2016

Real-time translation of American Sign Language using wearable technology

Jackson Taylor
University of Richmond

Real-Time Translation of American Sign Language using Wearable Technology

Jackson Taylor

Honors Thesis*

Department of Mathematics & Computer Science

University of Richmond

*Under the direction of Dr. Barry G. Lawson

The signatures below, by the thesis advisor, the departmental reader, and the honors coordinator for computer science, certify that this thesis, prepared by Jackson Taylor, has been approved, as to style and content.

(Dr. Barry Lawson, thesis advisor)

(Dr. Doug Szajda, departmental reader)

(Dr. Lewis Barnett, honors coordinator)

Abstract

The goal of this work is to implement a real-time system using wearable technology for translating American Sign Language (ASL) gestures into audible form. This system could be used to facilitate conversations between individuals who do and do not communicate using ASL. We use as our source of input the Myo armband, an affordable commercially-available wearable technology equipped with on-board accelerometer, gyroscope, and electromyography sensors. We investigate the performance of two different classification algorithms in this context: linear discriminant analysis and k-Nearest Neighbors (k-NN) using various distance metrics. Using the k-NN classifier and windowed dynamic time warping as the distance metric, our working prototype is able to obtain accuracies between $94\% - 98\%$ when classifying up to 20 different ASL gestures.

Contents

1 Introduction **1**

2 American Sign Language **3**

3 Background and Related Work **4**

4 The Myo Device **6**

4.1 Accelerometer 7

4.2 Gyroscope 7

4.3 Roll, Pitch, Yaw 8

4.4 Electromyography 8

4.5 Construction of Complete Data Signal 9

5 Classification Algorithms **9**

5.1 Linear Discriminant Analysis 10

5.2 k-Nearest Neighbors (k-NN) 13

5.2.1 Euclidean Distance 13

5.2.2 Manhattan Distance 14

5.2.3 Dynamic Time Warping 14

5.2.4 Dynamic Time Warping with Windowing 16

6 Implementation of the Prototype **18**

6.1 Capturing an ASL Gesture 18

6.1.1 Static Windowing Capture 18

6.1.2 Dynamic Windowing Capture 20

6.2 Signal Sampling 24

6.3 Classification Process 25

6.3.1 Classification using LDA 26

6.3.2 Classification using k-NN 26

6.4 Swift Programming Language 27

7 Experiments and Results **28**

7.1 Comparison of Classification Algorithms 28

7.2 Comparison of Classification Algorithms: Synthetic Data 30

7.3 Increasing the Vocabulary Size 32

7.4 DTW Window Size 33

8 Future Work **35**

9 Conclusion **35**

1 Introduction

American Sign Language (ASL) is one of the primary forms of communication used by those with hearing loss. The most recent estimates of ASL's user base place the number of users around one-half to two million people in the United States [12]. Still, this number is small compared to the current three hundred million United States citizens, indicating a large proportion of citizens who cannot communicate effectively via ASL. Some people may take it upon themselves to learn American Sign Language, but the process can be time-consuming, equivalent to learning another language entirely. Providing a mechanism to automate translation of ASL into audible form can help remove communicative barriers between those who do and do not regularly use ASL.

In general, ASL is broken down into about 6000 words, letters, and phrases [17]. For example, the most commonly used ASL words include gestures for home, mom, pizza, and dog. Before ASL and other forms of sign language were widely adopted, the process of spelling out words was commonly used [16], but this process quickly became cumbersome. The introduction of word- and phrase-level abstractions, in ASL and similar standards, facilitated a fast-paced, efficient communication similar to spoken languages.

The goal of the work here is to identify via classification select word-level ASL abstractions. The problem is similar to speech recognition in many ways, but presents unique challenges. Two possible approaches for speech recognition are analysis of the corresponding signal as a time series or analysis of features extracted from the signal. As an example of the latter, features can be extracted in the form of phonemes, units of sound used to differentiate one word from another [21]. For example, the word *pit* is differentiable from *bit* with the result of exchanging the phoneme /'b/ for /'p/. A word such as *bit*, when pronounced, registers the phonemes /'b/, /I/, and /t/ in that sequential order. Although sign language can be modeled using as phonemes the orientation and position of the arm as well as the shape of the hand, the recognition process is not as streamlined as for speech. In a single gesture, arm orientation and hand shape, i.e., sign language phonemes, can change simultaneously and sequentially, complicating the analysis of those features. In this work,

we focus our analysis on signals as time series rather than on phonemes of ASL.

To date, the most successful approaches in automated translation of ASL involve analyzing video of hand gestures. While this technique is very accurate in translating sign language, video analysis does not facilitate the most common forms of communication seen in today's world. That is, the associated requirement of having a camera present is constraining at best, prohibitive at worst. Video analysis is a very effective approach when communicating with someone, say, over a video conferencing system, but the motivation of the work presented here is to develop a much less constrained mechanism for translating ASL.

In this work, we propose use of a wearable device called the Myo, a small band worn on the forearm, which can measure a change in orientation, position, and movement of the arm, hand, and fingers. The Myo is very sleek and relatively unobtrusive, facilitating automated translation of ASL with only minor hardware requirements. The ultimate goal is to pair the Myo with a mobile device (e.g., smart phone) executing software that will translate the Myo signals into a corresponding ASL word which can then be reproduced audibly.

During movement of the user's arm and/or hand, the Myo device transmits 15 different streams of data to the mobile device. Machine learning techniques are used to classify these data into ASL words, which can then be emitted in audible form by the mobile device. Moreover, the hardware required is extremely portable and relatively low cost. This approach, pairing a small wearable device with a mobile device easily carried by the user, shows promise for effective communication between persons who do and do not understand ASL.

The remainder of this paper is organized as follows. In Section 2, we provide a brief overview of American sign language. In Section 3, we discuss background and related work. In Section 4, we discuss the Myo device and the signals it produces. In Section 5, we discuss the machine learning algorithms we use to classify signals into ASL words. In Section 6, we discuss details of the prototype implementation, and in Section 7 we present results from experiments using our approach. Finally, Section 8 suggests future directions.

2 American Sign Language

American sign language (ASL) is a form of nonverbal communication used by millions of Americans [12]. ASL contains hand shape gestures that correspond to individual letters as well as hand and arm gestures that represent words or full phrases. Figure 1 shows all letters and digits in the ASL finger-spelled dictionary and their corresponding hand gesture.

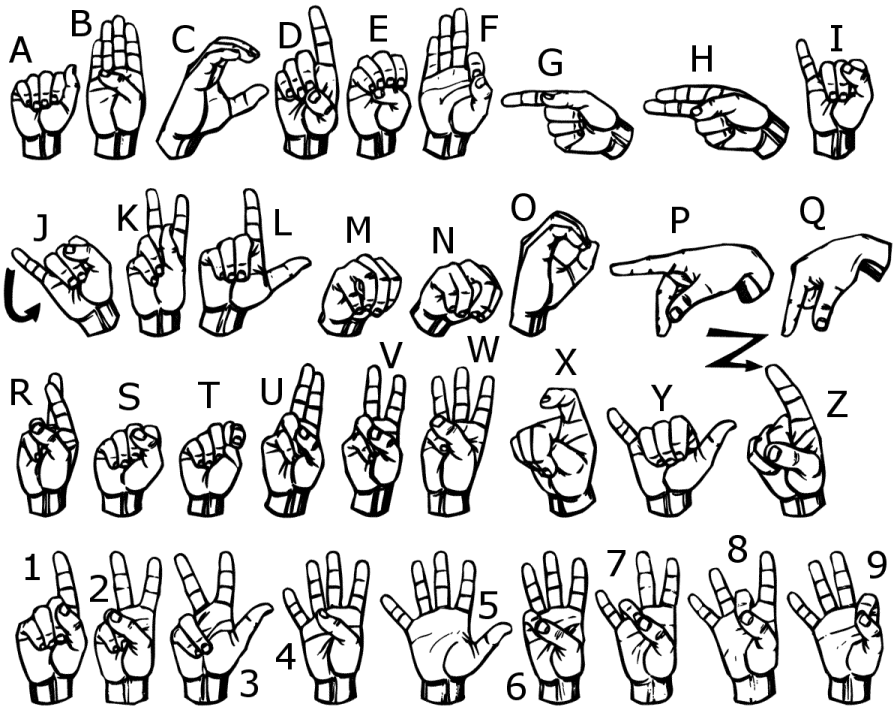


Figure 1: ASL alphabet and digits [19].

Other than finger-spelled letters, ASL includes word-level abstractions such as signs for “hello” and “goodbye”. Some word gestures in ASL stem from commonly practiced signals (such as the gesture for “hello”) while others are original (such as the gesture for the word “no”). Forming a sentence in ASL using gestures follows a different grammar than typical English speakers use. For example, consider the sentence “My Grandma has a green coat.” To construct the sentence in ASL, one would sign “my grandma have green coat”, where each of those words is a separate gesture. For some commonly used phrases, ASL has corresponding gestures such as “brushing teeth” and “look at that”. Figure 2 shows some ASL words and phrases.

Figure 2: Example ASL word and phrase gestures [4].

3 Background and Related Work

There have been multiple studies investigating automated classification of American Sign Language movements. Most recent work focuses primarily on the analysis of video [11, 17, 20, 22]. These methods are generally very accurate in classifying ASL using the relatively large amount of data the camera collects (compared to that obtained by most wearable devices), but a user would need to be constantly facing a camera in order to communicate with others. However, a camera-based approach would be applicable for video conferencing and visual communication over the Internet. In order to increase the accuracy of some classifiers, the participants wear gloves on each hand to easily distinguish their hands from the background [17, 22]. Though accuracy increases as a result, the efficacy of using this method in a real world context is suspect. Other efforts include three different positions of cameras in order to obtain a full three-dimensional mapping of the user's movements [20].

Of those projects that do use wearable devices, the focus is mainly on finger-spelling recognition. Some of these attempts include users wearing special gloves with sensors which transmit gesture data to the computer for recognition [9, 14]. Though providing benefits for recognizing the ASL

alphabet, this type of implementation would prove cumbersome in the real world. One recent master's thesis uses a sensor called the Shimmer as the wearable device [2]. The Shimmer device is able to produce medical-grade data analysis for EMG signals, as well as data from the on-board accelerometer and gyroscope, in order to determine location and movement of the wearer's arm. A more recent project combines both video analysis and wearable accelerometers achieving high accuracy in ASL movement classification [3]. However, the wearables in these works are generally cumbersome and/or are not commercially available.

Because of the computation and battery constraints of the intended mobile platform, scalability is an important consideration in this context. Speech recognition uses phonemes to recognize each word, but with ASL this method is quite difficult as the arm-orientation and hand-shape phonemes can change simultaneously. One study in particular constructed a set of phonemes for ASL consisting of 4 categories: 30 hand-shapes, 8 hand orientations, 20 major body locations, and 40 movements [20]. Though there are many more ASL phonemes than speech phonemes, the phonetic makeup of sign language (total of $4(30 + 8 + 20 + 40) = 392$ classes to identify in parallel) is still less than the currently increasing 6000 ASL vocabulary words. Other projects do not use phonetic breakdown of ASL movements, instead working with an aggregate signal of movements from their choice of device. Most of these aggregate-signal projects have a limited vocabulary of less than 50 ASL movements [2, 17, 21]. One study in particular includes 262 different ASL movements, producing very accurate classification results of up to 94% [7].

