Touchless Interaction with a Smartphone: EMG-Based Sleeve for Cyclists

Cristina Matonte
IT University of Copenhagen
calo@itu.dk

Tomas Miseikis
IT University of Copenhagen
tmis@itu.dk

Kyra Tijhuis
IT University of Copenhagen
kyrt@itu.dk

ABSTRACT
In this paper we present the development of a system that allows cyclists to control the music player on their smartphone in a touchless manner. A custom built electromyography (EMG) based sleeve is used to recognize 6 hand gestures that correspond to specific commands as input to the music player. The sleeve uses two channels to measure electrical potential generated in the Extensor Digitorum and Flexor Carpi Ulnaris muscles when performing gestures. To increase the accuracy of classifier a number of features were used, including Minimum value, Mean, Median, Modified Mean Absolute Value 2, Zero Crossing, and Successive Difference Sum. The performance of the system was evaluated using 6 subjects, who were using exercise bike while performing the various gestures. Overall, the highest accuracy of 68.00% was achieved using Neural Network classifier with a Low-Pass Filter for data smoothing.

Existing Technologies
One way to tackle the problems described above is to rely on touchless interaction. The touchless interaction paradigm aims at allowing users to interact with technology through touchless interaction and body movements [20]. There are many different ways in which this can be achieved. A popular approach is eye tracking, where eye movement is monitored and used as an input mechanism to a computing device [5]. Using human voice is another way of touchlessly interacting with a computing device, where different features of voice, such as pitch and volume, can be used as an input [15, 22]. Using wireless technology approaches, it is possible to extract gesture information from ambient RF signals (e.g., TV transmission) and use it to control your device [17]. Proximity sensors [3] and ultrasound chips [21] provide a basic way to touchless interaction, where users can move their hands towards and away from the device to execute a specific command. Approaches based on cameras use image processing techniques to track movements and gestures, and translate them to specific instructions as input to a device [25, 23, 31]. Finally, various body sensors can be utilized to track arm, feet, or head movements, and these can be used as input [4, 42, 10]. One of the devices that uses these sensors is the Myo, a bracelet that is worn around the thickest part of the forearm. The Myo has several EMG sensors, a gyroscope, an accelerometer and a magnetometer. These sensors enable the Myo to detect both hand gestures and the position of the forearm [36].

Limitations
The problem with these approaches is that most of them still require physical contact with the device, as well as eye focus. For example, if we consider a smartphone as our main device, then the mentioned proximity, ultrasound, and camera-based techniques would require users to hold their smartphone in one hand and perform a gesture with the other hand. Eye tracking approaches present an even more challenging situation, since the device must be held close to the face, and the users eyes must focus only on the device. Voice-based techniques are not a very good solution if the device will be used in public places, since it might present a privacy risk, as well as disturb other people. For these reasons, in this project we are focusing on body sensors, and more specifically sensors that measure hand movements. Since hands are one of the most flexible parts of our bodies we can consider a large variety of different gestures.
EMG Sensors
One of the approaches in recognizing hand gestures is to use electromyography (EMG) sensors. EMG sensors measure electrical potential that is generated in a muscle during its contraction. Researchers who seek to utilize the EMG technology have been mostly focused on the medical field. Majority of the applications focus on serving the needs of people with injuries or various disabilities, since EMG can help in diagnosing muscular disorders [14], controlling prosthetics [18, 30], and the measuring of mental stress [40].

The Need for Touchless Interaction
Although a lot of research has been conducted and applications have been developed based on EMG, there are still many areas of our life that could benefit from this technology. The idea of using EMG technology for enabling touchless interaction provides a perfect foundation for applications that allow users to use their computing devices, even if their hands are otherwise occupied. This is a quite common scenario for people who drive or cycle to school or work, and at the same time try to use their smartphone. Naturally, this can be dangerous, since using a smartphone while cycling is extremely distracting. Furthermore, cyclists have less control of their bicycle when they only have one hand on the handlebars for a prolonged period of time. According to the Cycling Embassy of Denmark, each year 17,500 cyclists are treated at the hospital for cycle-related injuries [7]. A number of these injuries are due to distractions while accepting a phone call or trying to change a song on the music player. In a report the U.S. Department of Transportation presents the results of an extensive research on “The Impact of Hand-Held And Hands-Free Cell Phone Use on Driving Performance and Safety-Critical Event Risk” [11]. Their research shows that the odds ratio of having an accident while trying to locate or answer your phone is very high, namely 3.65. Furthermore, looking at the phone while driving increases your chance of an accident by 1.73. In contrast, using hands-free equipment reduces the chances of getting into an accident to 0.79. Therefore we can conclude that, while driving or cycling, it is not the using of the smartphone that is most dangerous. It is the looking at or holding of a smartphone that increases the risk of an accident significantly.

