9-1-2014

An Assessment of Single-Channel EMG Sensing for Gestural Input

Travis Peters
Dartmouth College

An Assessment of Single-Channel EMG Sensing for Gestural Input

Dartmouth Computer Science Technical Report TR2015-767

Travis Peters

Abstract

Wearable devices of all kinds are becoming increasingly popular. One problem that plagues wearable devices, however, is how to interact with them. In this paper we construct a prototype electromyography (EMG) sensing device that captures a single channel of EMG sensor data corresponding to user gestures. We also implement a machine learning pipeline to recognize gestural input received via our prototype sensing device. Our goal is to assess the feasibility of using a BITalino EMG sensor to recognize gestural input on a mobile health (mHealth) wearable device known as *Amulet*. We conduct three experiments in which we use the EMG sensor to collect gestural input data from (1) the wrist, (2) the forearm, and (3) the bicep. Our results show that a single channel EMG sensor located near the wrist may be a viable approach to reliably recognizing simple gestures without mistaking them for common daily activities such as drinking from a cup, walking, or talking while moving your arms.

Disclaimer

The information contained in this paper is for informational purposes only, and is based on preliminary work carried out June - September of 2014.

1 Introduction

Wearable devices of all shapes, kinds, and sizes have become increasingly popular. One problem that plagues wearable devices, however, is how interaction with the device is implemented. Due to the wearable nature of these devices, they are often small, meaning they typically have limited space for screens and buttons to handle device input/output (I/O).

In an effort to address limitations in wearable device input, various techniques have been explored such as conventional buttons, voice commands [25], touch sensors that recognize "taps" and swipes" [10], and body gestures [5, 7, 8, 14]. Each method, as would be expected, has advantages and disadvantages. Specifically, we consider these advantages and disadvantages in the context of a mobile health (mHealth) wearable device that runs applications to help wearers track their fitness or manage a health condition.

Using buttons to interact with devices is familiar and fairly straightforward to implement, but as wearables become smaller, buttons become hard to press and take up precious room on an already limited surface. Voice command interactions are fairly simple and have the added benefit of providing "hands-free" input, but may reveal sensitive information about the device wearer, and are therefore less appealing for this type of device. Devices that use touch sensors to accept input make better use of the limited space on wearable devices for input and enable a less revealing method for interacting with an mHealth wearable device, however, due to the (typically) small size of such devices, screens are also generally small which means that a large portion of the screen becomes occluded when the user actually touches the screen – this can lead to annoying and error prone interactions. Body gestures have similar disadvantages, but, depending on the means of sensing, can detect subtle gestures that dont reveal sensitive information and these gestures can be used to improve the usability of wearable devices. For example, if a wearer could squeeze their fist once or twice to provide a sort of binary input to a device (e.g., yes/no, on/off, forward/backward), that would be both subtle and usable. One such sensor that offers these benefits is known as an electromyogram (EMG) sensor which measures the electrical activity in muscles; more details about EMG will be discussed in subsection 2.2.

In this work, we investigate the suitability of a single-channel EMG sensor from a new biosignals tool kit as the primary source of gestural input into a device. Although gestural recognition including EMG sensors has been explored previously [28], it is often coupled with other sensors such as accelerometers which actually provide the gestural data, while the EMG sensor data is used for "signal segmentation" (i.e., identifying and extracting meaningful parts of the input data). We found resources that describe experiments that can be done with EMG sensors and discuss how to prepare and place sensors about the body, but to our knowledge there exists no resource which provides a clear guide as to where an EMG sensor should be placed on the body and why it should be placed in that location.

Our work uses a prototype EMG sensor device built with components from the BITalino Biosignals toolkit [4] to measure muscle activation at several locations on the arm and perform gesture recognition. There is, however, no shortage of challenges in this work.

1.1 Challenges

Sensor Placement. One of the primary questions we seek to address is where to place an EMG sensor to maximize its utility. In practice, these sensors are typically placed along the arm on major muscles, but we can find no definitive guide as to why this choice was made. There is also the related challenge that once an optimal location is identified for the sensor, there are likely going to be slight errors in how the sensor is placed on the body each time since it must be precisely placed on the skin again any time the sensor is removed.

Gesture Recognition. Recognition of a specific gesture in this context can be particularly challenging due to the nature of the data that the sensor collects. Since the objective is to recognize gestures from EMG sensor data, we rely entirely on the electrical signals that fire upon muscle activation; determining the difference between an intended gesture and something as common as waving your arms while speaking or picking up an object is certainly non-trivial. This problem is made even more

challenging due to the fact that the EMG sensor we have chosen to use provides only a single channel of EMG sensor data, while most systems that leverage EMG sensor data rely on two or more channels of EMG sensor data.

1.2 Contributions

Our ultimate goal is to use an EMG sensor to enable gestural input into an Amulet device [26] and possibly other applications or devices that take gestural input to interact with them or realize something health related about the wearer of the sensor. We believe this work will help us better understand the utility of using the BITalino toolkits EMG sensor in the broader context of applications and/or devices that interact with, or run on, an Amulet.

