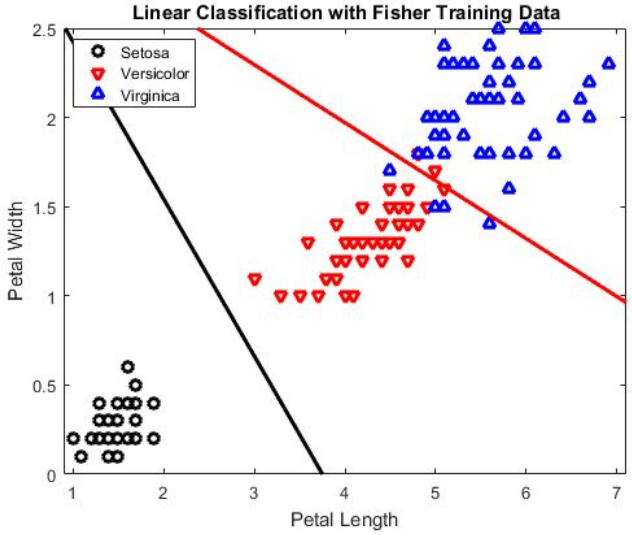

Figure 5. The Mathworks, Inc provides an online documentation which includes a tutorial on Discriminant Analysis in which they describe their software implementation as well as ways in which the LDA can be modified.

Support Vector Machine

The Support Vector Machine (SVM) algorithm at its most basic is a high dimensional classifier. The SVM needs a set of training data in order for it to start making classifications. Suppose the SVM is fed with a sample data of K dimensions. These data points will become the support vectors as they are the reference points in which a margin or a street dividing the different classes will be built. This set of data will then be the components for the dividing hyperplane, which will be of a higher dimension than the K dimension data set [4]. In terms of linear algebra, the equation for data sets in x that lie on the hyperplane comply $u \cdot x + b = 0$, in which u is normal to the hyperplane [1]. The ratio between $|b|$ and $\|u\|$ is the perpendicular distance from the origin to the hyperplane. $\|u\|$ is the Euclidian norm of the vector **u**. In other words, it is the total length in that vector space in the form of a straight line. In any case, the SVM algorithm calculates the best hyperplane with the largest margin to separate the different classes. Figure 6 shows a simple example in which there is two different sets of classes and they are being separated by a hyperplane. The hyperplane in Fig. 6 is a line because the classes are a one-dimensional vector; therefore, its hyperplane would be a two-dimensional vector, in this case a dotted red line. The classes, the little crosses, circled in green are the support vector chosen by the algorithm to construct the margin between the different classes.

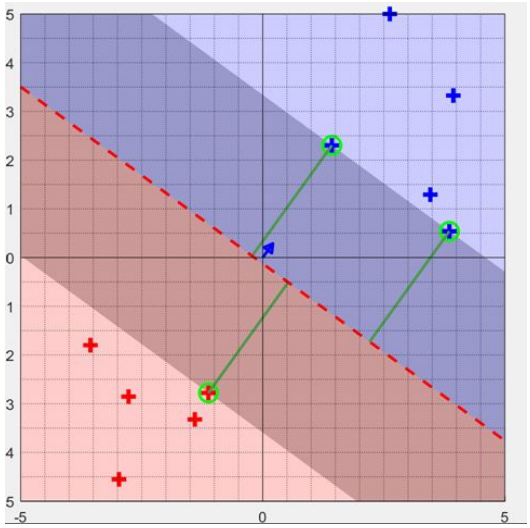

Figure 6. The graph is provided by software designed by Richard Stapenhurst which generates different graphs of SVM and the different iterations of it. The software is freely available for the community at The MathWorks, Inc. Here the image clearly defines the support vector (those circle in green), the margin (the darker shaded regions) and the hyperplanes (the dotted red line inside the shaded region).