Our Solution
In this work we present an EMG-based arm sleeve prototype that supports touchless interaction with a smartphone while the user is cycling. The sleeve has 6 electrode connectors that target muscles in both the anterior and the posterior compartment of the forearm. These muscles are monitored by two EMG sensors, one for each compartment of the forearm. With this sleeve the electrical activity of muscles can be measured in real time, and the data is then sent via Bluetooth to the smartphone, where the recognition of the hand gestures is performed. Using this sleeve we are able to recognize five hand gestures and one activation gesture. Our chosen scenario deals with controlling a music player on a smartphone while cycling, so each different gesture corresponds to a specific command, namely play/pause song, switch to next song, switch to previous song, volume up, and volume down. The activation gesture is there to prevent unintentional gestures being acted upon by the system.

Firstly, we present a short overview on other work that is in some way related to our solution. In later parts, the details of the developed system are provided, followed by an evaluation and a discussion of the system.

RELATED WORK
In this section we present related work that involves a short background on electromyography (EMG), and discussions on electrodes, classification, and pattern recognition.

Electromyography
Electromyography measures muscle activity as electrical potential between a ground electrode and a sensor electrode. It is possible to recognize diverse gestures, since different gestures produce distinct EMG signals. EMG can measure signals either inside the muscle (invasive EMG) or from the skin (surface EMG). For more detailed information on electromyography, see Merletti et al. [19]. Invasive EMG is not usable in this case, since it requires the electrodes to be inserted into the muscle. Surface EMG, even though less accurate, is better suited for our project.

Electrodes and Their Placement
There are mainly two types of surface electrodes used when measuring EMG signals, namely wet and dry. Wet electrodes in general provide a better signal-to-noise ratio and accuracy [26] than dry electrodes. This is due to the fact that wet electrodes are sticky and ensure better contact with the skin. However, there are studies showing that dry electrodes can perform as good as [12] and sometimes even better than wet electrodes [28]. In aforementioned paper a quantitative comparison is presented, which shows that interference experienced by dry electrodes can be 40dB less than that experienced by wet electrodes.

One of the disadvantages of wet electrodes is that they may induce some discomfort when applying to and removing from the skin. Furthermore, traditional wet electrodes are supposed to be used only once, so after every use they are meant to be thrown away and replaced with a new set. The main issue with dry electrodes is conductivity problems. The impedance between a dry electrode and the skin can be affected due to hairs lifting the sensor or differing moisture levels in the skin [39]. Furthermore, it is harder to keep dry electrodes in a fixed position, especially when on the move. The electrodes would need to be used together with some kind of stretchable and compressive fabric that would help to keep them in the same place.

In general, it is challenging to apply electrodes on the exact same place on the body, therefore the position of the electrodes may vary greatly from one day to the next. If the electrode placement differs a lot from the initial placement (when the classifier model was built), this can affect the accuracy of gesture recognition. There have been experiments [39] that measures the effects of minor (1-3 mm) and major (1-2 cm) variations in the placement of electrodes. Findings showed
that minor variations had no impact; however, major displace-
ments required re-training of the recognition models.

The electrodes themselves can be made from different types
of material. Silver and copper are commonly used [9]. A
different possibility is using conductive fabric, which is made
by plating nylon fabric with silver [34].

Besides the mentioned problems with wet and dry electrodes
there are other factors that may affect the quality of the EMG
signal. Individual physiological differences in people, such
as differences in arm lengths and widths, can make it chal-
lenging to put electrodes at the proper positions.

Concluding, dry electrodes seem to be most fitting in our
case, since we want to be able to use the electrode multi-
ple times. To ensure conductivity between the electrodes and
skin, and to make sure that the electrode placement does not
differ too much between usages, we use a compressive sleeve
to which we attach the electrodes. When we combine conduc-
tive fabric with a compressive sleeve [2] the resulting device
satisfies all our requirements.