In this paper, where we work toward the aforementioned goal, we propose the following contributions:

- We constructed a prototype EMG sensor device from the BITalino Biosignals toolkit. The current design is simple and can easily be made wearable in the next iteration.
- We implemented a proof-of-concept application that connects to the EMG sensor and collects sensor data to perform gesture recognition.
- We implemented a training harness for obtaining training data from our EMG sensor device and use an off-the-shelf machine learning module to train a model to recognize three classes of gestures from EMG sensor data.
- We explored various placements of the EMG sensor around the arm and show our findings as to which locations work best in terms of accurately recognizing gestures.

2 Background

In this section, we provide background information for the mHealth wearable platform that motivated this work, electromyography, and an experimental biosignals toolkit that we have chosen as the primary subject of this work.

2.1 Amulet

An *Amulet* [26] is a small mobile health (mHealth) wearable device that is currently being developed by researchers at Dartmouth College and Clemson University (Figure 1). The Amulet relates to this work in that it is the desired platform where our gestural recognition code will run in the future.

The Amulet is currently being designed to be (1) highly available, (2) secure, and (3) able to run multiple, third-party health applications. Security is outside of the scope of this paper, but in later iterations of this prototype intended to run on the Amulet, security concerns will need to be addressed. The high availability of the Amulet is also outside of the scope of this paper; various power saving techniques are implemented by Amulet and one of the implications of that is that any codes that will run on the Amulet must be implemented as finite state machines. Our prototype is not yet compatible with this requirement, thus the high availability of Amulet is not be discussed in detail in this paper. The ability to run multiple applications is also not entirely relevant except for the fact that this work is a result of exploring one such application for the Amulet.

While the Amulet itself is not a large component of this particular work, we seek to answer the question of whether or not the wrist would be a suitable location for an EMG sensor. If it is not, we would like to identify other locations where such an EMG sensor might be located to provide EMG gesture input to an Amulet. The Amulet, thus, should remain in the overall picture as we assess the suitability of the EMG sensor so that the question of "suitable for what?" can be addressed.

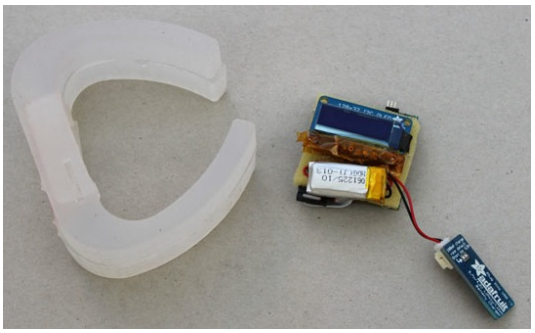


Figure 1: An early prototype of a wearable Amulet [1].

2.2 Electromyography (EMG)

As stated previously, electromyography (EMG) "is an experimental technique concerned with the development, recording, and analysis of myoelectric signals. Myoelectric signals are formed by physiological variations in the state of muscle fiber membranes" [13]. Specifically, in this paper we experiment with a *surface EMG* (sEMG) sensor from the BITalino Biosignals Toolkit (described in subsection 2.3), which is a non-invasive approach to measuring muscle electrical activity. An example of EMG sensor data is shown in Figure 2.

Surface electromyography has, in the past, been used in various applications such as physical rehabilitation (e.g., physical therapy), biomechanics (e.g., sports training and motion analysis), and ergonomics (e.g., workplace studies, product design) [3]. More recently, sEMG has also been explored as a viable mechanism for control, whether it be for mobile device control [6], or motor device control such as electric-powered wheelchairs [15].

Working with a surface EMG sensor essentially involves manually placing electrodes on the surface of the skin in the desired location. There are a few ways in which an EMG signal can be measured, but in the context of this paper we consider an EMG sensor that uses a bipolar differential front end for improved signal to noise ratio. This means that, while we use a three lead accessory to obtain EMG sensor data, two leads correspond to the common positive and negative voltage and the third lead is a reference lead. Measurements from each of the negative/positive poles are subtracted in the operational amplifiers, thus providing a 1-D time series [23].

While this approach is non-invasive, the requirement of placing sensors on the exterior of the body is one of the more prominent issues with sEMG experiments due to the fact that sensor placement is a manual, and thus error prone, setup action; this is also one of the primary issues addressed in this work.

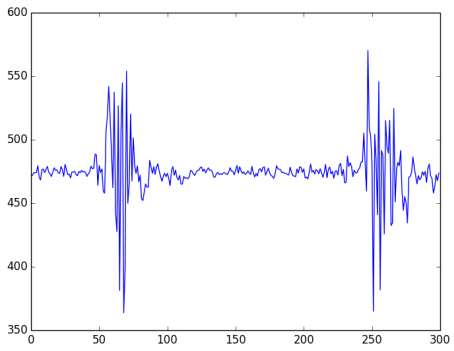


Figure 2: An example of what EMG signals look like – the more active parts of the signal correspond to muscular activity that the EMG sensor detected.

