Haaga-Helia
University of Applied Sciences Ltd.

Sensor Fusion with Python for Machine Learning and Data Modelling

Taito Halonen

Master's thesis
Degree Programme in
Information Systems
2022

Haaga-Helia
ammattikorkeakoulu Oy

Abstract
28 October 2022

Author(s)
Taito Halonen

Degree
Master of Business Administration

Thesis title	**Number of pages + number of appendices**
Sensor Fusion with Python for Machine Learning and Data Modelling	40

Wearable sensors are becoming more common and the market for health monitoring type of consumer tech devices is growing fast. The aim of this project was to discover if a proof-of-concept type of machine learning data model can be created to recognize and classify human knee position and movements using data from sensors and machine learning practices like Support Vector Machine (SVM) algorithm. A functional model created through this study could serve as a foundation for future development of smart therapeutic leg support device. Additional use case could be to use the resulting model for subject gait analysis.

The sensor devices used in the project were wireless advanced heart sensors with accelerometer and gyroscope. Instead of monitoring heart rate, the devices were repurposed as surface electromyography (EMG) monitors which also provide accelerometer data. The devices are relatively inexpensive when compared to previously available medical professional sensor equipment. The project team generated and collected the data material based on agreed forms of subject movements.

The data processing and modelling was done using Python and open-sourced libraries as the primary development platform. Both types of raw data (ACC and EMG) provided by sensors were pre-processed and combined in effort to achieve model recognition result via sensor fusion.

The result was a collection of signal processing and analysis functions and a multiclass SVM model implementation for classification.

Conclusion was that the original data signal quality needs to be high before additional processing and feature extraction. The data processing methods proved to require considerable amount of manual work for adjustments for new movements. Based on the implementations the increased model recognition precision of sensor fusion proved challenging due to data synchronization.

Keywords
Machine Learning, SVM, Python, EMG, human movement

Contents

1 Introduction

The life expectancy of population in the EU and in the Nordic countries continues to be in the rise (Eurostat 2022a). Statistically it is also evident that the total population is ageing, especially in Finland (Eurostat 2022b). Combining the statistics, it can also be expected that the amount of various health issues will increase. One of the most common issues are mobility related health problems and various prosthetic operations like prosthetic knee operations where Finland is also ranking high in EU countries (OECD Health Statistics 2019). Partly due to these facts the health tech industry is a booming trend. Various self-measurement devices are commonplace, and people are increasingly interested in monitoring themselves and receiving analysis and suggestions often based on machine learning algorithms.

This project was initiated to discover if a data model can be created to recognize and classify knee position and movements using data from sensors and machine learning practices like Support Vector Machine (SVM) algorithm. A functional model created through this study could serve as a foundation for future development of intelligent leg support devices where recognized movement would cause support to be distributed accordingly. Additional use case could be to use the model for subject gait analysis.

In this study I will research if a mobile wireless personal consumer heart rate monitor can be used to collect muscle electromyographic (EMG) data instead of electrocardiogram (ECG) signal as well as accelerometer data. After pre-processing the combined data would be used for machine learning model training with the purpose of creating a functional algorithm that could be further developed and used in gait analysis or an intelligent leg support device.

To reach the desired outcomes, the following research questions were selected to guide the research.

RQ1 How well is the SVM model suited for leg movement classification?

RQ2 What are the requirements for suitable data for training machine learning model on leg movement recognition?

RQ3 What type of pre-processing of data is required for SVM?

RQ4 How could wellness-monitor type of sensors be repurposed to provide sufficiently detailed data?

The model programming will be done in Python language and recognized Open-Source libraries

This thesis consists of the following chapters: In chapter two I introduce the objectives and background of the study as well as the scope and objectives. Chapter three is the theoretical framework, where I review the concepts of related human biomechanics recording, analysis, related research, and the tools such as programming languages and machine learning. Chapter four is the methodology chapter, where I describe the project team, the selected methods and work processes. In chapter five I introduce the implementations and outcomes, where I will go through the implementation of the selected methods and what outcomes were obtained. Chapter six presents the conclusions, where I describe how well the project reached the defined objectives, other conclusions after the project and what were the lessons learned as well as suggestions for further development.

2 Background and objectives

In this chapter I will explain what the background and objectives of this thesis are, what are the research questions, how the project is scoped and what are the expected outcomes of the project.

The challenge is that Finnish population is aging and that causes different kind of a health problems. Statistics in Figure 1 show that the population in the EU is living longer and in Figure 2 growing increasingly older in general.