Classification and Pattern Recognition

Previous work on classifying patterns based on EMG input
has been done using different machine learning techniques
such as Neural Network and Support Vector Machine.

An experiment using a Selective Desensitization Neural Net-
work (SDNN) [16] tries to overcomes some limitations of the
multilayer perceptron and is able to approximate some func-
tions by using only a few training data samples. In this ex-
periment, test subjects were asked to execute each of the avail-
able gestures for 2 seconds, then repeat this nine times. After-
wards a cross validation was performed over the total number
of data samples, in order to calculate the classification rate.
The total average classification rate over the subjects and the
available gestures was 95.26%.

A different experiment using Support Vector Machine [27]
has set a 3-phases laboratory experiment: Hands-free Fin-
ger Gestures, Hands Busy Finger Gestures and Controlling a
Portable Music Player Application. The first two phases were
used for training and testing of the predictive model. The
third phase used this training data for real-time control and
classification. The real-time data-processing consisted of a
basic signal processing, the generation of feature vectors and
further classification using a Sequential Minimal Optimization
of Support Vector Machines. The results of the exper-
iment varied from an accuracy of 65% to 91%, depending on
whether the user received some immediate feedback to cor-
rect their gestures or not. The best results were achieved on
phase 2, when the user got some feedback; phase 3 got an
average accuracy of 86%.

However, there are also many other classification algorithms
in use [4, 14, 39]. Therefore we will compare different clas-
sification algorithms, in order to find out which algorithm
works best for our solution.

THE SYSTEM

In this section we present the design of our system.

Figure 1 displays a high-level overview of the system. The
user wears the sleeve that is connected to the Arduino board.
The Android phone connects to the Arduino board via Blue-
tooth and waits for input data. After the user performs an
activation gesture, the Arduino sends the input data. The An-
droid device reads and preprocesses this data, extracts a num-ter of features and runs it through the classifier. Based on the
result of the classification, the device will perform the corre-
sponding action on the media player. When training data is
obtained the setup varies a little. The input data is sent to a
computer instead of a smartphone, and the activation gesture
is not required. Instead the user uses a button on the computer
to start a recording.

Wearable Device and Arduino

The wearable device is made by sewing strips of conductive
fabric onto a compressive arm sleeve. The strips of conduc-
tive fabric are the electrodes that measure the muscle activ-
ity. The compressive arm sleeve ensures that the electrodes
make contact with the skin at all times and also keeps them
in place. To connect the electrodes to the sensor we added
snap buttons to the sleeve. These metallic snap buttons con-
nect the electrodes on the inside with the cable leads on the
outside. Figure 2 shows the wearable device with the cables
connected.

Figure 2. Sleeve with the cable leads connecting the sensor to the elec-

Figure 3. System overview

The Android phone connects to the Arduino board via Blue-
tooth and waits for input data. After the user performs an
activation gesture, the Arduino sends the input data. The An-
droid device reads and preprocesses this data, extracts a num-ter of features and runs it through the classifier. Based on the
result of the classification, the device will perform the corre-
sponding action on the media player. When training data is
obtained the setup varies a little. The input data is sent to a
computer instead of a smartphone, and the activation gesture
is not required. Instead the user uses a button on the computer
to start a recording.

Wearable Device and Arduino

The wearable device is made by sewing strips of conductive
fabric onto a compressive arm sleeve. The strips of conduc-
tive fabric are the electrodes that measure the muscle activ-
ity. The compressive arm sleeve ensures that the electrodes
make contact with the skin at all times and also keeps them
in place. To connect the electrodes to the sensor we added
snap buttons to the sleeve. These metallic snap buttons con-
nect the electrodes on the inside with the cable leads on the
outside. Figure 2 shows the wearable device with the cables
connected.

Figure 2. Sleeve with the cable leads connecting the sensor to the elec-

Figure 3. System overview

The Android phone connects to the Arduino board via Blue-
tooth and waits for input data. After the user performs an
activation gesture, the Arduino sends the input data. The An-
droid device reads and preprocesses this data, extracts a num-ter of features and runs it through the classifier. Based on the
result of the classification, the device will perform the corre-
sponding action on the media player. When training data is
obtained the setup varies a little. The input data is sent to a
computer instead of a smartphone, and the activation gesture
is not required. Instead the user uses a button on the computer
to start a recording.