Most of the projects using video analysis apply a hidden Markov model (HMM) as the algorithm of choice for classification. Following the popularity of using HMMs for speech recognition, these projects use sub-sampled images from video cameras as feature vectors and obtain accurate results [7, 17, 21]. HMMs allow for fast recognition of ASL movements, but as the vocabulary increases the efficiency and accuracy of the model decreases [20]. Other research investigates parallel HMMs, increasing the speed and accuracy of training the models and improving the real-time response of the system [20]. Other work focusing on wearable devices uses linear discriminant analysis (LDA) as the classification algorithm of choice [2], and is able to achieve very high accuracy.

Because the data gathered from an ASL movement is time series data, other work investigates dynamic time warping (DTW) as a distance metric to compare new samples to samples previously recorded [11]. DTW is an algorithm used to align (by "warping") two different time series to allow comparison of the two series. Although HMMs have been shown to be generally superior to DTW in aligning two offset series, the authors introduce statistical dynamic time warping (SDTW), a modified version of DTW, along with a modified nearest-neighbor algorithm, achieving classification accuracies of up to 91% in a video-based setup [11].

4 The Myo Device

In our work, we use a wearable device called Myo for collecting and transmitting signals related to ASL gestures. The Myo [10] is a consumer electronic device released in March 2015 by Thalmic Labs. Myo's software development kit (SDK) allows developers to access the raw data transmitted from the Myo to a Bluetooth-connected computer. This raw data consists of the output of the onboard accelerometer, gyroscope, orientation, and EMG sensors. A schematic of the Myo device is given in Figure 3. In order to function properly, the Myo device should be positioned on the forearm of the user with the USB cable port pointed toward the wearer's hand. Depictions of the Myo and its orientations relative to the user's arm are given in Figures 4 and 5.

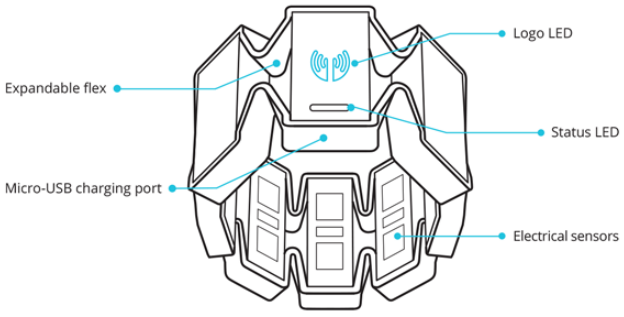


Figure 3: A schematic of the Myo device [10].

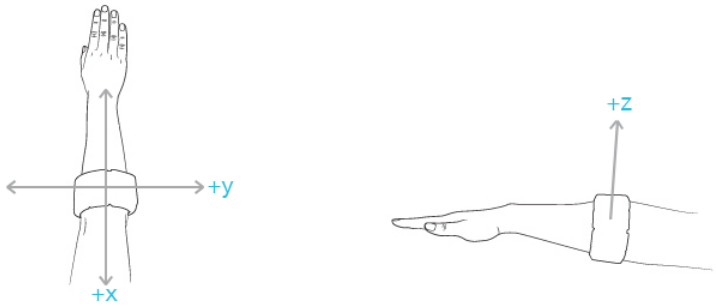


Figure 4: A representation of the Myo coordinate system for accelerometer and gyroscope [5].

4.1 Accelerometer

The Myo contains an accelerometer which estimates acceleration of the arm along the x, y, and z axes, as shown in Figure 4. (Note that our specific Myo returns the same value for accelerometer y and z axes, so we omit the duplicate. Omitting this third dimension is mitigated by the three dimensions of roll, pitch, and yaw, described in Section 4.3.) The device transmits accelerometer data in units of gravity (g), approximately $9.8\ m/s^2$.

4.2 Gyroscope

Along with accelerometer data, the Myo device also transmits gyroscope data, measuring rotation of the band (and therefore the arm). Gyroscope data is captured along all three of the x, y, and z axes shown in Figure 4. (Note that our specific Myo returns the same value for the gyroscope y and z axes, so, similar to the accelerometer, we omit the duplicate. Again, omitting this third dimension is mitigated by the three dimensions of roll, pitch, and yaw.) The Myo transmits gyroscope data in units of radians per second (rad/s).

4.3 Roll, Pitch, Yaw

Unlike the gyroscope and accelerometer, values for roll, pitch, and yaw transmitted by the Myo are all calculated in real-time using other sensor data. The Myo documentation does not report how these values are calculated but documents the particular axes in a way analogous to airplanes. Figure 5 depicts how roll, pitch, and yaw change as the user's arm moves while wearing the Myo.

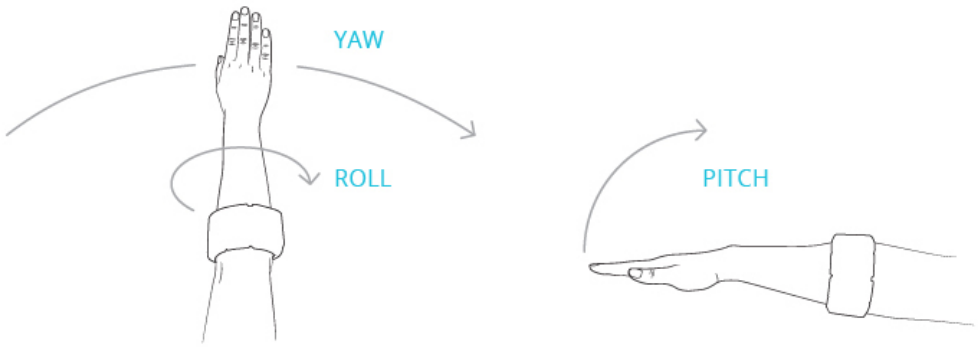


Figure 5: Roll, pitch, and yaw for the Myo device [5].

4.4 Electromyography

Electromyography (EMG) devices read electrical activity from muscles, and have even been used to help control motor-powered prosthetic limbs [13]. In a simple sense, on either contraction or relaxation of a muscle, an EMG device detects and reports an electrical change in that particular muscle. The Myo device has eight EMG sensors on board, corresponding to an additional eight signal components reported on movement. Because the Myo is a small band built specifically for use on the forearm, its EMG sensors are not able to pinpoint exact muscle movements. However, collectively the eight components can be used to identify certain finger gestures. This ability to identify finger movement is critical for determining ASL gestures that differ only in hand/finger positioning.

4.5 Construction of Complete Data Signal

Since Myo data is recorded from four different sensors, our classification system uses one signal consisting of data concatenated from all four sensors. Consider an ASL gesture's signal containing t data points. Let $\boldsymbol{a}_x = (a_{x,1},\ a_{x,2},\ \ldots,\ a_{x,t})$ and $\boldsymbol{a}_y = (a_{y,1},\ a_{y,2},\ \ldots,\ a_{y,t})$ correspond to data transmitted by the accelerometer. Let $\boldsymbol{g}_x = (g_{x,1},\ g_{x,2},\ \ldots,\ g_{x,t})$ and $\boldsymbol{g}_y = (g_{y,1},\ g_{y,2},\ \ldots,\ g_{y,t})$ correspond to data transmitted by the gyroscope. Let transmitted roll, pitch, and yaw signals be represented by $\boldsymbol{o}_r = (o_{r,1},\ o_{r,2},\ \ldots,\ o_{r,t})$, $\boldsymbol{o}_p = (o_{p,1},\ o_{p,2},\ \ldots,\ o_{p,t})$, and $\boldsymbol{o}_y = (o_{y,1},\ o_{y,2},\ \ldots,\ o_{y,t})$, respectively. Finally, let $\boldsymbol{e}_i = (e_{i,1},\ e_{i,2},\ \ldots,\ e_{i,t})$ where $i \in \{1, 2, \ldots, 8\}$ correspond to the eight EMG signals. For our work, we construct an aggregate signal $\boldsymbol{s}$ as follows:

$$\boldsymbol{s} = (\boldsymbol{a}_x,\ \boldsymbol{a}_y,\ \boldsymbol{g}_x,\ \boldsymbol{g}_y,\ \boldsymbol{o}_r,\ \boldsymbol{o}_p,\ \boldsymbol{o}_y,\ \boldsymbol{e}_1,\ \boldsymbol{e}_2,\ \ldots,\ \boldsymbol{e}_8)$$

Recall that identical y and z values are reported by both the accelerometer and the gyroscope. Accordingly, $\boldsymbol{a}_z$ and $\boldsymbol{g}_z$ are omitted from our signal $\boldsymbol{s}$. We normalize the values in each of the sub-vectors $\boldsymbol{a}_x, \boldsymbol{a}_y, \ldots, \boldsymbol{e}_8$ using the mean and standard deviation of the raw values from the corresponding sub-vector.

5 Classification Algorithms

In this section, we discuss the machine-learning algorithms used in this work to classify ASL gestures: linear discriminant analysis, k-nearest neighbors using Euclidean and Manhattan distances, and k-nearest neighbors using dynamic time warping.

5.1 Linear Discriminant Analysis

Linear discriminant analysis (LDA) [15] is a supervised classification algorithm that determines a linear combination of features to separate two or more classes of data. LDA's primary objective is to perform dimensionality reduction while preserving as much of the class specific information as possible [6].

Consider the case of classifying vectors of dimension $m \geq 2$ into $C = 2$ classes. Given m-dimensional training vectors

$$\boldsymbol{x}^i = \left[x_1^i, x_2^i, \ldots, x_m^i\right] \quad i \in \{1, 2, \ldots, N\}$$

where each $\boldsymbol{x}^i$ belongs to either known class ω_1 or ω_2, such that $|\omega_1| = N_1$, $|\omega_2| = N_2$, and $N = N_1 + N_2$, we want to choose a projection vector

$$\boldsymbol{w} = [w_1, w_2, \ldots, w_m]^T$$

such that projections $y^i = \boldsymbol{w}^T \boldsymbol{x}^i$ maximize separability of the scalars/classes. Define the mean of each class of vectors to be

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{\boldsymbol{x}^i \in \omega_1} \boldsymbol{x}^i \qquad \text{and} \qquad \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{\boldsymbol{x}^i \in \omega_2} \boldsymbol{x}^i$$

and the mean of the projection of each class to be

$$\tilde{\mu}_1 = \frac{1}{N_1} \sum_{y^i \in \omega_1} y^i = \frac{1}{N_1} \sum_{\boldsymbol{x}^i \in w_1} \boldsymbol{w}^T \boldsymbol{x}^i = \boldsymbol{w}^T \boldsymbol{\mu}_1 \qquad \text{and} \qquad \tilde{\mu}_2 = \frac{1}{N_2} \sum_{y^i \in \omega_2} = \boldsymbol{w}^T \boldsymbol{\mu}_2 \ .$$

For each class, denote the *scatter*, which measures within-class variability after projection, as

$$\tilde{s}_1^2 = \sum_{y^i \in \omega_1} \left(y^i - \tilde{\mu}_1\right)^2 \quad \text{and} \quad \tilde{s}_2^2 = \sum_{y^i \in \omega_2} \left(y^i - \tilde{\mu}_2\right)^2$$

so that $\tilde{s}_1^2 + \tilde{s}_2^2$ is a measure of the *within-class scatter*.