2.3 BITalino Biosignals Toolkit

The BITalino Biosignals toolkit is a fairly new, experimental platform that allows hobbyists and researchers alike the ability to construct low-cost prototypes for measuring various body signals. The toolkit comes in various "flavors," but in this work weve chosen the BITalino freestyle kit (Figure 3) which consists of various "blocks" that each perform their own function. Specifically, our kit provided us with an ATmega328P microcontroller, power management block, BC417 bluetooth module with Class II Bluetooth v2.0 for wireless data transfer, various sensor blocks (i.e., electromyography sensor, triaxial MEMS accelerometer sensor, electrodermal activity sensor, light sensor, electrocardiogram sensor), as well as an LED, pre-gelled electrodes with two- and three-lead accessories, and a 320 mAh Li-Po battery [4].

Specific details about the components we used in the prototype EMG device will be covered in section 3.

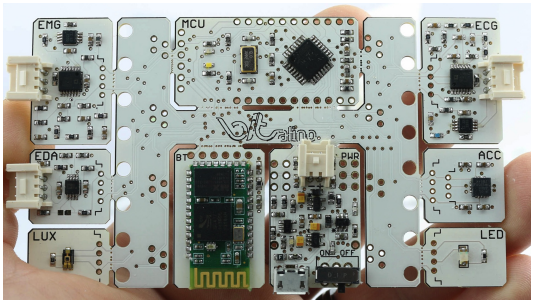


Figure 3: BITalino biosignals toolkit [4].

3 Implementation

In this section we describe the status of our hardware and software prototypes that we built to realize a system capable of recognizing gestures.

3.1 Hardware Prototype

Our current hardware implementation is fairly straightforward and not very different from other EMG controllers in use today. The prototype device, depicted in Figure 4, consists of a 320mAh battery that powers the ATmega328P microcontroller, the EMG sensor, and the Class II Bluetooth v2.0 module. In the future we envision using a more customizable microprocessor or modifying the BITalino firmware so that EMG sensor data processing can be implemented on the EMG device itself. For experimental purposes, however, we paired our device to a MacBook Pro and stream the sensor data to the laptop (via Bluetooth) where the processing is implemented.

A three-lead accessory with pre-gelled electrodes runs from the EMG sensor; the three electrodes are placed on the surface of the skin in the muscle region of interest.

3.2 Software Prototype

On the software side, we have implemented two primary components: (1) a live EMG streaming script in Python that leverages the Python API for BITalino device interaction, and (2) the entire pipeline for gesture recognition on EMG signals.

The live EMG streaming script is fairly simple and handles connecting to the device, performing some device configuration such as setting the sampling rate, then continuously obtains data until the script is interrupted. Currently, the script does not fit into the gesture recognition pipeline that is explained below. We verified that we can grab windows of data from the EMG sensor data in real-time

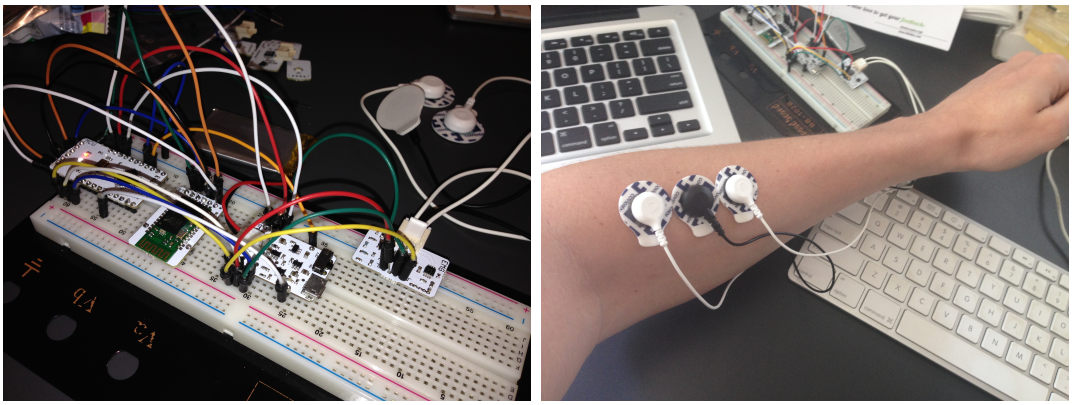


Figure 4: (Left) Our prototype of the EMG sensor device. (Right) Three pre-gelled electrodes placed on the brachioradialis muscle (one of the more common sites for placing EMG sensors) connected to the EMG sensor on our prototype.

and threshold on the values to determine if there is muscle activation at that moment, however, it does not recognize gestures per sa.