Wearable Device and Arduino

The wearable device is made by sewing strips of conductive
fabric onto a compressive arm sleeve. The strips of conduc-
tive fabric are the electrodes that measure the muscle activ-
ity. The compressive arm sleeve ensures that the electrodes
make contact with the skin at all times and also keeps them
in place. To connect the electrodes to the sensor we added
snap buttons to the sleeve. These metallic snap buttons con-
nect the electrodes on the inside with the cable leads on the
outside. Figure 2 shows the wearable device with the cables
connected.
connect both grounding cables to the same electrode, therefore we only need 5 instead of 6 strips of conductive fabric. Two of the strips are supposed to target the Extensor Digitorum muscle in the posterior compartment of the forearm. This muscle is used to extend the fingers (from a curled to a stretched position). However, since not all arms are the same, it might occur that other muscles are targeted as well. The muscle that is targeted by the other EMG sensor is the Flexor Carpi Ulnaris, which flexes and adds the hand [35]. Again, it might be possible that other muscles are targeted as well.

Figure 3. Sleeve worn inside out. The upper electrodes target the Extensor Digitorum muscle, the lower electrodes target the Flexor Carpi Ulnaris

The electrodes have a width (the direction perpendicular to the muscle fibres) of 4.5 cm and a height of 1 cm. With these dimensions they follow the recommendations made by the SENIAM initiative [8]. To ensure conductivity between the skin and electrode, it is important to wet the electrodes before usage, otherwise the readings are not usable. It is also important to make sure that the cable leads connecting the electrodes to the sensors are in a fixed position, since movement of the cables will increase the signal-to-noise ratio.

Figure 4. Arduino board with all connected components

The rest of the device-side components can be seen in Figure 4. The red, blue and black cables connect the electrodes to the EMG sensors. These sensors then amplify, rectify and smooth the data, so it can be immediately put through to the Arduino RedBoard [33]. The data is sent wirelessly to the smartphone using a Bluetooth module [32].

Gestures
Initially, 16 different gestures were tried out; however, some of them were complex to perform or were not easily recognizable by the system. For the purpose of this project a minimum of five gestures is required, so that each gesture corresponds to a particular action in the media player. The original 16 gestures were grouped into smaller sets to try out which combination provided the best accuracy. Also, a few bikers were asked about the gestures’ intuitiveness and ease of use while cycling. Based on these two sets of information, six gestures were chosen to be used in the system. Five of them will be used to control the media player, namely Snap, Side Left, Gun Right, Wrist Down, and Wrist Up; while the sixth gesture, Fist, is used as an activation gesture. The main goal of having an activation gesture is to avoid interfering with the activities the user is performing. Fist was chosen as an activation gesture, since it is easy to perform it for a period of time. The minimum period of time that the activation gesture needs to be performed is 150ms. During this period the values need to be above 150mV for the muscle in the posterior compartment, and above 50mV for the muscle in the anterior compartment.

The selected gestures can be seen in Figure 5.

Figure 5. Gestures

The graph in Figure 6 shows that the different gestures have distinct forms.

Training Data
In order to capture the training data to be used for building the classifier, we invited 7 people to put the sleeve on and perform the gestures illustrated in Figure 5. Each person performed every gesture at least 20 times. The recording frequency was set to 30 samples per second and each recording was performed for 1 second. A snapshot of the recorded data is presented in Figure 7. The first number is the measurement from the electrodes on the posterior compartment of the forearm, whereas the second number is the measurement from the
Figure 6. Graphs of measurements of the six different gestures, with time (in s) on the x-axis and electrical activity of the muscle (in mV) on the y-axis. The red line shows the measurements from the electrodes on the posterior part of the forearm, the blue line from the electrodes on the anterior part of the forearm. These numbers show the electrical potential in mV and can be between 0 and 1024. The third attribute is the timestamp of the measurement in nanoseconds. The last attribute describes the gesture and the first name letter of the person that is associated with the reading.

Figure 7. Snapshot of the recorded data

Feature Extraction

For each recording we generate a set of features. First of all, we convert each recording from a sequential learning problem into a traditional classification problem by turning the electrical potential measured by both sensors into vectors of size 30. This is represented as \((x_1, \ldots, x_{30}, y_1, \ldots, y_{30})\), where \(x\) is data received from the first sensor and \(y\) is data received from the second sensor. Additionally, we generate 12 more features, which were chosen based on prior work and research by people that dealt with EMG sensors and applications [24, 6]. The chosen features are Maximum Value, Minimum Value, Median, Mean, Variance, Standard Deviation, Root Mean Square, Modified Mean Absolute Value 1, Modified Mean Absolute Value 2, Wilson Amplitude (WAMP), Zero Crossing, and Successive Difference Sum. These features are generated for the measurements from both sensors, therefore in the end the total number of features is 24. The features are described more in detail below.