II. Design and Implementation

A. Expansion upon HMI Gesture Recognition Software

Figure 7 shows the additions made to the HMI in efforts to accomplish the proposed goals. The blue boxes indicate the additions or changes made and the arrows help indicate in which part of the HMI were those changes were applied.

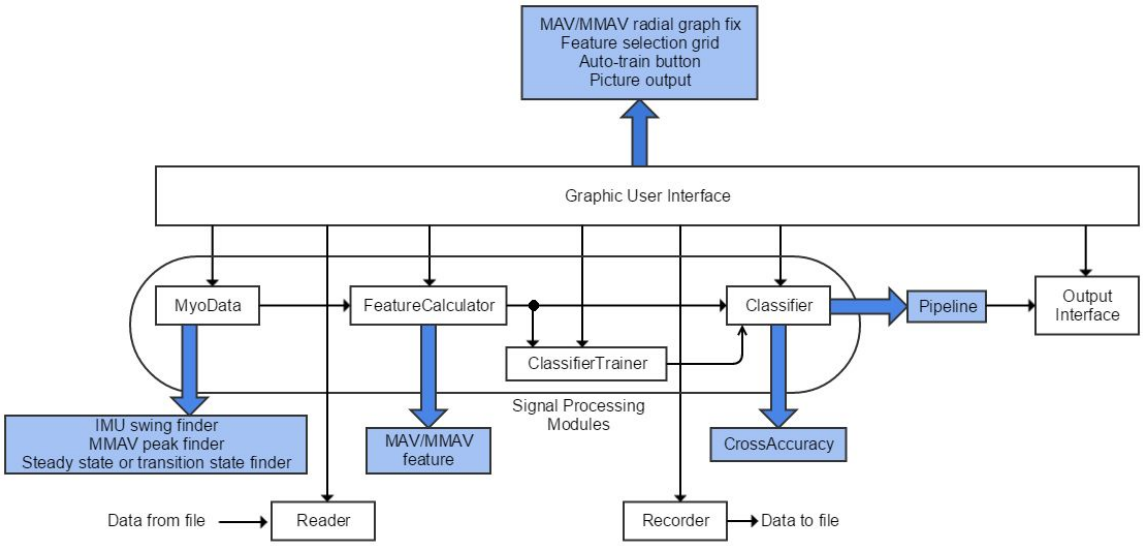

Figure 7. Human machine interface structure. Blue items indicate features that have been added.

FeatureCalculator

An additional EMG feature was added to *FeatureCalculator* called scaled mean absolute value (*MAV/MMAV*). This feature is calculated for each channel. Equation 6 shows the calculation for this feature:

$$MAV/MMAV = MAV_i / \left(\frac{1}{I}\sum_{i=1}^{I} MAV_i\right) \qquad (6)$$

Where N is the number of samples in the window and I is the number of EMG signal channels. *MMAV* is the average of the sum of the *MAV* across all channels. Then, the *MAV* value of each channel is divided over this result.

Classifier

The *classifier* module had the *crossValidation* function added. This function was originally only implemented in the SVM model. This made it impossible to generate the confusion matrix by using the LDA model because the HMI would stop working. The function was moved from the *SVMAgent* file to the classifier file, so that both models would have access to use this function via inheritance. Also, the function was rewritten so that it would take 20% of the data passed to it to train the selected model, and the other 80% to make predictions and compare them to their actual values. Thus, both models would output more realistic classifications. Figure 8 shows the HMI using the SVM model and Figure 9 shows a trained LDA model.

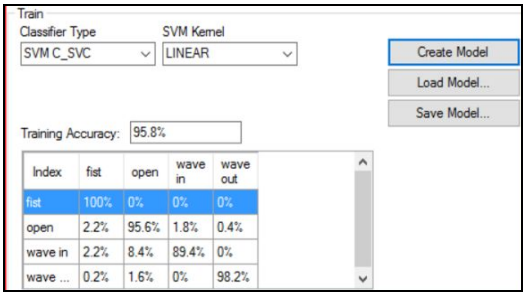

Figure 8. Confusion matrix output based on the SVM model with four gestures: fist, open, wave in, and wave out.