To approach the problem of recognizing gestures, we implement a gesture recognition pipeline – see Figure 5. This pipeline first aggregates data we have collected through the OpenSignals [18] application. We chose to use OpenSignals for data collection in this phase because it is a live signal viewing application that gives meaningful feedback about the current EMG activity and it writes the session data to a structured text file on the file system to can easily be parsed and loaded. With the data loaded, the next step in the pipeline is to determine windows of interest (i.e., slices of the EMG signal that correspond to muscular activity). Next, the pipeline extracts features from the aforementioned windows and uses those features to describe various muscle activity (i.e., gestures). Those features are then used to train a model for the gestures that are input. In fact, we create multiple models by exposing multiple classifiers in an attempt to identify the best classifier for this data. Since our model training and classification is currently performed offline (i.e., we use a static dataset that is loaded initially), we decompose our dataset into a training set and a testing set; the data from the training set is used to train a classifier, and we measure the accuracy of that classifier by comparing the predictions of the classifier over the test set with the ground truth.

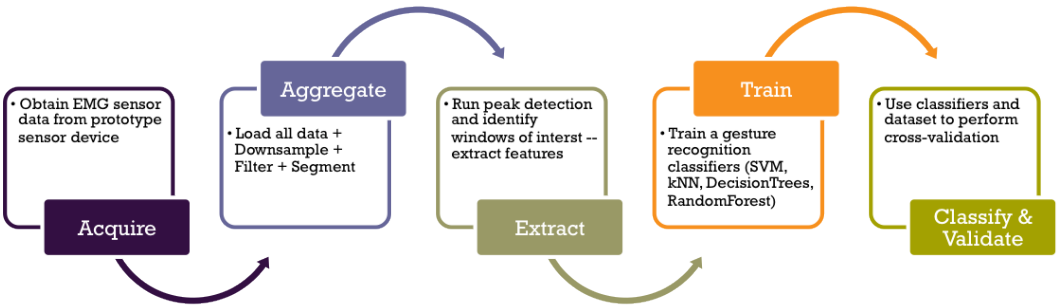


Figure 5: An overview of our gesture recognition pipeline.

Before we were able to determine windows of interest in the data collected from OpenSignals, we performed several operations on the data after loading it. In all of our work we configured the BITalino to sample the EMG sensor at 1000 Hz. We did that to ensure that the most detail possible would be available to us in the earlier stages of our analysis and system development. Doing so, however, also introduced a lot of noise into our data. To address this, we:

- First downsample the data with an order 8 Chebyshev type 1 filter, as this is one of the most

common classes of low-pass filters and the higher order yields better filtering performance.

- Next, since we collected our data as a single session of periodically repeated gestures, we segment the session roughly into segments of 1-2 seconds to obtain multiple samples of a given gesture.
- Thirdly, we perform peak detection on each of the segments applying a continuous wavelet transform to the signal for a set of specific window widths. We chose this peak detection approach because it selects only the relative maxima which appear frequently in the specified windows with a high enough signal-to-noise ratio.
- Finally, once we have identified peaks, we consider a window around that peak as a window of interest for feature extraction.

In our software prototype, we leverage an existing Python machine learning module, scikit-learn [21], for training gestural models and classifying sensor data as well as other Python modules for tasks such as signal processing [12] and graphing [11].

4 Evaluation

In this section we describe our experiments and evaluation of our work presented in this paper. In the description of our experiments and evaluation that follow, we use the hardware and software prototypes as discussed in section 3.

4.1 Experimental Setup

In this phase of our work, one right-hand dominant, male subject, age 24, participated in this initial study. The subject has no known history of neuromuscular or joint diseases and was aware of risks that can be involved when working with the experimental prototype device.

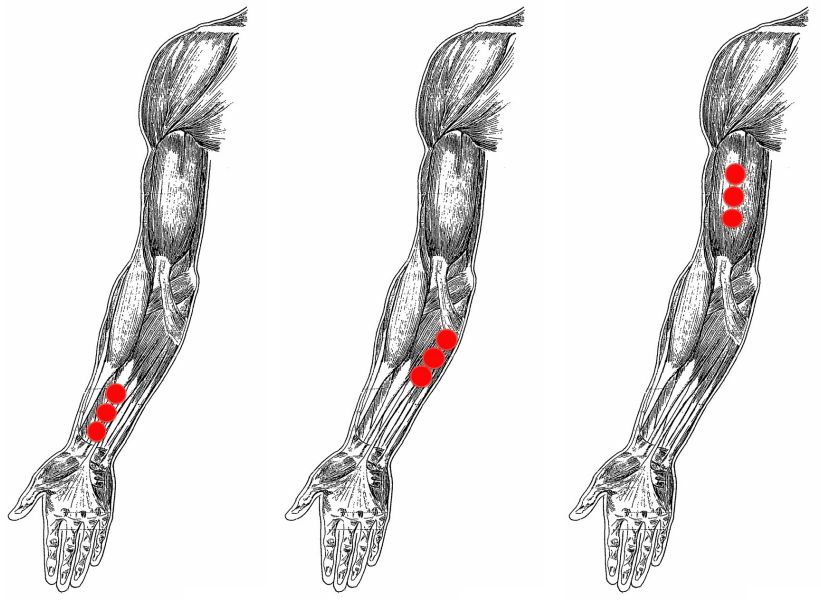
Recall that it is our goal to recognize simple and subtle gestures made by the user that could relate to a sort of binary input to a device (e.g., yes/no, on/off, forward/backward). Another goal is to reliably recognize these gestures among all sorts of activities.