- **Maximum Value (MAX).** Gives us the highest electrical potential read by the sensors in the given recording.
- **Minimum Value (MIN).** Gives us the lowest electrical potential read by the sensors in the given recording.
- **Median.** Median gives us the middle value of the EMG signal within a recording and is calculated by sorting the numbers in increasing order and simply taking the middle number.
- **Mean (MEAN).** The mean is an easy way to see the average muscle contraction levels for different gestures. It is defined as
  \[
  \text{MEAN} = \bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n. 
  \]  
- **Variance (VAR).** The variance allows us to measure the power of the signal and is calculated as
  \[
  \text{VAR} = \frac{1}{N-1} \sum_{n=1}^{N} (x_n - \bar{x})^2, 
  \]  
  where \(x_n\) is the \(n\)th sample, \(N\) is the total number of samples, and \(\bar{x}\) is the mean of the samples.
- **Standard Deviation (SD).** SD allows us to see how spread out the numbers of the EMG signal are. SD is calculated by taking the square root of variance
  \[
  SD = \sqrt{\text{VAR}}.
  \]  
- **Root Mean Square (RMS).** In the context of EMG, root mean square is related to the constant force and non-fatiguing contraction [24] and can be expressed as
  \[
  \text{RMS} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} x_n^2}, 
  \]  
- **Modified Mean Absolute Value 1 (MMAV1).** MMAV1 extends calculation of the mean by applying weighting window \(w_n\). It is defined as
  \[
  \text{MMAV1} = \frac{1}{N} \sum_{n=1}^{N} w_n x_n, 
  \]  
  where
  \[
  w_n = \begin{cases} 
  1, & \text{if } 0.25N \leq n \leq 0.75N \\
  0.5, & \text{otherwise.}
  \end{cases}
  \]  
- **Modified Mean Absolute Value 2 (MMAV2).** MMAV2 is similar to MMAV1, however continuous weighting window function is used to improve smooth window. MMAV2 is calculated as
  \[
  \text{MMAV2} = \frac{1}{N} \sum_{n=1}^{N} w_n x_n, 
  \]
Where

\[ w_n = \begin{cases} 
1, & \text{if } 0.25N \leq n \leq 0.75N \\
\frac{4n}{4(nN)}, & \text{if } 0.25N > n \\
\frac{1}{N}, & \text{if } 0.75N < n.
\end{cases} \]

- **Wilson Amplitude (WAMP).** WAMP calculates the amount of times that the change in EMG signal amplitude between two adjacent readings exceeds the given threshold, 50\(\mu\)V in our case. It is related to the muscle contraction level and is defined as

\[ WAMP = \sum_{n=1}^{N-1} f(|x_n - x_{n+1}|), \tag{7} \]

where

\[ f(x) = \begin{cases} 
1, & \text{if } x \geq \text{threshold} \\
0, & \text{otherwise}.
\end{cases} \]

- **Zero Crossing (ZC).** ZC is the amount of times the signal passes the zero amplitude axes. In our case, the signal received from EMG sensors is amplified, rectified, and smoothed, thus ZC can be easily computed as

\[ ZC = \sum_{n=1}^{N} f(x_n), \tag{8} \]

where

\[ f(x) = \begin{cases} 
1, & \text{if } x = 0 \\
0, & \text{otherwise}.
\end{cases} \]

- **Successive Difference Sum (SDS).** Calculates the summed difference between two adjacent EMG signal readings in a given recording. It is defined as

\[ SDS = \sum_{n=1}^{N-1} (x_n - x_{n+1}). \tag{9} \]

### Building the Classifier

Before starting to build the classifier, some considerations were made regarding the features. As it was described in previous section, we use two vectors of size 30 for the data received from the sensors and we generate 12 extra features for both sensors. Therefore, if we consider using all the features, the built classifier will end up with 84 attributes. However, using all the features does not necessarily produce the highest accuracy. The complicated part regarding feature extraction is to figure out which combination of features works best and produces the classifier with the highest accuracy. In order to find that out, we decided to generate all possible combinations of the features and build a classifier for every combination.