The goal of LDA is to identify a projection vector $\boldsymbol{w}$ that maximizes the criterion function

$$J(\boldsymbol{w}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} \quad .$$

Intuitively, the numerator indicates that we want large separation between the means of the projections of each class, while the denominator indicates that we want small variance within each class. This, in turn, will maximize the separability of the classes.

Let the covariance matrices of each class be denoted as

$$\boldsymbol{S}_1 = \sum_{\boldsymbol{x}^i \in \omega_1} \left(\boldsymbol{x}^i - \boldsymbol{\mu}_1\right)\left(\boldsymbol{x}^i - \boldsymbol{\mu}_1\right)^T \qquad \text{and} \qquad \boldsymbol{S}_2 = \sum_{\boldsymbol{x}^i \in \omega_2} \left(\boldsymbol{x}^i - \boldsymbol{\mu}_2\right)\left(\boldsymbol{x}^i - \boldsymbol{\mu}_2\right)^T$$

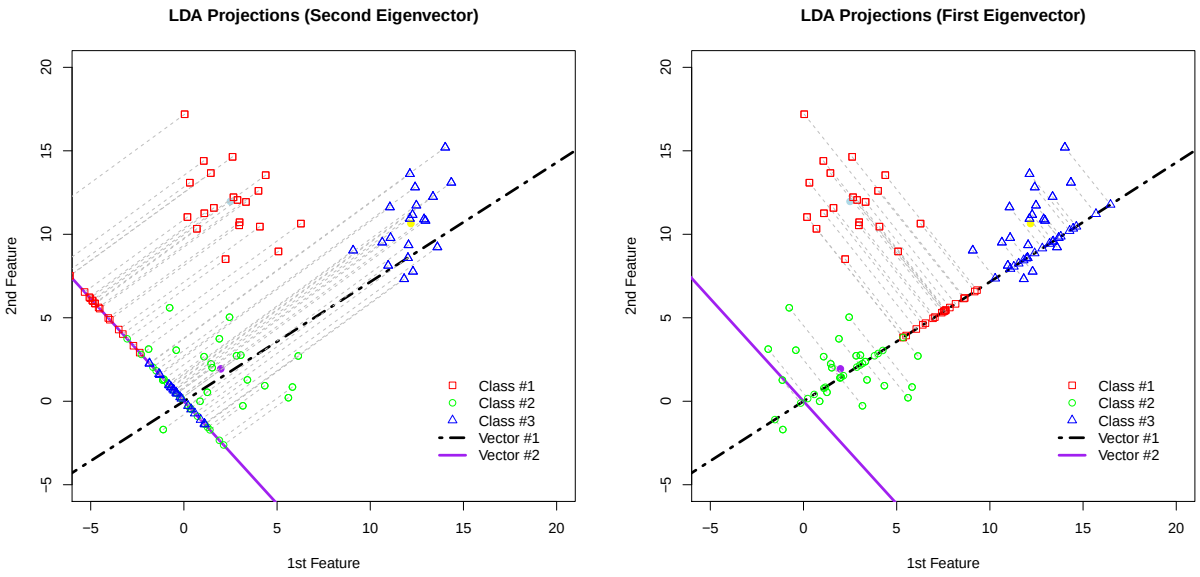
and let $\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2$ be the *within-class scatter matrix*. Similarly, let $\boldsymbol{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ be the *between-class scatter matrix*. The criterion function can then be written as [6]

$$J(\boldsymbol{w}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} \quad .$$

Maximizing the criterion function equates to solving the generalized eigenvalue problem

$$\boldsymbol{S}_W^{-1} \boldsymbol{S}_B \boldsymbol{w} = \lambda \boldsymbol{w}$$

so that the solution will be the eigenvectors of $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$. (A derivation for this claim is provided in [6].) The optimal projection vector $\boldsymbol{w}$ is the eigenvector corresponding to the largest eigenvalue from the solution. This notation can be extended to $C > 2$ classes [6].



(a) Points projected onto second eigenvector.

(b) Points projected onto first eigenvector.

Figure 6: LDA projection onto eigenvectors for three classes of vectors.

Consider an example for $C = 3$ classes. Let $\boldsymbol{x}^i$, $i \in \{1, 2, \ldots, 60\}$, be two-dimensional vectors where each component of a 2-D vector is randomly generated from a standard normal distribution. Assign 20 vectors to each of the three classes, and then, via transformation using an arbitrary covariance matrix, shift the center and orientation of each of the three classes in 2-D space. Let $\boldsymbol{w}_1$ be the projection vector (eigenvector) that corresponds to the largest eigenvalue from the solution described earlier, and $\boldsymbol{w}_2$ be the eigenvector corresponding to the second largest eigenvalue. Figure 6 shows two graphs with the vectors projected onto the eigenvectors $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. In Figure 6a the vectors are projected onto $\boldsymbol{w}_2$, corresponding to the second largest eigenvalue, with limited separation between classes 2 and 3. However, Figure 6b depicts the projection of the vectors onto $\boldsymbol{w}_1$, corresponding to the largest eigenvalue, with distinguishable separation among all three classes.

Now consider classification using LDA. Let $\boldsymbol{s}$ be a data signal with an unknown class, and M be the set of training signals with class identification. LDA will compute the optimal projections for M, and will compute the means of the projections of each class. Then $\boldsymbol{s}$ is classified using the distance

between the projection of $\boldsymbol{s}$ and the class projection means.

5.2 k-Nearest Neighbors (k-NN)

k-Nearest Neighbors (k-NN) [1] is an alternative (supervised) classification algorithm that classifies a vector of unknown class by finding k classified vectors having minimum distance to the unclassified vector, where the distances are some quantifiable metric. The training process of k-NN is straightforward: k-NN simply stores an array of training data vectors and their respective classes. Because k-NN is a supervised algorithm, the class of each training vector must be indicated.

Now consider classification using k-NN. Let $\boldsymbol{s}$ be a data signal with an unknown class, and M be the set of training signals with class identification. k-NN will compare $\boldsymbol{s}$ to every record in the set M, and classify $\boldsymbol{s}$ as belonging to the class most common among the k records in M closest to $\boldsymbol{s}$.

As mentioned above, k-NN requires a means of quantifying distance, i.e., similarity, of two different samples. The three measures of similarity we consider for k-NN in this work are Euclidean distance, Manhattan distance, and dynamic time warping distance.

5.2.1 Euclidean Distance

The Euclidean distance is a measure of the distance between two points in a given coordinate space. Let $\boldsymbol{a} = (a_1, a_2, a_3, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, b_3, \ldots, b_n)$ be points in n-space. The Euclidean distance $d(\boldsymbol{a}, \boldsymbol{b})$ between points $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined by

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + \ldots + (a_n - b_n)^2} \ .$$

In the context of k-NN, if $\boldsymbol{m}$ is an arbitrary signal from the training set M and $\boldsymbol{s}$ is a yet-unseen signal, $d(\boldsymbol{s}, \boldsymbol{m})$ can be used to quantify the similarity of the two n-dimensional signals.

5.2.2 Manhattan Distance

The Manhattan, or taxicab, distance is the vertical and horizontal distance between two points in a given coordinate space. Let $\boldsymbol{a} = (a_1, a_2, a_3, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, b_3, \ldots, b_n)$ be points in n-space. The Manhattan distance $d(\boldsymbol{a}, \boldsymbol{b})$ between points $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined by

$$d(\boldsymbol{a}, \boldsymbol{b}) = |a_1 - b_1| + |a_2 - b_2| + |a_3 - b_3| + \ldots + |a_n - b_n| \ .$$

Similar to Euclidean distance, if $\boldsymbol{s}$ and $\boldsymbol{m}$ are both arbitrary signals, $d(\boldsymbol{s}, \boldsymbol{m})$ can be used to quantify the similarity of the two n-dimensional signals.

5.2.3 Dynamic Time Warping

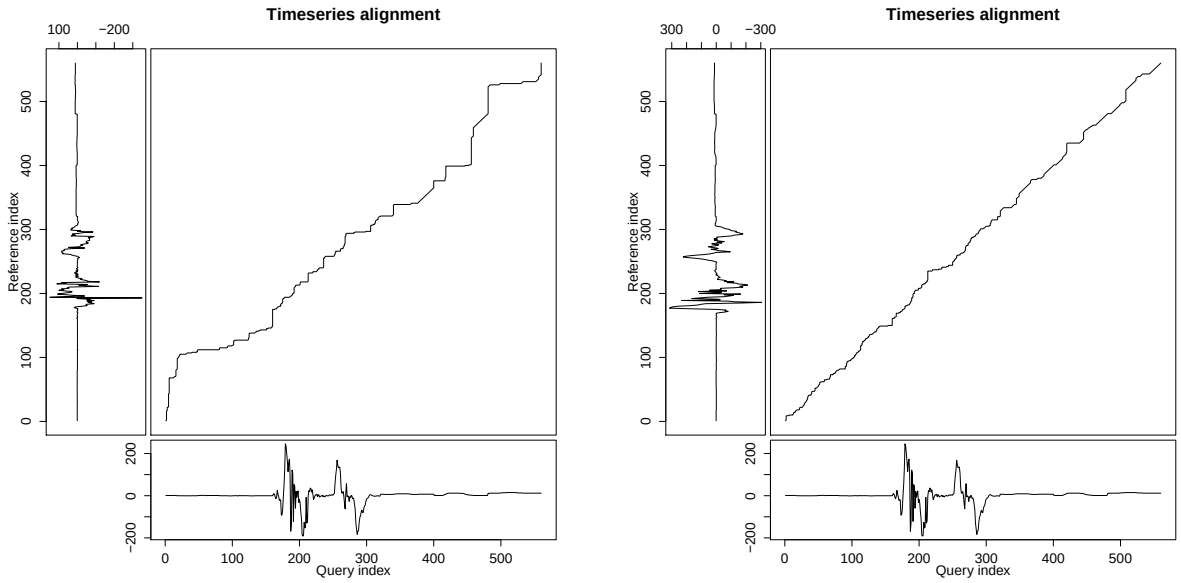
Dynamic time warping (DTW) is a distance metric for measuring the similarity between two time series. The primary benefit of DTW is its ability to equate two time series that differ in length but are similar in overall shape. DTW was once widely used in speech recognition applications, but has generally been supplanted by HMMs in that context.