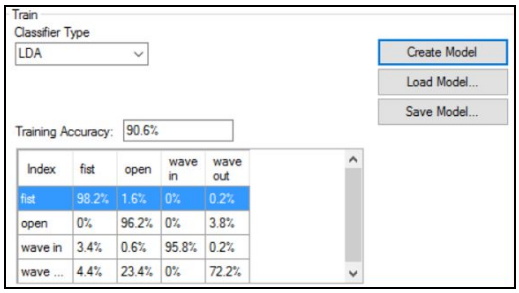

Figure 9. Confusion matrix output based on the LDA model with four gestures: fist, open, wave in, and wave out.

Graphical User Interface (GUI)

Overall, the GUI had most of the changes done to it. In the EMG tab, the *MAV/MMAV* feature was added to the radial graphs; additionally, a bug related to those graphs were fixed. Originally, if the EMG values were greater than the maximum values of the graphs, the EMG data would not display. Now, if the value is greater than the graph's maximum value, the EMG data is shown as the maximum value. This is shown in Fig. 10.

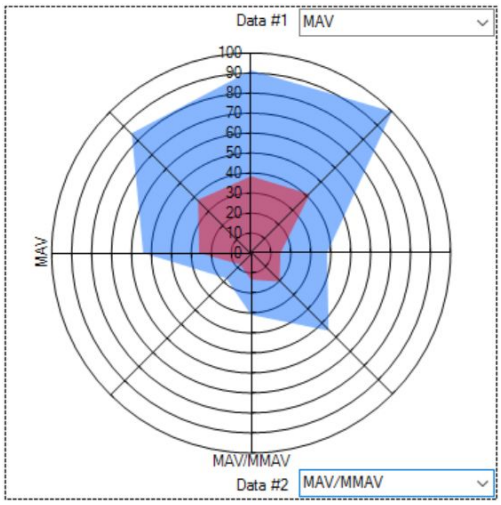

Figure 10. Fixed radial graph showing MAV and MAV/MMAV EMG signals.

Additionally, as shown in Fig. 11, we added a feature selection grid, which allows the user to determine which features from which channel to pass to the classifier. It also gives flexibility to the HMI as more features can be added or removed in future versions of the HMI.

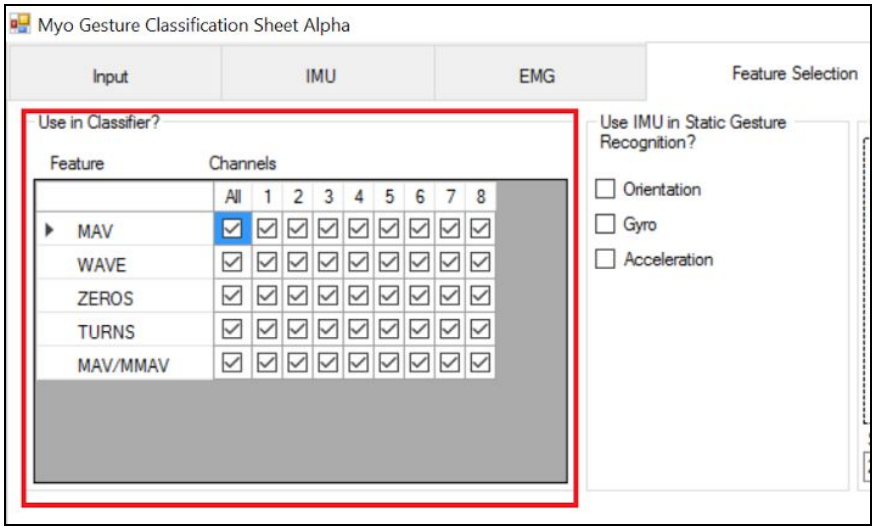

Figure 11. Feature selection grid is shown inside the red box. Here the number of channels and types of gestures to be used by the classifier can be selected with ease.