For building classifiers we used Weka [38], which contains machine learning algorithms for data mining tasks. We tested 5 different classifiers, namely K*, KNN, Neural Network, C4.5, and Bayes. All the classifiers were trained using 10-fold cross-validation. Furthermore, default Weka parameters were used to train the classifiers, with exception of Neural Network classifier, where training time was increased from 500 milliseconds to 1 second. After building classifiers for all the possible feature combinations, we found that 6 particular features produce the highest accuracy. These features are **Minimum Value**, **Mean**, **Median**, **Modified Mean Absolute Value 2**, **Zero Crossing**, and **Successive Difference Sum**. The built classifiers and their accuracy using these 6 features can be seen in Table 1. The three most accurate classifiers, namely KNN, Neural Network, and K*, were chosen to be used for evaluation.

### Android Application

An Android Application that communicates with the Arduino board was developed. It consists of a simple user interface that allows users to connect to the Arduino board and can be seen in Figure 8. The application will establish communication with the Arduino board through a Bluetooth socket and then create an Asynchronous Task that listens for incoming data. When the user performs the activation gesture, the Arduino will send a ‘start’ message to the Android device and it will play a short notification sound. The user performs the action after the notification sound has played and the data is sent to the Android device for approx. 1s. The received data is preprocessed, the features discussed in subsection Feature Extraction are extracted, and an arff file [37] is created. This file is then passed to the Weka library to be evaluated with the trained model using the KNN classifier. Weka proceeds to classify the data received into a gesture mapped to an action to control the media player, which will immediately be executed. See Table 2 for a relation between the gesture performed and the action executed by the Media Player on the Android device.

As it can be seen in Figure 8, a field for the participant name has been added for evaluation purposes. The participant name should consist of the action that is intended to be performed and the subject name. This information is later used to store files on the device containing the preprocessed data, so it can be reused later to evaluate the performance of different classifiers. Also a visual notification is displayed providing feedback on what gesture and action was interpreted by the device. This can be seen at the bottom part of the figure.
EVALUATION AND RESULTS

Experimental Setting
The perfect setting for evaluating the system would involve subjects cycling on their chosen route while trying to interact with the system. This, however, was difficult to achieve with the current system setup, because it would be hard to carry the Arduino board with all the connected components while cycling. Instead, we decided to use stationary exercise bikes for evaluation, as they are the closest alternative that resembles a real cycling scenario.

In total 6 subjects, 3 female and 3 male, were asked to participate in the evaluation. Overall, the experiment took 30 minutes per subject, where 20 minutes were given to familiarize the subject with the sleeve, gestures, and the commands the gestures correspond to (see Table 2), and the last 10 minutes were dedicated to the actual control of the music player by using hand gestures. In these last 10 minutes subjects were asked to perform every gesture exactly 10 times. The first 3 subjects received visual and audio feedback after every gesture during the entire experiment, so they would know whether their performed gesture was interpreted correctly or not. This allowed them to adjust the way they performed the gesture if the system did not interpret it correctly. The last 3 subjects were only given audio feedback, making it harder for them to know which action was performed.

Furthermore, the generated test samples for gestures were saved locally on the smartphone. These were later used to test the accuracy of different classifiers.

Results
In this section we present the results of our experiment.

Table 3 shows the general accuracy for each test subject using different classifiers without data smoothing, as well as the average accuracy for all of the subjects. The first 3 subjects, who were given audio and visual feedback, have a better accuracy than the ones who only got audio feedback. From this we can argue that the more familiarized the user is with the correct execution of the gestures, the better the accuracy will be. On average K* performs the best, closely followed by KNN. Neural Network seems to perform the worst. This differs from the results that were achieved during the training phase (see Table 1), where KNN had the best accuracy. These findings suggest that the model built in the training phase does not necessarily perform the best in a real scenario.

After noticing that the model used was not having such a high accuracy as in the training phase, we decided to check which gestures were being classified incorrectly. As it can be seen in Table 4, Side Left, Snap and Wrist Down are quite accurate. However, Gun Right tends to be confused with all other gestures, and Wrist Up is most of the times confused as Gun Right. Looking at Figure 6, we can see that the pattern for Wrist Up and Gun Right are slightly similar, which could justify such confusion. However, based on the patterns these does not seem to be a clear relation between Gun Right and the other gestures.