DTW attempts to determine an optimal alignment between two time series using a dynamic programming approach. Consider the comparison of two vectors $\boldsymbol{a} = (a_1,\ a_2,\ \ldots,\ a_n)$ and $\boldsymbol{b} = (b_1,\ b_2,\ \ldots,\ b_m)$ — unlike Euclidean distance, the vectors need not be of the same size. Construct an $(n+1) \times (m+1)$ distance matrix where entry (i, j) represents the distance between a_i and b_j. Finding the optimal alignment is equivalent to finding the shortest path from the bottom-left $(0, 0)$ to the top-right (n, m) of the distance matrix. The resulting distance between the two vectors is the sum of the entries along the shortest path. Pseudocode for the $O(nm)$ DTW algorithm is given below.

input: $\boldsymbol{a} = (a_1, a_2, a_3, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, b_3, \ldots, b_m)$

$dtw \leftarrow$ array$[0 \ldots n][0 \ldots m]$
set all cells in dtw to ∞
$dtw[0][0] = 0.0$
$i, j \leftarrow 1$
while $i < n$ **do**
    $j \leftarrow 1$
    while $j < m$ **do**
        $c \leftarrow |a_i - b_j|$
        $dtw[i][j] \leftarrow c + \min(dtw[i-1][j], dtw[i][j-1], dtw[i-1, j-1])$
        $j \leftarrow j + 1$
    end
    $i \leftarrow i + 1$
end
return $dtw[n][m]$

Figure 7 provides an example of the shortest path between two ASL gesture signals as determined by DTW. For brevity, the data used here include only the accelerometer x values produced by the Myo (see Section 4.1). Figure 7a shows the comparison of signals from two different gestures ("apple" and "dog"). Note that the optimal alignment path deviates from the diagonal, indicating that the two signals are not relatively similar. Figure 7b shows the comparison of two different signals of the same gesture ("apple"). In this case, the optimal alignment path closely follows the diagonal, indicative that the two signals are similar. Note that the overall distance computed, $dtw([n][m])$, is smaller for Figure 7b than for Figure 7a.



(a) “apple” signal vs. “dog” signal (b) Two different signals of “apple”

Figure 7: Optimal alignment paths between two signals as determined by DTW.

5.2.4 Dynamic Time Warping with Windowing

The $O(nm)$ run-time of DTW can be improved using windowing. To do so, rather than computing distances for all $(n + 1)(m + 1)$ entries in the distance matrix, only those entries in a window around the optimal path should be computed. An appropriate choice of window size will improve efficiency without affecting accuracy. Unfortunately, trial-and-error must be used for selecting the window size. Pseudocode for DTW with windowing is given below. Compared to the previous DTW algorithm, note the additional input parameter w and the changes to incorporate w on lines 6 and 7 of the pseudocode.

input: $\boldsymbol{a} = (a_1, a_2, a_3, \ldots, a_n)$, $\boldsymbol{b} = (b_1, b_2, b_3, \ldots, b_m)$, and w = Window Size

```
dtw ← array[0...n][0...m]
set all cells in dtw to ∞
dtw[0][0] = 0.0
i, j =← 1
while i < n do
    j ← max(1, i − w)
    while j < min(m, i + w) do
        c ← |a_i − b_j|
        dtw[i][j] ← c + min(dtw[i − 1][j], dtw[i][j − 1], dtw[i − 1, j − 1])
        j ← j + 1
    end
    i ← i + 1
end
return dtw[n][m]
```

Figure 8 provides an example. The black squares represent a portion of the optimal path between two signals, similar to the paths shown in Figure 7. The gray squares represent the window of entries computed in determining the optimal alignment path. Uncolored squares represent entries whose values are not computed.

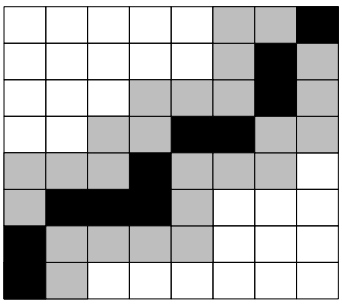


Figure 8: DTW Windowing Example

6 Implementation of the Prototype

In this section, we describe implementation details of our prototype, including capturing gestures from the Myo signals, sampling of signals, details of the classification process, and choice of programming platform.

6.1 Capturing an ASL Gesture

To recognize the signal corresponding to an ASL movement, our system must know when the movement starts and ends — a similar process is needed for recognizing speech. Unlike speech, ASL movements do not have pauses between words. Most movements by native ASL speakers never have the same starting position but instead mesh together to create one long movement potentially consisting of multiple words.

To ease the problem of gesture recognition, for our prototype we add a constraint that the user must place their arms at their side at the start and end of each signed word. This gives the classifier a base starting point from which to determine the start and end of any gesture. Given this constraint, we used two different approaches for recognizing an ASL gesture by attempting to automatically recognize the transition from and to that base resting position.

6.1.1 Static Windowing Capture

To recognize the starting and ending times of an ASL gesture, our first approach uses a fixed window consisting of all data points that occur within a particular range. To define this range, various data signals are collected with the user's arm at their side. The window range is then defined to be 10% above and below the average of the accelerometer x values from the resting arm position. Because the accelerometer x values are approximately zero whenever the arm is positioned at the side pointing down toward the ground, we focus only on accelerometer x values here.

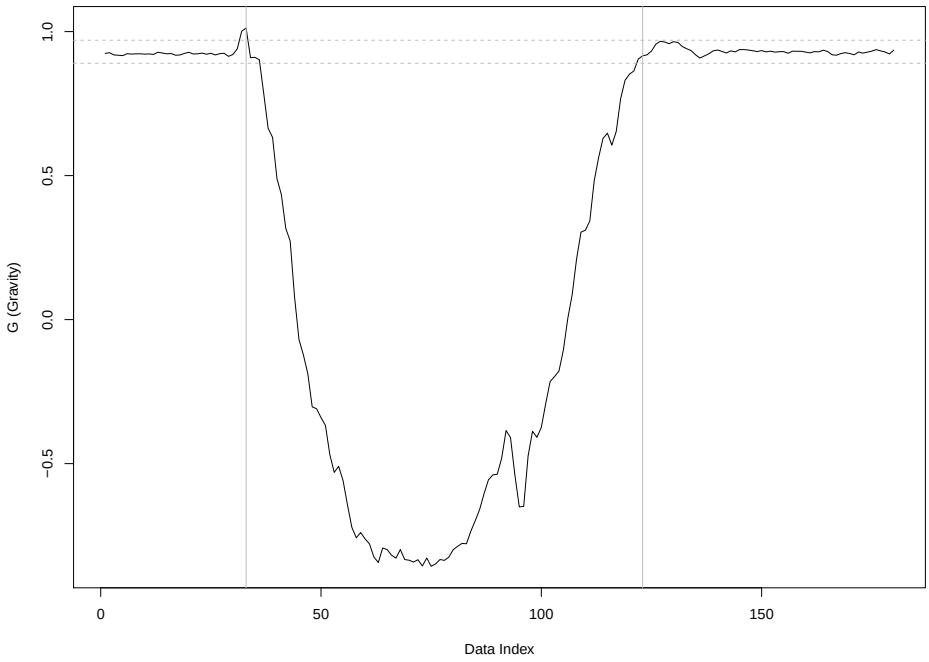


Figure 9: Static windowing capture of “mom” ASL Movement. Dashed horizontal lines depict the static window presumed to be the resting position. Solid vertical lines indicate the start and end of the gesture as determined by our system. Only accelerometer x values are shown.

In Figure 9, the accelerometer x values reported by the Myo for the ASL gesture “mom” show relatively flat portions of signal at the start and end. These correspond to the user having their arms at their side, transitioning from the previous or to the next gesture. Once the signal begins to shift away from the flat portion, those data are considered to be the start of a gesture and are collected by the classifier. In Figure 9, the gray-dashed horizontal lines depict a window around the signal that the system considers to be a resting arm. Once the signal leaves the window for a specified amount of time, a gesture is considered to have started. Once the signal reenters and remains within the window for the same specified amount of time, the gesture is considered to have stopped. These start and stop locations are depicted by solid vertical lines in Figure 9. The data between the two vertical lines is then sent for classification.

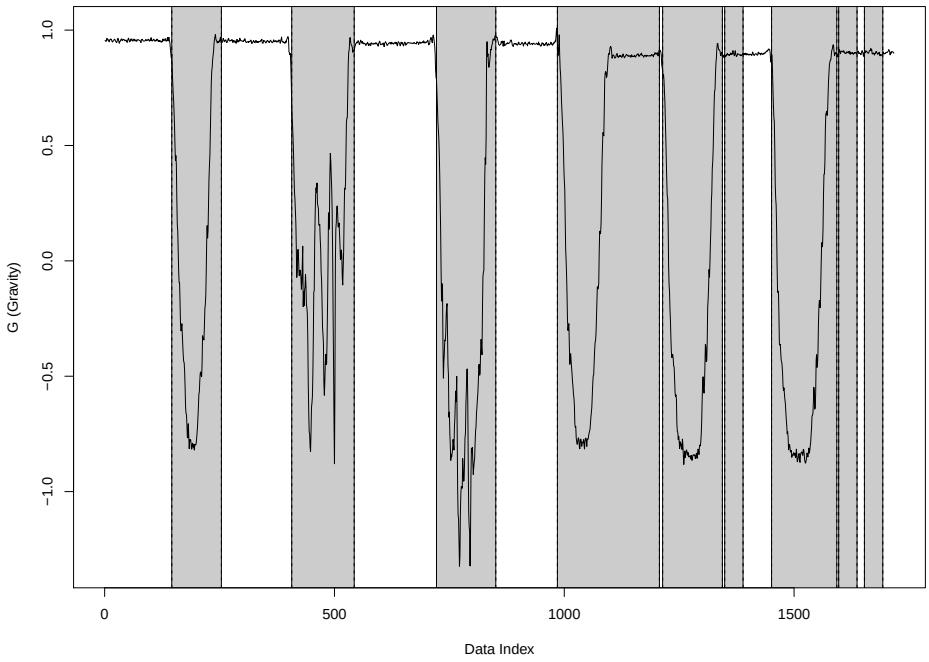


Figure 10: Static windowing capture of multiple movements. Only accelerometer x values are shown.

The static window approach works for the majority of isolated test cases (gestures), but fails when applied in real-time tests. Figure 10 highlights two issues encountered when using this approach in real-time: (1) in some cases too much resting signal is included as part of a legitimate gesture and (2) in some cases, resting signal alone is considered to be an entire gesture. For the first issue, some of the data corresponding to a resting position are not in the predefined "window". Increasing the size or position of the static window does not alleviate these issues. Because the resting position is not consistent between gestures, a window determined dynamically is more appropriate.

6.1.2 Dynamic Windowing Capture

A more effective approach for determining the start and end of an ASL gesture uses a dynamically-determined window for the resting position. While the static window approach attempts to identify

long periods of data inside a fixed vertical range, the dynamic window approach identifies data within a similar vertical range regardless of location of that range. In this way, the dynamic window approach allows the resting window to shift to accurately capture different resting positions.

Our approach for dynamically identifying a resting-position window proceeds as follows. Let the range (height) of the resting-position window be 2ϵ, where $\epsilon > 0$. Let data point r, with value $v(r)$, be the reference point for the window. Let $\alpha \geq 1$ be the window's current length (i.e., number of successive data points in the window), and let $\beta \geq 1$ be the minimum length of a window considered to be a resting window.

Initially, the first data point p_1 encountered becomes the reference, so that $r = p_1$, the window has range $[v(r) - \epsilon, v(r) + \epsilon]$, and the window length is $\alpha = 1$. If data point p_2 has value $v(p_2) \in [v(r) - \epsilon, v(r) + \epsilon]$, then p_2 is added to the window, now of length $\alpha = 2$. If instead p_2 has value $v(p_2) \notin [v(r) - \epsilon, v(r) + \epsilon]$, then a new window is constructed with p_2 as the reference point, i.e., $r = p_2$, redefining the window range, and with window length now $\alpha = 1$.