An auto-run button was added, which makes the training and data gathering process easier. The user doesn't need to continuously interact with the HMI. The feature gives the user the ability to set a specific delay time between training two features. Another feature present in the classifier tab is the picture output, which was added back to the HMI; it was removed during the code clean up process. Both features can be seen in Fig. 12.

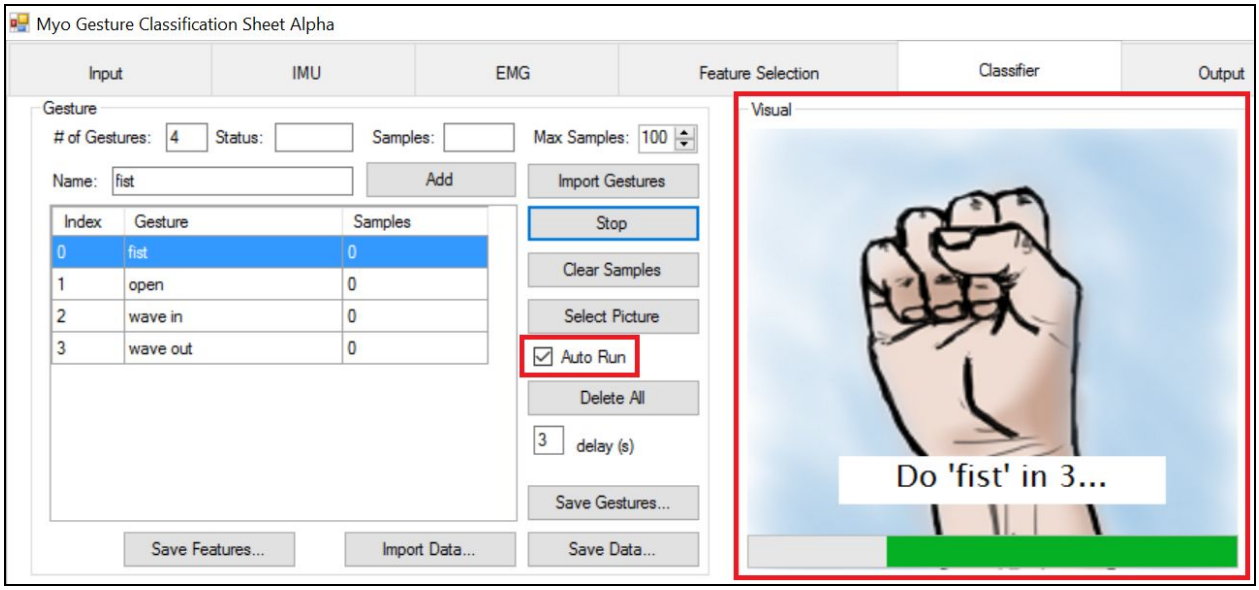

Figure 12. Auto-run button and picture output features in the current version of the HMI.

MyoData

The addition of an accurate dynamic gesture recognition module to our interface was a goal for this study, but due to time constraints, we were only able to implement it at a very basic level. Taking advantage of the IMU data fed into our interface, we built an arm swing detector that recognized when the user's arm was swung. This feature was implemented by taking a running *k*-average of the most recent *k* samples of IMU orientation data, where *k* may be set by the user. A swing would be recognized for each time orientation crossed this average.

We did consider using Hidden Markov Models (HMM) to implement a more robust dynamic gesture recognition system. Our approach would require an algorithmic method discriminate steady states (or moments of relatively steady behavior) from transitional states (transitions between steady states) within the motion trajectories of dynamic gestures (for further explanation, see ***Future Works***). To find these steady states, we measure the standard deviation of the mean absolute value of the EMG signal in short time windows:

$$STD_{window} = \sqrt{\sum_{k=1}^{N} \sigma_k^2} \qquad (5)$$

where N is the number of EMG channels and σ_k is the standard deviation of the mean absolute values of the *k*th channel in the window. If the standard deviation falls below some threshold, we assume the user's motion has entered a steady state. The optimal value of this threshold depends on the user, and can be changed within the interface.