For our experiments, we deployed our EMG sensor at three distinct locations along the arm, as shown in Figure 6. At each identified location, the bicep (i.e., biceps brachii), forearm i.e., brachioradialis), and center of the inside part of the wrist (i.e., flexor digitorum superficialis), all three pre-gelled electrodes were placed contiguously, lengthwise along the muscle, on the surface of the skin. The dominant arm of the subject was used when collecting EMG sensor data.

In our experiment, the subject performed 5 gestures in a way that felt natural and repeated each gesture 10 times in a single session. The gestures were recorded in a continuous stream so the subject performed each gesture every 2-3 seconds, resulting in a session lasting approximately 20-30 seconds. Thus, we collected 50 gesture samples in each of the three sensor placement locations, resulting in 150 gesture samples in all. The samples taken at each of the different sensor placement locations are considered as three separate experiments due to the fact that the sensors ability to measure muscular activity is highly dependent on the location of the sensor.

For our experiments, our only instructions to the subject were to repeat the gesture for a particular session every 2-3 seconds but not to rush. Our goal was to capture, as best as possible, a realistic sample of electromyographic activity when performing these gestures. The gestures that the participant performed were as follows:

1. Single Fist Squeeze
2. Double (Consecutive) Fist Squeezes
3. Drink
4. Talk
5. Walk



(a) Experiment I (wrist) (b) Experiment II (forearm) (c) Experiment III (bicep)

Figure 6: Here we display the deployment of the three electrodes for the EMG sensor in each of our three experiments. As shown, we chose three locations (a-c) where we placed the three electrodes contiguously along the arm.

4.2 Experimental Results

Each of the 5 gestures was performed 10 times at each sensor placement location identified in Figure 6. When training our gesture recognition models, we considered three gesture classes: (1) single squeeze gesture, (2) double squeeze gesture, and (3) other. The other class consisted of a combination of the drinking, talking, and walking gestures to were meant to represent the class of all things that may be falsely recognized as an intended gestural input to the device.

We considered the performance of each of our classifiers separately at each sensor placement location. For each sensor placement location, we considered four different classifiers that were trained and tested on the same data. The classifiers we used to recognize gestures are:

1. A polynomial support vector machine (SVM) classifier,
2. A k-Nearest Neighbors classifier (k=3) using euclidean distances,
3. A Decision Tree classifier using Gini Impurity to measure information gain, and
4. A Random Forest which used 100 estimators (random trees) and Gini Impurity to measure information gain.

Figure 7 shows the results of each classifier attempting to classify EMG sensor data samples at the wrist, forearm, and bicep, respectively. We measure the success of each classifier in terms of their precision and recall. Precision is essentially the the ability of a classifier not to label as positive, a sample that is negative. Precision can thus be expressed as:

$$precision = \frac{TruePositive}{(TruePositive + FalsePositive)}$$

Recall, on the other hand, can be thought of as the ability of a classifier to find all of the positive samples. Recall can be expressed as:

$$recall = \frac{TruePositive}{(TruePositive + FalseNegative)}$$

From the three experiments we conducted, we found that in the worst case our kNN classifier would exhibit less than 40% precision and recall on sensor data from the bicep, which is far from acceptable for a reliable gesture recognition system. Overall, the kNN classifier was one of the lower performing classifiers, especially when you consider that regardless of where the sensor was located in our three experiments, the ability of our kNN classifier to reliably find all correct samples was less than 70%. The SVM and Random Forest classifiers were more consistent in their abilities to recognize gestures, and performed about as well as one another, whereas the Decision Tree classifier seemed to fair the best overall.

Due to the lower performance of each of the classifiers, with the exception of the SVM classifier, when trying to recognize gestures from sensor data collected at the bicep, we believe that this location may not be ideal for recognizing the sorts of gestures we are interested in capturing. In fact, neither the bicep nor the forearm are ideal for sensor placement in the context of the Amulet device proposed in subsection 2.1 since these locations are physically far from the Amulet; the most desireable location