At any time, let the reference point be denoted by $r = p_i$ where $i = 1, 2, \ldots$ Any subsequent data point p_{i+m}, where $m = 1, 2, \ldots$, is either added to the existing window or used to create a new window as follows:

- If $v(p_{i+m}) \in [v(r) - \epsilon, v(r) + \epsilon]$, then p_{i+m} is added to the window, increasing the window length α by 1.
- If $v(p_{i+m}) \notin [v(r) - \epsilon, v(r) + \epsilon]$, then the smallest integer j is chosen to satisfy $v(p_{i+k}) \in [v(p_{i+j}) - \epsilon, v(p_{i+j}) + \epsilon]$, where $j = 1, 2, \ldots, m$ and $k = j, j+1, \ldots, m$. In other words, the earliest point p_{i+j} is chosen so that each of p_{i+j}, p_{i+j+1}, $\ldots$, p_{i+m} are within the range $[v(p_{i+j}) - \epsilon, v(p_{i+j}) + \epsilon]$. Then $r = p_{i+j}$ becomes the new reference point, defining a new window with range $[v(r) - \epsilon, v(r) + \epsilon]$ and resulting window length $\alpha = m - j + 1$.

Whenever the window length $\alpha \geq \beta$, any subsequent data point outside the window is considered to be the starting point of an ASL gesture. The ending point of that ASL gesture is determined to be

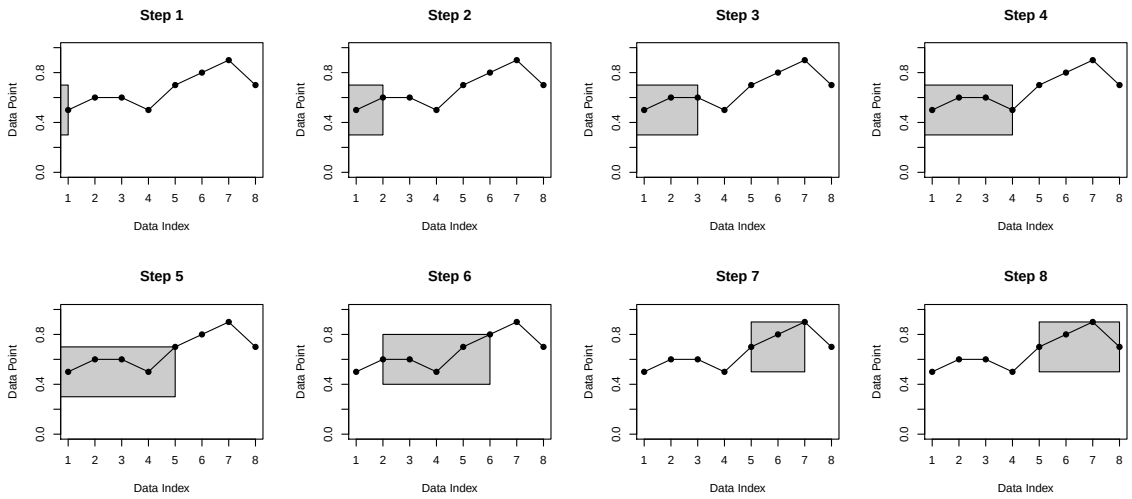


Figure 11: Dynamic windowing example.

the next point in sequence that occurs just before the reference point of the next (resting) window having length $\alpha \geq \beta$. (We assume that the user does not pause during execution of a gesture.)

Figure 11 provides an illustration of our dynamic window approach for determining a resting window. In this example, the window half-width $\epsilon = 0.2$ and minimum-length parameter $\beta = 6$.

Step 1: The first data point p_1 is added to the window, becoming the window's reference point. Since the value of p_1 is 0.5, the window range is $[0.3, 0.7]$, i.e., $0.5 \pm \epsilon$, and window length is 1.

Step 2: The second point p_2 has value 0.6, within the window's range $[0.3, 0.7]$. Thus p_2 is added to the window, which now has length 2.

Step 3: Point p_3 also has value 0.6, within the window's range $[0.3, 0.7]$. Thus p_3 is added to the window, which now has length 3.

Step 4: Point p_4 has value 0.5, and so is added to the window, now of length 4.

Step 5: Point p_5 has value 0.7, and so is added to the window, now of length 5.

Step 6: Point p_6 has value 0.8, outside the window's range $[0.3, 0.7]$, so a new window must be found. Back-tracking is performed to select a reference point for the new window such that the

new window contains p_6. p_2 is the earliest point in the sequence that, when considered the reference point for the window, results in a window containing points $p_2, p_3, \ldots, p_6$. Thus p_2, with value 0.6, becomes the new reference point, the window's range becomes $[0.4, 0.8]$, and the length of the window is 5.

Step 7: Point p_7 has value 0.9, outside the window's range $[0.4, 0.8]$, so a new window must again be found. Back-tracking to find the earliest possible reference point results in p_5 being selected as the new window's reference point. Thus the new window length is 3 with range $[0.5, 0.9]$.

Step 8: Point p_8 has value 0.7, within the window's range $[0.5, 0.9]$, so p_8 is added to the window, now of length 4.

The process above is repeated until (a) the window length is at least β and (b) the subsequent data point lies outside the window's range. This new point is considered to be the starting point of the ASL gesture. (Note that if in this example the minimum-length parameter were $\beta = 5$, the point at step 7 would be considered the first point of the ASL gesture.) The ending point of the gesture is considered to be the next point that occurs just before the next resting window is identified. Once the ending point of the gesture is determined, the data for the ASL gesture are sent for classification.

This dynamic window approach improves the reliability for real-time capturing of ASL movements. Figure 12 shows the successful capture of different ASL gestures in sequence using this approach. The approach is effective at isolating gestures even when the user does not always position their arm in exactly the same resting position.

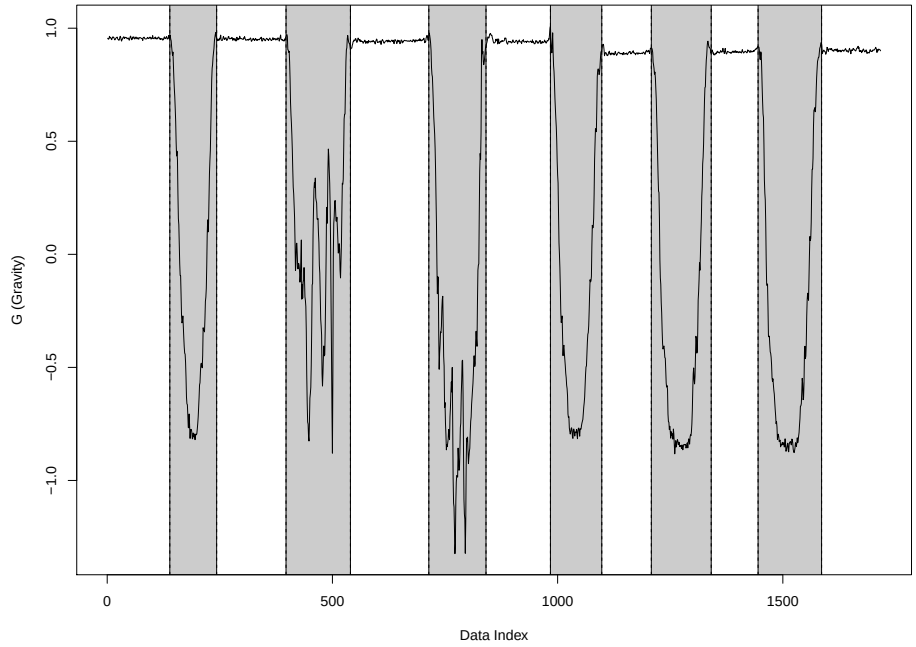


Figure 12: Dynamic windowing capture of multiple ASL gestures. Only accelerometer x values are shown. Note that different resting positions (horizontal portions of signal) are correctly omitted.

6.2 Signal Sampling

Once an ASL gesture has been appropriately "trimmed" of resting position signal, we sample a fixed number of evenly-spaced samples across the entire signal, regardless of signal duration. This sampling process improves computational efficiency in classification by reducing the number of data points in a signal passed to the classifier. More importantly, the sampling allows us to meaningfully compare two signals that are similar in shape but vary in duration.

An example is given in Figure 13. On the left side, Figures 13a and 13c show two signals of different duration for the ASL gesture "mom" prior to sampling. The vertical lines represent the fixed number of samples being chosen. Notice how the two signals are very similar in shape but the signal in 13c was completed in roughly half the time as that in Figure 13a. The resulting sampled versions of each signal, show in Figures 13b and 13d, are very similar both in shape and in duration, facilitating more accurate classification.

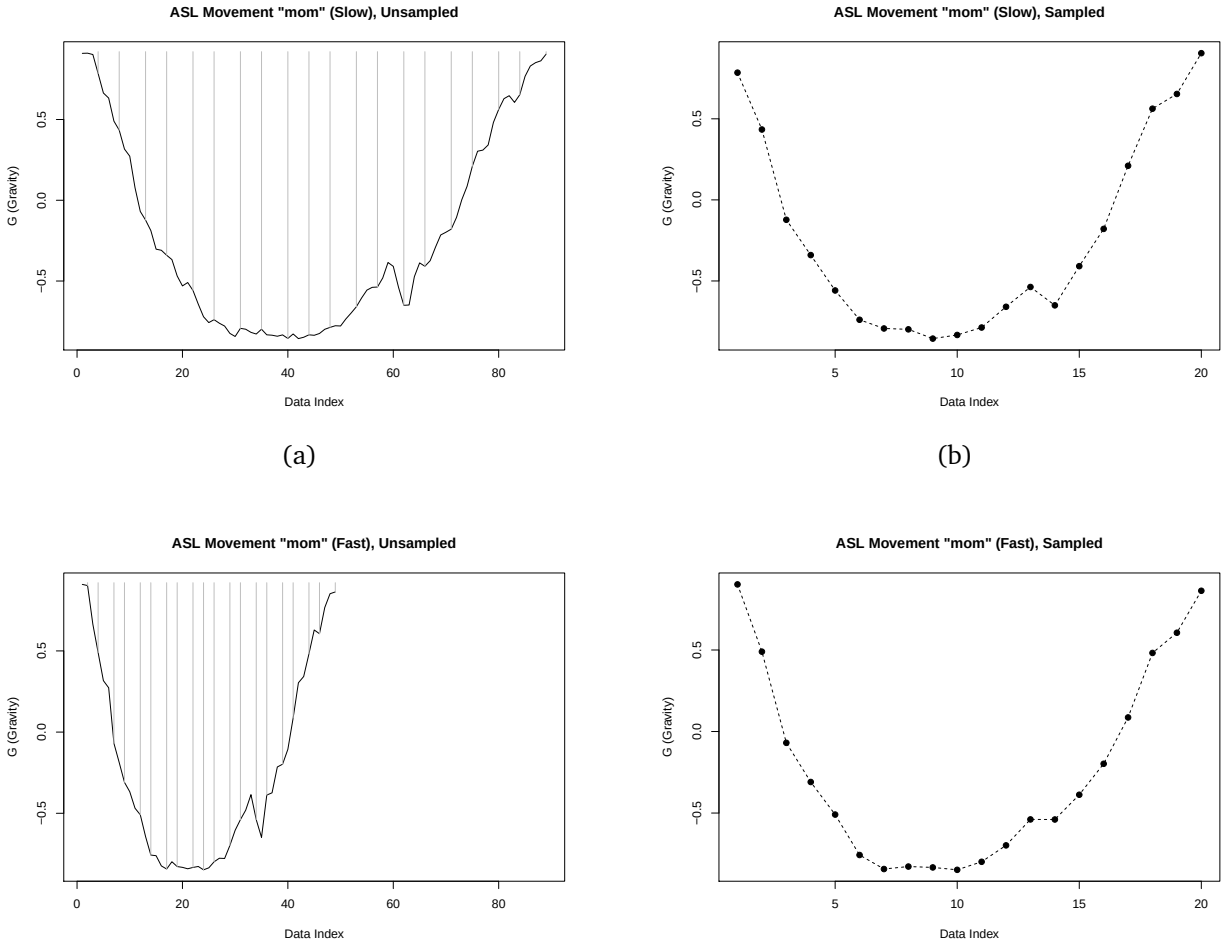


Figure 13: Examples of sampling two signals of the same ASL gesture that vary in duration.