Table 5 shows the averaged accuracies of different classifiers when applying various smoothing techniques. We can see that using Weighted Low-Pass Differential (WLPD) [41] with a window width of 5 decreases the accuracy of all three classifiers. Using Moving Average (MA) [29] with window size of 30 decreases the accuracy for KNN and K* classifiers; however, it increases the accuracy of Neural Network classifier by 6%. We can notice a similar trend when using a Low-Pass Filter (LPF) [13] (with smoothing = 17). Here, the accuracy of KNN and K* classifiers decreases, however, the accuracy of Neural Network classifier increases by 6.67%. In the end, it seems like using Neural Network classifier with LPF to smooth the data allows us to achieve the highest accuracy.

Table 6 shows the average gesture accuracies when using different classifiers with Low-Pass Filter for data smoothing. We can notice that the Side Left and Wrist Down gestures have the highest classification accuracy out of all gestures. This indicates that these particular gestures have unique characteristics and are easily distinguishable in the context of other

<table>
<thead>
<tr>
<th>Subject</th>
<th>KNN</th>
<th>K*</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 1</td>
<td>74.00%</td>
<td>74.00%</td>
<td>68.00%</td>
</tr>
<tr>
<td>Subject 2</td>
<td>84.00%</td>
<td>86.00%</td>
<td>84.00%</td>
</tr>
<tr>
<td>Subject 3</td>
<td>62.00%</td>
<td>68.00%</td>
<td>64.00%</td>
</tr>
<tr>
<td>Subject 4</td>
<td>54.00%</td>
<td>54.00%</td>
<td>46.00%</td>
</tr>
<tr>
<td>Subject 5</td>
<td>50.00%</td>
<td>62.00%</td>
<td>46.00%</td>
</tr>
<tr>
<td>Subject 6</td>
<td>60.00%</td>
<td>44.00%</td>
<td>60.00%</td>
</tr>
<tr>
<td>AVG.</td>
<td>64.00%</td>
<td>64.67%</td>
<td>61.33%</td>
</tr>
</tbody>
</table>

Table 3. Subjects and their corresponding accuracies with different classifiers without data smoothing

<table>
<thead>
<tr>
<th></th>
<th>GR</th>
<th>SL</th>
<th>Snap</th>
<th>WD</th>
<th>WU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR</td>
<td>26.67%</td>
<td>30.00%</td>
<td>3.33%</td>
<td>16.67%</td>
<td>23.33%</td>
</tr>
<tr>
<td>SL</td>
<td>0.00%</td>
<td>95.00%</td>
<td>3.33%</td>
<td>1.67%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Snap</td>
<td>0.00%</td>
<td>8.33%</td>
<td>73.33%</td>
<td>18.33%</td>
<td>0.00%</td>
</tr>
<tr>
<td>WD</td>
<td>0.00%</td>
<td>3.33%</td>
<td>0.00%</td>
<td>96.67%</td>
<td>0.00%</td>
</tr>
<tr>
<td>WU</td>
<td>58.33%</td>
<td>13.33%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>28.33%</td>
</tr>
</tbody>
</table>

Table 4. Confusion matrix using KNN classifier
Table 5. Averaged accuracies of classifiers when applying different smoothing techniques

<table>
<thead>
<tr>
<th></th>
<th>KNN</th>
<th>K*</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG. WLPD</td>
<td>62.33%</td>
<td>59.33%</td>
<td>57.00%</td>
</tr>
<tr>
<td>AVG. MA</td>
<td>63.33%</td>
<td>62.00%</td>
<td>67.33%</td>
</tr>
<tr>
<td>AVG. LPF</td>
<td>61.00%</td>
<td>62.33%</td>
<td>68.00%</td>
</tr>
</tbody>
</table>

Table 6. Gesture accuracies with different classifiers using LPF for data smoothing

<table>
<thead>
<tr>
<th></th>
<th>KNN</th>
<th>K*</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snap</td>
<td>63.33%</td>
<td>83.33%</td>
<td>83.33%</td>
</tr>
<tr>
<td>Side Left</td>
<td>80.00%</td>
<td>81.67%</td>
<td>85.00%</td>
</tr>
<tr>
<td>Gun Right</td>
<td>45.00%</td>
<td>45.00%</td>
<td>55.00%</td>
</tr>
<tr>
<td>Wrist Down</td>
<td>86.67%</td>
<td>65.00%</td>
<td>95.00%</td>
</tr>
<tr>
<td>Wrist Up</td>
<td>30.00%</td>
<td>31.67%</td>
<td>21.67%</td>
</tr>
</tbody>
</table>