B. Interaction with Client Applications

A pipeline was developed in order to give the ICE HMI the capability to output gesture classifications and IMU data to a client application. The client application can then use this data for its desired purposes. A virtual reality video game project was developed in conjunction with this project that implemented this pipeline to use gesture output from the HMI for control purposes inside the game as opposed to standard keyboard/mouse control.

The game naturally accepts discrete keystroke commands as input. One issue in interfacing the HMI with the game was that the HMI was designed to output a continuous stream of gesture classifications. The challenge was in finding a means by which to pick the correct classifications out of this continuous stream to feed to the game as keystroke commands. The method implemented used the previously mentioned windowed standard deviation of the MMAV of the EMG signals to send the latest classification after this value had dropped below some user defined threshold.

Currently, the HMI pipeline has the ability to output a 3-byte package to a client application with each byte representing a specific value: the index of the most recent gesture classification, the average magnitude of that gesture, and an 8-bit byte with each of the bits representing either a gesture command or information from the IMU (i.e. arm swings). Each bit can be toggled to represent the current state of the gesture that was assigned to that bit. Three control schemes to

set/reset a bit have been implemented: continuously (bit set for as long as the gesture is being classified), when entering a steady state of the gesture, or when an MMAV peak of the gesture has occurred (i.e. the user has made a tense of the gesture).

Upon successfully receiving the package, the client application sends back a single byte as an acknowledgement to the server (HMI), this way the client can give feedback to the server. The HMI currently can receive a number of different acknowledgements which are specifically defined to vibrate the Myo armband, providing haptic feedback to the user.

C. Experimental Protocol

As previously stated, a virtual reality video game was developed in conjunction with this project as an application of the ICE HMI. Certain gesture classifications and arms swings from the IMU were streamed to the VR game through the pipeline. The game uses this information from the HMI as a means of control. For example, if the player makes a fist gesture, the character inside the game will fire a weapon. Likewise, if the player swings his or her arm, the character inside the game will take a step.

Figure 13. Gameplay of virtual reality game for usability assessment platform

The VR game was used as a usability assessment platform to test the feasibility of using gesture classifications and IMU data from the HMI to control the game as opposed to a standard keyboard/mouse setup. The game is a first-person shooter (FPS) with the objective to go from start to finish, without dying from the horde of zombies attacking the player, killing as many zombies as possible. The metrics used in the usability test included: firing accuracy (hit-to-miss ratio), the final score of the match (based on the number of zombies killed), and the remaining health of the player.

Four gestures were trained to control the game: fist (fire weapon), rock on (change weapon), fist right (reload weapon), and index, middle, and ring fingers up (toggle vehicle headlight). In addition to the four gestures, arm swings were used to control a walking portion of the game, each arm swing equates to a one step inside the game.

Eleven subjects participated in the experiment (9 male, 2 female). A pre-survey was given to each subject to collect information about his or her age range, gender, and whether or not the subject had prior experience with virtual reality or first-person shooters. Each subject conducted two trials, one for each control scheme: using mouse only and using the Myo armband only. Prior to each trial, the subject was given a training session on how to play the game. Before the Myo armband trial, each subject had to train each of the four gestures in the ICE HMI to ensure accuracy of gesture classification. The subject then put on an Oculus Rift virtual reality headset and played each trial of the game using one of the control schemes. Each subject then filled out a post-survey meant to gauge the subject's experience in using the two control methods.

III. Results and Discussion

A. Experiment Results

On average, the trials produced higher scores with the mouse controls than with gesture controls (see Fig. 14). In addition, only four out of the eleven subjects were able to complete the game without their character dying when using gesture controls, whereas all but two of the subjects completed the game with health remaining using the mouse. The post-surveys suggest that subjects found gesture controls to be difficult to use. 50% of the respondents answered that the gesture controls were either hard or very hard to use, whereas only 10% answered that they were easy to use.