6.3 Classification Process

Consider a user wearing the Myo who begins to perform an ASL gesture from one of the C classes of gestures. Our system will capture the ASL gesture from the Myo device using dynamic windowing and signal sampling, as discussed in Sections 6.1.2 and 6.2. The result is a data signal $\boldsymbol{s}$ of unkown class where each component is a time series (as described in Section 4.5)

$$\boldsymbol{s} = (\boldsymbol{a}_x,\ \boldsymbol{a}_y,\ \boldsymbol{g}_x,\ \boldsymbol{g}_y,\ \boldsymbol{o}_r,\ \boldsymbol{o}_p,\ \boldsymbol{o}_y,\ \boldsymbol{e}_1,\ \boldsymbol{e}_2,\ \ldots,\ \boldsymbol{e}_8)\ .$$

The classification algorithm (either LDA or k-NN) takes the data signal $\boldsymbol{s}$ and determines the signal class that most closely matches according to the algorithm's similarity metric.

Training signals are used by each of the classification algorithms. To this end, given a collection of C different ASL gestures to classify, in advance we capture n signals of each gesture. Let $m_{i,j}$ be a captured signal, where $i = 1, 2, \ldots, C$ represents the class (ASL gesture) and $j = 1, 2, \ldots, n$ represents a particular trimmed, sampled signal of that class. Then the set

$$M = \{m_{1,1},\, m_{1,2},\, \ldots,\, m_{1,n}, \quad m_{2,1},\, m_{2,2},\, \ldots,\, m_{2,n}, \quad \ldots, \quad m_{k,1},\, m_{k,2},\, \ldots,\, m_{k,n}\}$$

defines the set of training signals.

6.3.1 Classification using LDA

The process of classification using LDA proceeds similar to the discussion given in Section 5.1. LDA accepts the set of training signals M, computes optimal projections for each class of signals, and then computes the means of each class of projections. Then given a signal $\boldsymbol{s}$ of unknown class, LDA classifies $\boldsymbol{s}$ by determining the class having minimum distance between the class projection mean and the projection of $\boldsymbol{s}$.

6.3.2 Classification using k-NN

As discussed in Section 5.2, the k-NN algorithm classifies using a majority vote of its k nearest neighbors. To balance computational efficiency with classification accuracy, we used k-NN with $k = 1$ along with a set of reference signals defined for each class as follows. Given the training set M, the Cn signals are grouped based on their corresponding gesture, as shown below.

$$\underbrace{m_{1,1} \quad m_{1,2} \quad \ldots \quad m_{1,n}}_{\bar{\boldsymbol{m}}_1} \quad \underbrace{m_{2,1} \quad m_{2,2} \quad \ldots \quad m_{2,n}}_{\bar{\boldsymbol{m}}_2} \quad \ldots \quad \underbrace{m_{k,1} \quad m_{k,2} \quad \ldots \quad m_{C,n}}_{\bar{\boldsymbol{m}}_C}$$

For each class $i = 1, 2, \ldots, C$, we compute the average signal $\bar{\boldsymbol{m}}_i$ for that class using

$$\bar{\boldsymbol{m}}_i = \frac{\bar{\boldsymbol{m}}_{i,1} + \bar{\boldsymbol{m}}_{i,2} + \ldots \bar{\boldsymbol{m}}_{i,n}}{n} \quad .$$

Let the set $\bar{M} = \{\bar{\boldsymbol{m}}_1, \bar{\boldsymbol{m}}_2, \ldots, \bar{\boldsymbol{m}}_C\}$ be the collection of reference signals for the C classes, each corresponding to a different ASL gesture.

For k-NN with either Euclidean or Manhattan distance, the corresponding distance function can be applied directly to signal $\boldsymbol{s}$ with each of the reference signals in $\bar{M}$. For k-NN with DTW distance, a direct comparison between $\boldsymbol{s}$ and a reference signal is not appropriate because of the "warping". That is, the DTW algorithm should not compare values from different sensors on the Myo, particularly those with different units. Therefore, we compute the distance, $DTW^*(\boldsymbol{s},\ \bar{\boldsymbol{m}}_i)$, between signal $\boldsymbol{s}$ and a reference signal $\bar{\boldsymbol{m}}_i$, $i = 1, 2, \ldots, C$, by summing the DTW computed distances between the individual components:

$$DTW^*(\boldsymbol{s},\ \bar{\boldsymbol{m}}_i) = DTW(\boldsymbol{s} : \boldsymbol{a}_x,\ \bar{\boldsymbol{m}}_i : \boldsymbol{a}_x) + DTW(\boldsymbol{s} : \boldsymbol{a}_y,\ \bar{\boldsymbol{m}}_i : \boldsymbol{a}_y) + \ldots + DTW(\boldsymbol{s} : \boldsymbol{e}_8,\ \bar{\boldsymbol{m}}_i : \boldsymbol{e}_8)$$

where the notation $\boldsymbol{s} : \boldsymbol{a}_x$ indicates use of only the $\boldsymbol{a}_x$ component of the signal $\boldsymbol{s}$.

DTW windowing reduces the run-time of the DTW distance metric by limiting the number of computations needed for the final distance calculation (see Section 5.2.4). Unfortunately there is no known algorithm to optimally select a window size because of the window's dependence on the type of data. Instead, multiple window sizes must be tested and compared. (For this comparison, see Section 7.4.)

6.4 Swift Programming Language

Because of the prevalence of iOS devices in today's society, an iOS mobile device acts as an appropriate hardware platform for our application. Therefore, Swift — an open source, general purpose programming language built and released by Apple [8] — was chosen as the base language for

our implementation. Swift offers comprehensive built-in functionality, executes efficiently even on mobile platforms, and can be easily installed on either a desktop computer or mobile device. This portability allows for careful testing of the prototype in a desktop computer setting and, appropriately, deployment on a mobile device. In addition, the Myo SDK was built for the Swift language, making Swift a natural choice.

7 Experiments and Results

In this section, we present the results of experiments comparing the accuracy of the LDA and k-NN algorithms for classifying Myo-generated ASL gesture signals. We also present results from experiments that consider computational efficiency.

7.1 Comparison of Classification Algorithms

The purpose of this experiment is to compare the accuracy achieved by each of the classification algorithms under the same training and test sets of ASL gestures. We include 10 different ASL gestures: "apple", "cat", "dog", "home", "hot", "like", "mom", "please", "shirt", and "thank you" [18]. For each gesture, 20 different signals of the gesture were captured, giving 200 signals total. Each signal is captured as the user performs the gesture while wearing the Myo device, with their arm resting at their side before and after the gesture. Each signal is captured using dynamic windowing (see Section 6.1.2) and sampled 60 times (see Section 6.2). To construct the training set, for each of the 10 gestures we select at random 10 signals, giving a training set of size 100. The test set then consists of the remaining 100 signals. We then compare the accuracy of each classification algorithm on the test set.

For this experiment, we use the $lda(\ldots)$ function implemented in the MASS library in R. For k-NN, we implemented the classifier in Swift using three separate distance metrics: (1) Euclidean distance, (2) Manhattan distance, and (3) dynamic time warping (DTW). Collectively, these result

in four different classifiers: LDA, k-NN Euclidean, k-NN Manhattan, and k-NN DTW.

Figure 14 and Table 1 show the accuracy of each classifier in correctly classifying signals from the test set. The first three values (solid bars) of each classifier section represent three different replications of the experiment using different training and test sets selected at random. The fourth value (bar with pattern) is the average accuracy of 30 replications with 95% confidence intervals superimposed. Figure 14 and Table 1 demonstrate that k-NN is consistently more accurate than LDA. k-NN DTW consistently outperforms k-NN with either Euclidean or Manhattan distance metrics.

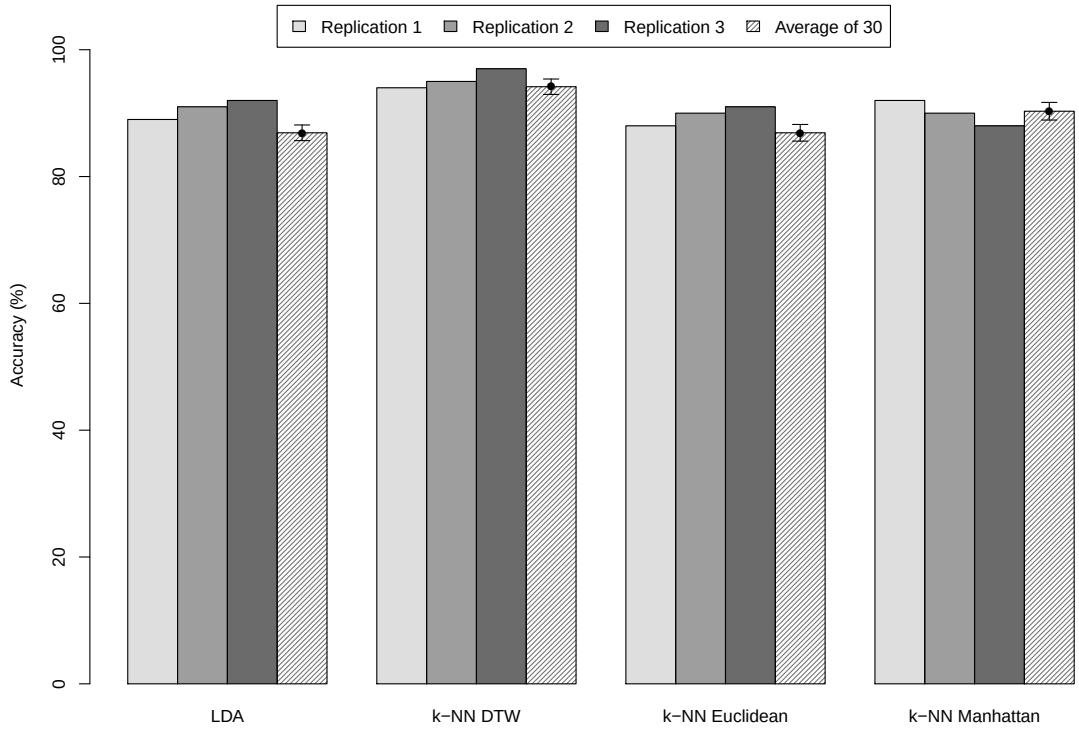


Figure 14: Classification accuracy comparison of LDA, k-NN DTW, k-NN Euclidean, and k-NN Manhattan. Each accuracy uses a test set of 100 signals selected at random.