DISCUSSION

During the development of the system we encountered many challenges. Firstly, the performance of the built sleeve varied greatly with different people. For some, the sleeve worked immediately after being put on, for others it was non-responsive. We soon found out that wetting the conductive strips (with water) would increase conductivity between skin and electrode immensely. With this trick we could make the sleeve work for everybody, for a short period of time at least. Because after 15-20 minutes the electrodes would dry, making the sleeve non-responsive again. Since compressive sleeves are meant to keep skin warm and dry (from Sub Sports Amazon page: “...Sub RX Arm Sleeves are highly breathable and excellent at wicking moisture away from the skin...”), the moisture we put on the electrodes would be quickly removed from the skin by the compressive sleeve. A possible solution would be to put a waterproof backing between the electrode and the compressive sleeve. This would help to keep the electrode moisturized. As for the difference between people, we postulate that, since some people have dry skin, they need to wet the electrodes before usage, while others, whose skin has higher moisture levels, can use the sleeve immediately. Furthermore, people with hairier forearms seemed to have more problems, most likely due to the hairs pushing the conductive strips off the skin. Lastly, our built sleeve became less compressive over the time, due to constant application and removal of the sleeve. This seemed to affect the readings from the sensors, especially for people with smaller and slimmer forearms, since the compression of the sleeve was not enough to ensure contact between skin and electrodes.

The wires connecting the sensors and electrodes presented another considerable issue. Even a slight movement of the wires seemed to greatly affect the readings from the sensors. In order to avoid this interference wires had to be placed in a fixed position, which seemed to help slightly. Naturally, in a real cycling scenario a person is constantly moving, making the received data very noisy and most likely unreliable.

For gathering the training and test data, as well as evaluation of the system, the number of subjects was small. This is due to the fact that it takes quite some time to prepare the system, introduce the subject to the sleeve, describe the different gestures, and do the gestures a number of times, while making sure the sleeve is still wet enough. More subjects would be needed to build better classifier and perform a more extensive evaluation.

Moreover, our evaluation was very constrained and limited, since exercise bikes were used to simulate cycling on the road. However, our system targets cyclists that use real bicycles and are often exposed to different conditions. Because of this, a better evaluation of the system is needed that involves subjects using real bicycles and cycling through different paths.

The results we got from our evaluation were not as good as we initially hoped for. Our highest achieved accuracy of 68.00% for classifying gestures shows that there is still much work to be done. It seems necessary to find alternative gestures to Gun Right and Wrist Up, as they have the lowest accuracy of being classified correctly.

Regarding future work, there are a number of changes that could be made to the current setup of our system. Instead of using an activation gesture, we could use a fixed length sliding window. This way the users would not need to worry about performing the activation gesture before performing the actual gesture. Nevertheless, this approach would present different problems, for example related to the battery consumption, since the data would be continuously streamed to the smartphone for processing.
Furthermore, the Android application could support a way for users to customize which gesture corresponds to which particular command. This way, different profiles could be set up for different applications, allowing users to use the sleeve for more than just media playback control.

**CONCLUSION**

The objective of this project was to develop EMG-enabled sleeve for cyclists, which would allow them to control the music player on their smartphone using hand gestures. One activation gesture and five control gestures were chosen to be used in the system. The evaluation of the system showed that by using a Neural Network classifier with a low-pass filter for smoothing, it is possible to achieve 68.00% accuracy in correctly classifying different gestures. The average accuracy of the classifier was primarily reduced by two gestures, which were classified correctly only 55.00% and 21.67% of the time, respectively. The other three gestures had a much higher accuracy, namely 85.00%, 83.33%, and 95.00%.

These results show that it is possible to recognize some gestures using simple EMG sensors and a custom built compressive sleeve with conductive fabric strips stitched to it. Nevertheless, more research and work needs to be done to find a better combination of gestures, so that the accuracy of the classifier can be increased. Furthermore, different types and combinations of sensors, conductive fabrics, compressive sleeves, features, classifiers, and preprocessing algorithms need to be tested to find the best solution.

**REFERENCES**