	Mouse Controls	**Gesture Controls**
Score	199.09	104.46
Final health	46.818	16.36
Accuracy	65.56	50.86

Figure 14. Average Results of VR Implementation Tests

B. Discussion

There are many possible contributing factors to the overall poor usability of gesture controls, the most likely being flawed implementation. The control system that we built into the interface was by no means intuitive or natural. The performance of one gesture would often not register, or trigger an unintended command. Perhaps measuring the windowed MMAV standard deviation against some static threshold was an ineffective way to define steady state. A later version of the interface might benefit from using a more adaptive threshold to detect steady states. We suspect that the interface is capable of serving as the foundation for a responsive gesture based control scheme, unfortunately the experimental results provide no support for this hypothesis.

Aside from shortcomings in our implementation with the VR application, we feel that our contributions during this project have significantly improved the HMI. The additions that we've

made to the interface have improved usability as well as functionality. The availability of IMU data has improved the gesture recognition capabilities of the interface, in that it is able to recognize more gestures than in the previous version. A large part of what we've done during this project has been laying the groundwork for future improvements to be made to the interface. This has been accomplished through the addition of such features as the feature selection grid, which will allow for more signal features to be implemented in later versions of the interface, and the steady state finder, which can be used to feed the classifiers data only when the user has settled into a gesture, and not when transitioning between gestures. This is expected to reduce the amount of misclassifications.

C. Future Works

Dynamic Gesture Recognition Using Hidden Markov Models

Dynamic gesture recognition was a goal for this project that was only implemented at a rudimentary level. The interface would benefit from a more robust dynamic gesture recognition system. Previous studies have adopted Hidden Markov Models (HMM) as a means to recognize dynamic gestures. Englehart implemented a continuous HMM classification system by which a set of six gestures performed by eleven subjects were classified with an accuracy rate of 94.63% [2]. Our proposed approach would calculate classifications only in the event of the user's movement reaching a steady state, rather than continuously. This modification is intended to have only raw gesture data from those steady states fed to the classifier, and not data from transitional states. Typically, EMG signals from transitions are unreliable for accurate gesture recognition, as they are generated in between gestures, and do not lend themselves to the discrete outcome nature of classifiers (a fist or a thumbs up can be recognized more easily than some position between the two). With this modification, we suspect to see improved accuracy over the continuous classification approach, while also lowering the required processing power.

Custom Pipeline

Due to time constraints we were unable to make the pipeline as customizable as we initially desired. The way the current pipeline handles the third byte of the package was written specifically for the use of the virtual reality application and much of its functionality was hardcoded into the HMI and not very flexible in making changes for other client applications. Ideally, inside the GUI the user would have an option in the output tab to set a desired gesture to any of the 8 bits of the value and decide how that bit is set before the package is sent (either continuously set as a certain gesture is being made, upon entering a steady of the current gesture, or upon a tense of the current gesture).

IV. Conclusion

The work completed during this project has significantly benefited the HMI. Usability, functionality, and versatility have been improved over the previous version. Although obstacles could not be overcome when it came time to interface the HMI with our usability assessment platform (the VR application), our contributions have laid the groundwork that will enable more responsive gesture based controls, and other advancements to implemented in the interface.

V. Acknowledgement

This work was supported through the ASPIRES (Accelerated STEM Pathways through Internship, Research, Engagement, and Support) program. A collaboration between Cañada College's Engineering Department, San Francisco State University School of Engineering, and the University of California, Merced. The project was funded by a grant from the U.S. Department of Education through the Minority Science and Engineering Improvement Program (MSEIP), Grant No. P120A150014.

We would like to thank Dr. Amelito Enriquez from Cañada College for the opportunity to participate in this internship and for guiding us through the whole program. Also, we would like to express our appreciation to Dr. Xiaorong Zhang from San Francisco State University, our faculty advisor, and our San Francisco State graduate mentors, Ian Donovan and Kartik Bholla, for all their guidance and advices throughout the whole internship.