Classifier	Accuracy (%)			
	Replication 1	Replication 2	Replication 3	Average of 30 Replications
LDA	89.0	91.0	92.0	86.9
k-NN DTW	94.0	95.0	97.0	94.2
k-NN Euclidean	88.0	90.0	91.0	86.9
k-NN Manhattan	92.0	90.0	88.0	90.3

Table 1: Comparison of classification accuracies for four classifiers.

7.2 Comparison of Classification Algorithms: Synthetic Data

To compare classifier accuracy under a large data set we generated synthetic data by randomly perturbing signals from the previous experiment. Since k-NN DTW consistently outperformed the other classification algorithms, k-NN DTW was the algorithm of choice for this experiment (see Section 7.1).

As in the previous experiment, 10 ASL gestures are used: "apple", "cat", "dog", "home", "hot", "like", "mom", "please", "shirt", and "thank you" [18]. The training set again consists of 10 original signals selected at random from each of the 10 gestures. The test set then consists of 500 perturbed signals, where each of the 100 original signals is perturbed 5 times.

We perturb a signal using an offset o ranging from 0.01 to 0.20. For example, let $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$ be a signal to be perturbed. A perturbed signal $\boldsymbol{s}^* = (s_1^*, s_2^*, \ldots, s_n^*)$ is generated using

$$s_i^* = s_i + (\textit{uniform}(-o, o) \times s_i),$$

where $i = 1, 2, \ldots, n$ and *uniform*$(-o, o)$ randomly selects a number in the range $(-o, o)$ such that all real numbers between $-o$ and o are equally likely.

Figure 15 shows the classification accuracy of k-NN DTW versus increasing perturbation offset. Each value shown is the average of 30 replications for that specific offset value, with 95% con-

fidence interval superimposed. k-NN DTW again performs consistently well, achieving approximately 98% accuracy across the replications.

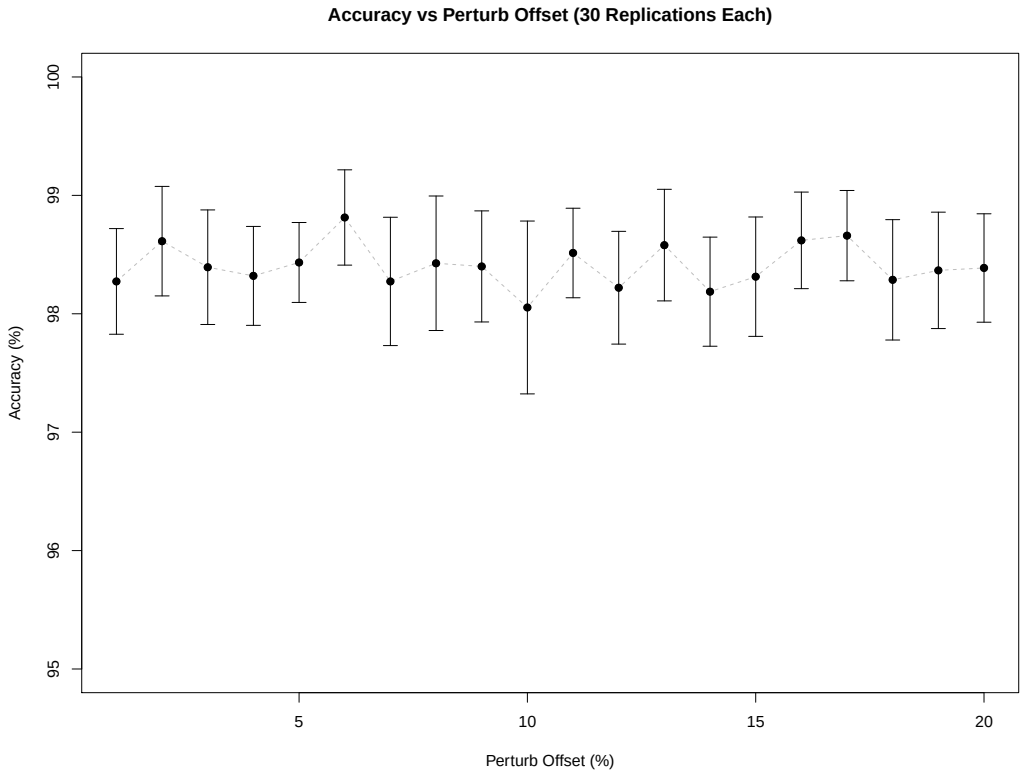


Figure 15: k-NN classification accuracy versus increasing perturbation of synthetic signals.

Perturb Offset	Accuracy (%)	Perturb Offset	Accuracy (%)	Perturb Offset	Accuracy (%)
0.01	98.2 ± 0.4	0.08	98.4 ± 0.6	0.15	98.3 ± 0.5
0.02	98.6 ± 0.5	0.09	98.4 ± 0.5	0.16	98.6 ± 0.4
0.03	98.4 ± 0.5	0.10	98.1 ± 0.7	0.17	98.6 ± 0.4
0.04	98.3 ± 0.4	0.11	98.5 ± 0.4	0.18	98.3 ± 0.5
0.05	98.4 ± 0.3	0.12	98.2 ± 0.5	0.19	98.4 ± 0.5
0.06	98.8 ± 0.4	0.13	98.6 ± 0.5	0.20	98.4 ± 0.5
0.07	98.3 ± 0.5	0.14	98.2 ± 0.5		

Table 2: Classification accuracy of k-NN DTW for increasing perturbation of synthetic signals.

7.3 Increasing the Vocabulary Size

The purpose of this experiment is to test the k-NN DTW classifier as the number of ASL gestures in the vocabulary increases. For the setup, 20 ASL gestures are recorded with 20 different signals of each gesture. The 20 ASL gestures include: "apple", "bathroom", "brushing teeth", "candy", "cat", "cereal", "cow", "drink", "home", "horse", "hot", "hungry", "milk", "mom", "please", "shirt", "sleep", "sorry", "thank you", and "water" [18]. Each signal is captured using dynamic windowing (see Section 6.1.2) and sampled 60 times (see Section 6.2).

In this experiment, for each $C = 4, 5, 6, \ldots, 20$, we randomly select C classes (gestures) from the 20 available and use the C selected classes as the new vocabulary. The training set then consists of $10C$ signals selected at random with the remaining 10C signals used for the testing set.

Table 3 and Figure 16 show the average accuracy of k-NN DTW as the vocabulary size increases. Each value is the average of 30 replications of the experiment for that value of C, with 95% confidence intervals provided.

# Gestures	Accuracy (%)	# Gestures	Accuracy (%)	# Gestures	Accuracy (%)
4	96.5 $\pm$ 1.2	10	96.8 $\pm$ 1.0	16	95.7 $\pm$ 1.4
5	97.0 $\pm$ 1.1	11	96.4 $\pm$ 1.2	17	95.9 $\pm$ 1.2
6	94.6 $\pm$ 1.3	12	96.5 $\pm$ 1.4	18	95.6 $\pm$ 1.4
7	95.9 $\pm$ 1.3	13	97.1 $\pm$ 1.1	19	96.0 $\pm$ 1.2
8	96.6 $\pm$ 1.2	14	96.6 $\pm$ 1.1	20	96.0 $\pm$ 1.0
9	96.7 $\pm$ 1.2	15	95.0 $\pm$ 1.8		

Table 3: The accuracy of the k-NN DTW classifier as the vocabulary size increases.

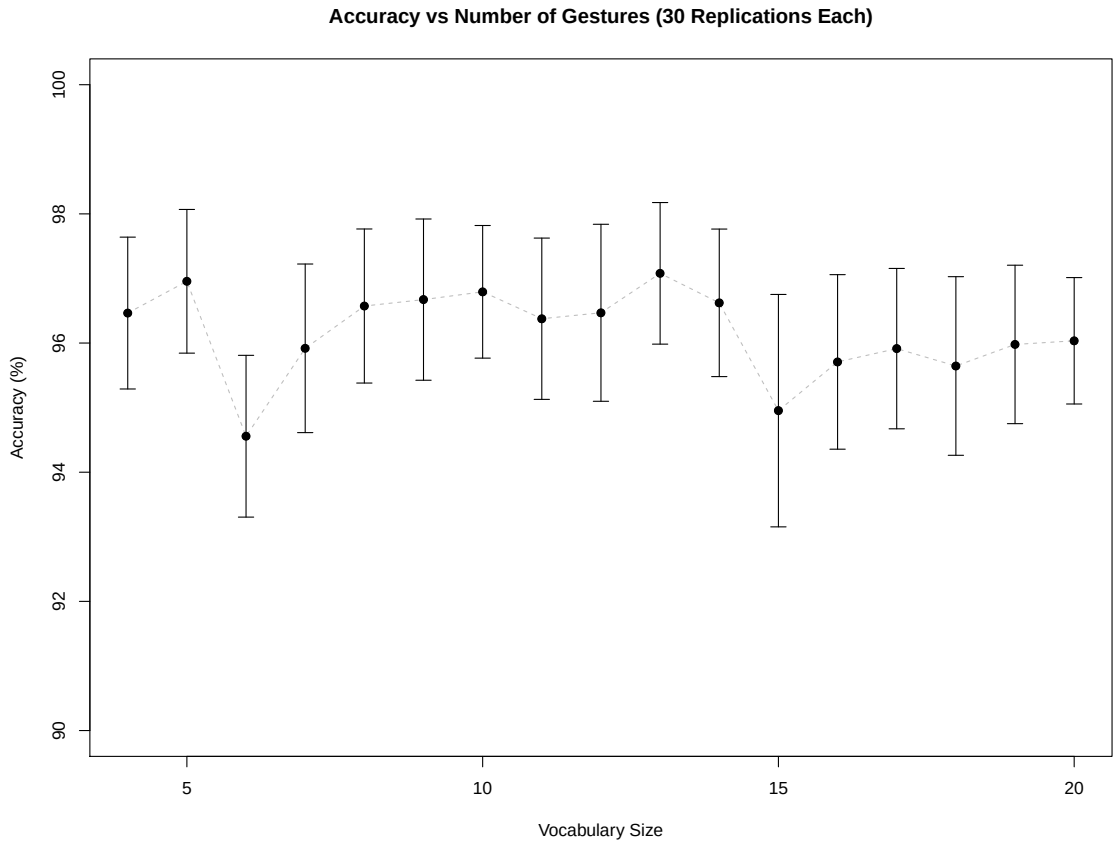


Figure 16: Accuracy of k-NN DTW versus vocabulary size.