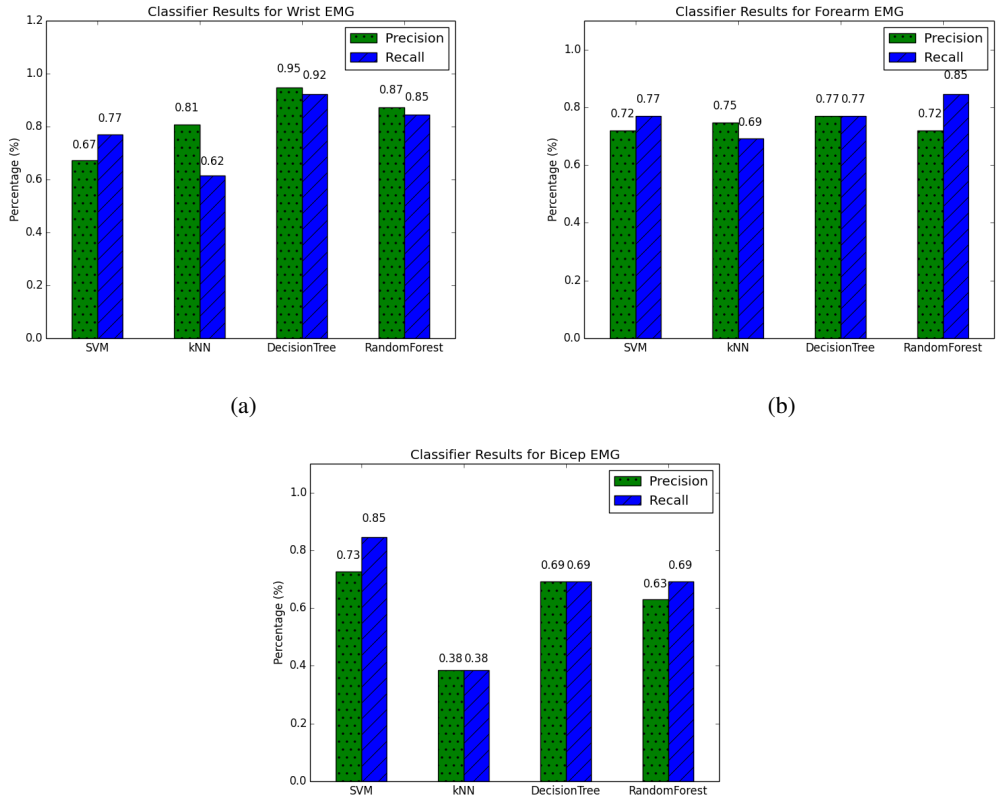


Figure 7: Precision and recall for each of the classifiers we tried. Each graph depicts the precision and recall of the classifiers on sensor data collected from the EMG sensor while placed at the wrist (a), forearm (b), and bicep (c).

for an EMG sensor that could enable gestural input to an Amulet would be at the wrist. In the best case, according to these experiments, the best placement for the EMG sensor would be the wrist, and the ideal classifier to use would be a Decision Tree that uses Gini Impurity to effectively isolate the features that most clearly differentiate the different gestures. This classifier achieved better than 90% precision and recall on our sensor data collected at the wrist which would be acceptable for a reliable gesture recogniztion system. This is also promising for future work since a Decision Tree classifier can be trained offline, has a small memory footprint when storing the classifier, and is computationally efficient [9].

5 Discussion & Limitations

According to the data we collected, the wrist and forearm were consistently more reliable sensor locations for performing gesture recognition on our 5 gestures. Even though the bicep is generally a good location for obtaining strong EMG sensor data, the gestures we were interested in recognizing dont seem to engage the bicep muscles as much as they do the forearm and wrist muscles. This fact likely results in more ambiguity during training, and thus, lower performing classifiers.

In many ways I am encouraged by the results we see. For instance, the fact that our Decision Tree classifier and Random Forest classifier achieve 95% and 87% precision, respectively, suggests that a single-channel EMG sensor located on the wrist could in fact perform fairly reliable gesture recognition on a set of simple gestures such as ours. The precision and recall percentages achieved on the forearm, however, are not as encouraging as I would have hoped given that there is such as wealth of muscular activity happening in that area. It may be though, that there is in fact a higher noise-to-signal ratio, which makes the classifiers less effective on data collected from that sensor location.

Unfortunately, these results are in fact faced with many limitations. The performance of our gesture recognition classifiers are highly dependent on the placement of the sensor. Additionally, due to the limited context of only having a single channel of EMG sensor data to analyze, too much noise in the sensor data can cripple our system's ability to generate a meaningful classifier. Another possible limitation is our dependence on having a fairly large amount of sensor data to analyze for gesture input. We currently require roughly 2 seconds of data to extract features from, so that we can determine if muscle activity corresponds to a specific gesture – this amount of processing obviously does not enable real-time gesture recognition. Finally, we need to further consider the memory storage and computational requirements of the classifiers we have sampled to determine if they are feasible to run on a resource-constrained device.

6 Related Work

In this section, we describe related methods of obtaining gestural input data such as gesture recognition using computer vision techniques, sensors that measure inertial motion with accelerometers, and sensors that measure electromyography, as well as techniques that combine various sensors.

Computer Vision-Based Gesture Recognition. This approach to gesture recognition relies on a camera that is capable of observing a user that performs various gestures in order to interact with a system or device. In order to realize camera-based gesture recognition, the system must first detect a person, isolate their hand(s), and then track their motion to determine the users intended input to the system. Chen et. al. [5] implemented real-time tracking algorithms to detect and track a hand, analyzed the motion of the hand using a Fourier descriptor (FD), and used Hidden Markov Models (HMMs) to train a model and recognize gestures. More recently, Ren et. al. [22] implored similar methods and leveraged the color video camera and depth sensor on a Kinect sensor, which is a commercialized Microsoft camera product that works with the Xbox 360 gaming console, to perform gesture recognition enabling users to do arithmetic computations and even play rock-paper-scissors.