Lastly, we would like to acknowledge the 2015 ASPIRES computer engineering interns and Kevin Valenzuela, a student of San Francisco State University, who were previous authors of the HMI code we started with at the beginning of our internship. In addition, we thank Christopher Thomas, another 2016 ASPIRES intern, for the collaboration and guidance between our projects.

References

1. Burges, Christopher JC. "A tutorial on support vector machines for pattern recognition." *Data mining and knowledge discovery* 2.2 (1998): 121-167.

2. Chan, A.d.c., and K.b. Englehart. "Continuous Myoelectric Control for Powered Prostheses Using Hidden Markov Models." *IEEE Transactions on Biomedical Engineering IEEE Trans. Biomed. Eng.* 52.1 (2005): 121-24.

3. Englehart, Kevin. "A Robust, Real-Time Control Scheme for Multifunction Moelectric Control". *IEEE Transactions on Biomedical Engineering*. 50.7 (2003).

4. Hearst, Marti A. "Support Vector Machine." *IEEE Intelligent Systems* (n.d.): n. pag. University of California Berkeley. Web. 11 Aug. 2016. <http://svms.org/tutorials/Hearst-etal1998.pdf>.

5. Hudgins, B., P. Parker, and R.n. Scott. "A New Strategy for Multifunction Myoelectric Control." *IEEE Transactions on Biomedical Engineering IEEE Trans. Biomed. Eng.* 40.1 (1993): 82-94.

6. Linear Classification with Fisher Training Data. Digital image. Discriminant Analysis. The MathWorks, Inc., n.d. Web. 8 Aug. 2016.

7. Mika, S., G. Ratsch, J. Weston, B. Scholkopf, and K.r. Mullers. "Fisher Discriminant Analysis with Kernels." *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)* (n.d.): n. pag. Web. 9 Aug. 2016.

8. Phelan, Ivan, Madelynne Arden, Carol Garcia, and Chris Roast. "Exploring Virtual Reality and Prosthetic Training." *IEEE Virtual Reality (VR)*. IEEE (2015).

9. Phinyomark, Angkoon, Chusak Limsakul, and Pornchai Phukpattaranont. "A Novel Feature Extraction for Robust EMG Pattern Recognition." *Journal of Computing*, (2009).

10. Rabiner, Lawrence, and B. Juang. "An introduction to hidden Markov models." *IEEE ASSP Magazine*. 3.1 (1986): 4-16.

11. Razi, M., Tsvirkunova, L., Chow, J., Reus, R., Donovan, I., Enriquez, A., Pong, W., and Zhang, X. "Engaging Community College Students in Engineering Research through Design and Implementation of a Human-Machine Interface for Gesture Recognition." *Proceedings: 2016 American Society of Engineering Education Conference*, Pomona, CA, April 21-23, 2016.

12. Stapenhurst, Richard. "SVM Demo - File Exchange - MATLAB Central." SVM Demo. Mathworks, 26 July 2010. Web. 04 Aug. 2016. <https://www.mathworks.com/matlabcentral/fileexchange/28302-svm-demo>.

13. Wei, Lai, and Huosheng Hu. "EMG and visual based HMI for hands-free control of an intelligent wheelchair." *Intelligent Control and Automation (WCICA), 2010 8th World Congress*. IEEE (2010).

14. Zhang, Xiaorong, He Huang, and Qing Yang. "Implementing an FPGA System for Real-time Intent Recognition for Prosthetic Legs." *Proceedings of the 49th Annual Design Automation Conference on - DAC '12* (2012): n. pag. Web.

15. Zhang, Xu, Xiang Chen, Yun Li, V. Lantz, Kongqiao Wang, and Jihai Yang. "A Framework for Hand Gesture Recognition Based on Accelerometer and EMG Sensors." *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*. IEEE (2011).