7.4 DTW Window Size

As explained in Section 5.2.4, the efficiency of DTW can be improved by only computing values within a window around the shortest path. However, there is no known algorithm for immediately selecting the ideal window size. Instead, the process involves trial and error, testing each window size. The purpose of this experiment is to investigate this computational efficiency, as a function of window size, versus classification accuracy.

This experiment has a similar setup to that in Section 7.1. There are 10 ASL gestures with 20 captured signals per gesture. Thus there are 200 signals that can be used for training. For each gesture, we select 10 signals at random for training, with the remaining 10 for testing. Here,

for the purpose of consistently comparing replications across window sizes, we use the same 30 randomly-selected test sets across all window sizes.

For this experiment we use a range of $[1, 50]$ for the DTW window size. Figure 17 depicts the classification accuracy and the computation time of k-NN DTW versus DTW window size. Timing results were obtained by executing the prototype on a 3.2 GHz MacBook Pro. Figure 17 shows that execution time increases with increasing window size, as expected. Although omitted from the figure, there is no change in accuracy beyond a window size of 13, but computation time continues to increase. For this reason, we use 13 as our implemented window size. This ensures that DTW with windowing will not affect accuracy while being more computationally efficient then traditional DTW.

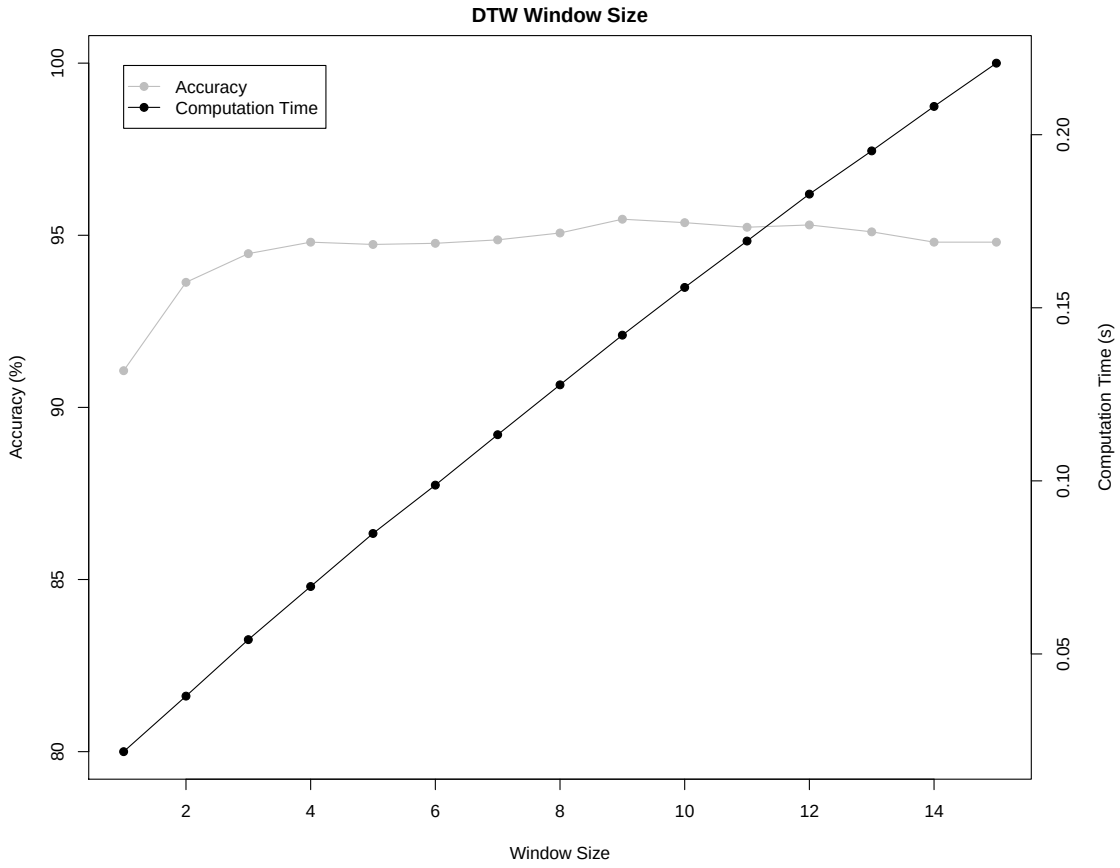


Figure 17: Classification accuracy and execution time versus DTW windowing.

8 Future Work

The work in this thesis focuses on two classification algorithms: LDA and k-NN. The DTW-based k-NN, once widely used for speech recognition, provides high classification accuracy for our work. However, because HMMs are now generally the classification algorithm of choice in speech recognition, future work will investigate HMMs in an effort to further improve the accuracy of our system. As noted in Section 3, HMMs seem to be the most common classification algorithm used for recognizing ASL gestures, particularly for video based systems.

Increasing the vocabulary size of our system would increase the efficacy of using the classifier in the real world. Our prototype uses only 20 ASL gestures for proof of concept, but the system should be expanded to better represent the large ASL dictionary. As discussed in Section 3, other research has investigated use of phonemes to construct ASL gestures for classification. Our work can also be extended to use phonetic construction for gestures.

Finally, we are interested in using syntax identification for improving classification of ASL words and phrases. This approach would use rules for constructing sentences in the English language, along with context-specific information, to choose words most likely to fit properly in a sentence. This feature could prove very beneficial in increasing the overall accuracy, and therefore usability, of our classification system.

9 Conclusion

The goal of this work was to develop a practical real-life application to help remove barriers in communicating via ASL. A key aspect of our work was to develop a system that works effectively in real-time on mobile systems, and that requires only streamlined, readily-available hardware. Previous work in this area is often restricted by the need for bulky hardware, probes, or cameras. The Myo device, a small portable consumer electronic device, acts as the input source for our classifier because of its sleek design, wide availability, and relatively inexpensive cost. The Myo

has on-board accelerometer, gyroscope, orientation, and EMG sensors, providing appropriate data needed for accurately classifying ASL gestures.

These different sensors are then used to form one aggregate signal consisting of data concatenated from all four sensors, which is then sent to our system for classification. Using the dynamic windowing approach we are able to use accelerometer x values to identify the start and end of ASL gestures. To facilitate this identification process, we require the user to place their arm in a resting position at their side between multiple gestures.

Our work focuses on two classification algorithms: linear discriminant analysis and k-NN. In our experiments, k-Nearest Neighbors using dynamic time warping produces accurate results using a fast training process. Thus k-NN with DTW was our choice of classification algorithm for the prototype. The computationally expensive DTW distance metric is improved by using a windowing system, facilitating real-time use for classification.

In experimentation, our system achieves classification accuracies of 94 – 98% in identifying ASL gestures using real and synthetic data. Further research is necessary to improve overall classification accuracy measurements while increasing the number of ASL gestures recognizable by our system.

References

[1] Naomi S Altman. “An introduction to kernel and nearest-neighbor nonparametric regression”. In: *The American Statistician.* 46.3 (1992), pp. 175–185.

[2] Alex vijay raj Amalaraj. *Real-time hand gesture recognition using sEMG and accelerometer for gesture to speech conversion*. Masters Thesis. San Francisco State University, 2015. URL: http://sfsu-dspace.calstate.edu/bitstream/handle/10211.3/141234/AS362015ENGRA43.pdf.

[3] Helene Brashear et al. *Using multiple sensors for mobile sign language recognition*. Georgia Institute of Technology, 2003. URL: http://www.cc.gatech.edu/~thad/p/031_10_SL/iswc2003-sign.pdf.

[4] National Institute on Deafness and Other Communication Disorders. *American Sign Language*. June 2015. URL: https://www.nidcd.nih.gov/health/american-sign-language.

[5] Mark Difranco. *GUI without the G: Going Beyond the Screen with the Myo Armband*. Mar. 2014. URL: http://developerblog.myo.com/gui-without-g-going-beyond-screen-myotm-armband/.

[6] Aly A. Farag and Shireen Y. Elhabian. *A Tutorial on Data Reduction: Linear Discriminant Analysis*. Oct. 2008. URL: http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid437773.pdf.

[7] Kirsti Grobel and Marcell Assan. “Isolated sign language recognition using hidden Markov models”. In: *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation.* Vol. 1. IEEE. 1997, pp. 162–167.

[8] Apple Inc. *Swift. A modern programming language that is safe, fast, and interactive.* 2016. URL: https://developer.apple.com/swift/.

[9] Waleed Kadous et al. *GRASP: Recognition of Australian sign language using instrumented gloves*. 1995. URL: https://www.researchgate.net/publication/2830297_GRASP_Recognition_of_Australian_Sign_Language_Using_Instrumented_Gloves.

[10] Thalmic Labs. *Myo Gesture Controlled Armband*. Sept. 2013. URL: https://www.myo.com/.

[11] Jeroen F Lichtenauer, Emile A Hendriks, and Marcel JT Reinders. "Sign language recognition by combining statistical DTW and independent classification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 30.11 (2008), pp. 2040–2046.

[12] Ross E Mitchell et al. "How many people use ASL in the United States? Why estimates need updating". In: *Sign Language Studies.* 6.3 (2006), pp. 306–335.

[13] Daisuke Nishikawa et al. "EMG prosthetic hand controller using real-time learning method". In: *IEEE International Conference on Systems, Man, and Cybernetics.* Vol. 1. IEEE. 1999, pp. 153–158.

[14] Cemil Oz and Ming C Leu. "American Sign Language word recognition with a sensory glove using artificial neural networks". In: *Engineering Applications of Artificial Intelligence.* 24.7 (2011), pp. 1204–1213.

[15] Zhihua Qiao, Lan Zhou, and Jianhua Z Huang. "Effective linear discriminant analysis for high dimensional, low sample size data". In: *Proceeding of the World Congress on Engineering*. Vol. 2. Citeseer. 2008, pp. 2–4.

[16] The Deaf Society. *INSIGHTS INTO AUSLAN*. July 2015. URL: http://deafsocietynsw.org.au/documents/SignLanguage1Handouts.pdf.

[17] Thad Starner, Joshua Weaver, and Alex Pentland. "Real-time american sign language recognition using desk and wearable computer based video". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 20.12 (1998), pp. 1371–1375.

[18] ASL University. *Basic ASL: 100 First Signs*. URL: http://www.lifeprint.com/asl101/pages-layout/concepts.htm.

[19] William Vicars. *Sign Language Alphabet*. 2007. URL: http://www.lifeprint.com/asl101/topics/wallpaper1.htm.

[20] C. Vogler and D. Metaxas. "Parallel hidden Markov models for American sign language recognition". In: *The Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. 1999, 116–122 vol.1. DOI: 10.1109/ICCV.1999.791206.

[21] Christian Vogler and Dimitris Metaxas. “A framework for recognizing the simultaneous aspects of american sign language”. In: *Computer Vision and Image Understanding*. 81.3 (2001), pp. 358–384.

[22] Zahoor Zafrulla et al. “American Sign Language Recognition with the Kinect”. In: *Proceedings of the 13th International Conference on Multimodal Interfaces*. ICMI ’11. Alicante, Spain: ACM, 2011, pp. 279–286. ISBN: 978-1-4503-0641-6.