These methods use common machine learning algorithms to recognize gestures and enable users to interact with a system in a natural way, however, there are many disadvantages to this approach.

The system depends on an external camera, which assumes the user must remain in sight of the camera and gesture in a specific direction (i.e., toward the camera). Another notable disadvantage is the performance of these systems. In ideal conditions these systems perform well, however, amongst complex backgrounds, occlusions to the camera, and poor lighting, these methods suffer. To solve the problem of accurately detecting a hand in a camera image, some researchers proposed the use of a specialized glove that can be worn on the hand [27]. This introduces more components, and thus more complexity, to the gesture recognition system which is not ideal.

Wearable Sensor-Based Gesture Recognition. This approach to gesture recognition relies on sensors that are either contained within a device which a user holds in their hand or wears on their wrist/arm, or sensors that can easily be placed upon the body. Gesture recognition via wearable sensing has become widely explored and used due to the proliferation of consumer electronics that contain sensors such as accelerometers, gyroscopes, and so forth.

In the commercial product world, we are seeing an explosion of capable wearable devices. Fitbit [8] is a simple health-tracking device that uses a three-axis accelerometer to detect user activity. The Pebble smartwatch [20] was one of the first wearable smartwatches to become open to third-party developers and hosts a slough of sensors such as three-axis accelerometer, magnetometer, and light sensor, enabling developers to create applications that sense all sorts of user activity and interactions. More recently, the newly released Apple Watch [2] was released and is equipped with multi-touch and force touch sensors which allow it to recognize user input in the form of touches and swipes, and has the ability to determine between touches with varying pressure. Apple watch is also equipped with an accelerometer, gyroscope, barometer, and heart rate sensor which gives it a wealth of sensing possibilities. Other recent smartwatches such as Moto 360 [16] and Gear S [24] are equally capable with many of the same built-in sensing capabilities. These commercial products respond only to touch and swipe gestures "out of the box," but as these platforms are becoming available to developers, they are equipped with most of the necessary sensors to enable gesture recognition for third-party applications. The Myo device [17] has not been officially released yet but is the only wearable device that we are aware of that claims to use a combination of sensors like accelerometer and EMG to perform gesture recognition.

In academics, Dong et. al. [7] proposed Bite Counter, a wrist-worn device with an internal gyroscope to detect bites taken by the wearer in a given eating session. Liu et. al. [14] proposed uWave which uses a single three-axis accelerometer in a Wii remote and dynamic time warping (DTW) to perform efficient gesture recognition. Costanza et. al. [6] proposed the idea of "intimate interfaces," an EMG-based solution for interacting with a mobile device in a subtle way. By placing a single wearable EMG armband around the bicep, 4 "motionless gestures" were recognized with reasonable accuracy, however, this work did not try to recognize gestures so much as it tried to simply identify when a threshold had been exceeded and for how long it had been exceeded; this work also did not consider more complex issues such as recognizing their desired gestures amongst everyday activities such as lifting objects or regular arm movement during a conversation. Moon et. al. [15] proposed a wearable EMG device to control an electric-powered wheelchair for users that had suffered spinal injuries. Two surface EMG sensors were placed on the shoulders of the user and control commands were determined by "combinations of left-, right-, and both-should elevation gestures" [15]. Ultimately, this work demonstrated that a two-channel EMG sensor could be used to successfully recognize four shoulder gestures. Zhang et. al. [28] even proposed a hybrid of the accelerometer-based approach and the EMG-based approach by combining the two sensors and then comparing the performance of gesture recognition for accelerometer sensor only vs. multi-channel EMG sensor only vs. a combination of the accelerometer and multi-channel EMG sensor. Their results, observed from 5 subjects performing 18 specific gestures, show that nearly 100% accuracy can be realized when combining the accelerometer and EMG sensors, while only 85.5% - 90.7% accuracy could be achieved using only the accelerometer sensor and 65.9% - 80.3% accuracy could be achieved using only the EMG sensor.

Gesture recognition via wearable sensors can be quite accurate, especially when combining multiple sensors to obtain more contextual information about the wearer and their interactions. A minimal, yet highly accurate, configuration of sensors remains a topic which we would like to

continue to explore in the future.

7 Conclusion & Future Work

In this paper, we propose the use of our prototype EMG sensor device to perform gesture recognition. We also propose a gestural recognition pipeline to explore the performance of a SVM classifier, a kNN classifier, a Decision Tree classifier, and a Random Forest classifier on a signle channel of EMG sensor data recorded for 5 gestures in 3 different sensor placement locations along the dominant arm. The evaluation of our sensor placement, data collection, and gestural recognition pipeline suggests that a Decision Tree classifier, which achieves 95% precision and 92% recall, may in fact be a suitable classifier for recognizing EMG sensor data recorded at the wrist.

This work can be used to continue to work toward a minimalist external sensing mechanism to enable reliable gestural input to I/O constrained devices. Our future work may involve exploring the utility of including additional EMG sensors in our EMG sensing device for gestural input. We may also explore the technique of correlating our EMG sensor data with other sensors such as accelerometers and/or gyroscopes to improve accuracy and grow the size of our gesture alphabet. We plan to continue to work toward bringing our prototype to a wearable form factor and implementing the recognition pipeline in Amulet-compatible C – the real challenge may in fact be making our implementation a real-time control system such that all of the signal pre-processing, feature extraction, and classification can be done within 300ms [19].

Acknowledgements

This research results from a research program at the Institute for Security, Technology, and Society, supported by the National Science Foundation under award number CNS-1314281. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

I thank Professor David Kotz for guidance and feedback while determining the direction and goals for this work, and Hugo Silva for numerous clarifications and tips in working with the BITalino biosignals toolkit. I also thank the other members of the mHealth Security and Privacy lab at Dartmouth College (Shrirang Mare, Aarathi Prasad, Tim Pierson, Tianlong Yun, Eric Chen, Emily Greene, Xiaohui Liang, Ron Peterson, and Andrés Molina-Markham) for their encouragement, support, and willingness to listen and provide feedback throughout the development of this work.

References

[1] Amulet wearable prototype. Online at http://circuitcellar.com/.

[2] Apple Watch. Online at http://www.apple.com/watch, visited September 2014.

[3] Basics of surface electromyography applied to psychophysiology, October 2008.

[4] BITalino DiY Biosignalset. Online at http://bitalino.com/, visited June 2014.

[5] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745 – 758, 2003. DOI http://dx.doi.org/10.1016/S0262-8856(03)00070-2.

[6] E. Costanza, S. A. Inverso, and R. Allen. Toward subtle intimate interfaces for mobile devices using an emg controller. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 481–489, New York, NY, USA, 2005. ACM. DOI 10.1145/1054972.1055039.

[7] Y. Dong, A. Hoover, J. Scisco, and E. Muth. A new method for measuring meal intake in humans via automated wrist motion tracking. *Applied Psychophysiology and Biofeedback*, 37(3):205–215, 2012. DOI 10.1007/s10484-012-9194-1.

[8] Fitbit. Online at https://www.fitbit.com, visited June 2014.

[9] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *ICML*, volume 99, pages 124–133, 1999.

[10] S. Hodges, S. Izadi, A. Butler, A. Rrustemi, and B. Buxton. Thinsight: Versatile multi-touch sensing for thin form-factor displays. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 259–268, New York, NY, USA, 2007. ACM. DOI 10.1145/1294211.1294258.

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[12] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2014-08], Online at http://www.scipy.org/.

[13] P. Konrad. The abc of emg. *A practical introduction to kinesiological electromyography*, 1, 2005.

[14] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657 – 675, 2009. PerCom 2009, DOI http://dx.doi.org/10.1016/j.pmcj.2009.07.007.

[15] I. Moon, M. Lee, J. Chu, and M. Mun. Wearable emg-based hci for electric-powered wheelchair users with motor disabilities. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2649–2654, April 2005. DOI 10.1109/ROBOT.2005.1570513.

[16] Moto 360. Online at https://moto360.motorola.com/, visited September 2014.

[17] Myo Gesture Control Armband. Online at https://www.thalmic.com/en/myo/, visited July 2014.

[18] OpenSignals. Online at https://github.com/bitalino/OpenSignals, visited July 2014.

[19] M. A. Oskoei and H. Hu. Myoelectric control systemsa survey. *Biomedical Signal Processing and Control*, 2(4):275 – 294, 2007. DOI http://dx.doi.org/10.1016/j.bspc.2007.07.009.

[20] Pebble Smartwatch. Online at https://getpebble.com/, visited June 2014.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Z. Ren, J. Meng, J. Yuan, and Z. Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 759–760, New York, NY, USA, 2011. ACM. DOI 10.1145/2072298.2072443.

[23] H. Salva. Personal communication, July 1999.

[24] Samsung Gear S. Online at http://www.samsung.com/global/microsite/gears/, visited September 2014.

[25] D. Schnelle. Mobile speech recognition, 2008. DOI doi:10.4018/978-1-60566-054-7.ch254.

[26] J. Sorber, M. Shin, R. Peterson, C. Cornelius, S. Mare, A. Prasad, Z. Marois, E. Smithayer, and D. Kotz. An amulet for trustworthy wearable mhealth. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 7:1–7:6, New York, NY, USA, 2012. ACM. DOI 10.1145/2162081.2162092.

[27] Y. Yin and R. Davis. Toward natural interaction in the real world: Real-time gesture recognition. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, ICMI-MLMI '10, pages 15:1–15:8, New York, NY, USA, 2010. ACM. DOI 10.1145/1891903.1891924.

[28] X. Zhang, X. Chen, W.-h. Wang, J.-h. Yang, V. Lantz, and K.-q. Wang. Hand gesture recognition and virtual game control based on 3d accelerometer and emg sensors. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 401–406, New York, NY, USA, 2009. ACM. DOI 10.1145/1502650.1502708.