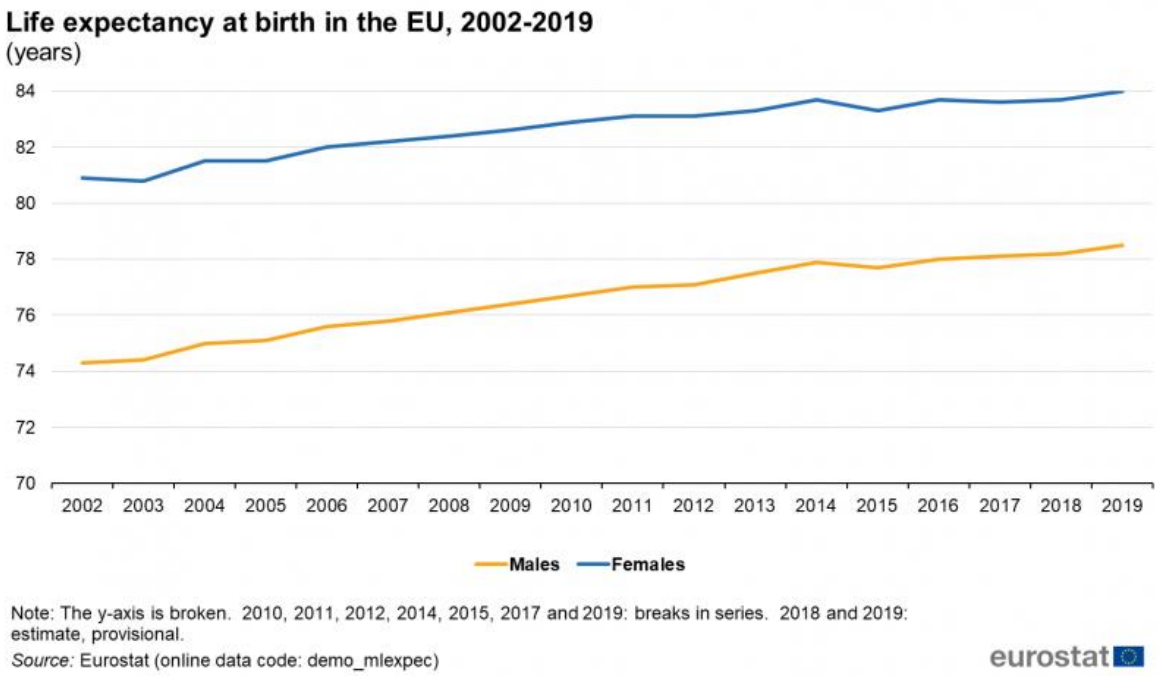

Figure 1. Life expectancy at birth in the EU, 2002–2019 (Eurostat 2022a).

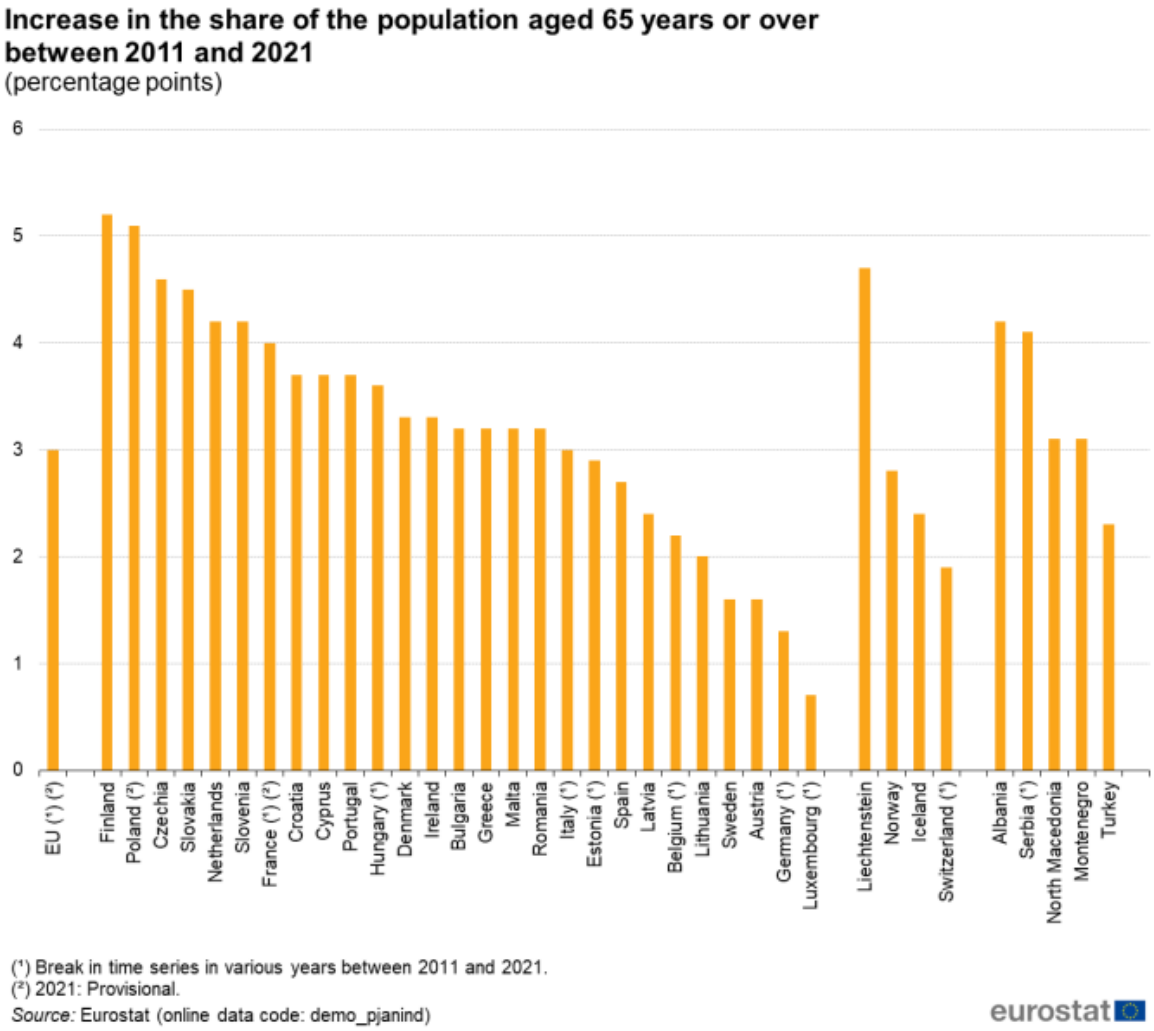

Figure 2. Increase in the share of the population aged 65 years or over between 2011 and 2022. (Eurostat 2022b)

The combination of growing aging populace leads to increase in national general health expenditure and operations. Knee replacement surgeries have been increasing and as can be seen in Figure 3, Finland is one of the top countries in recent years with the number of operations performed.

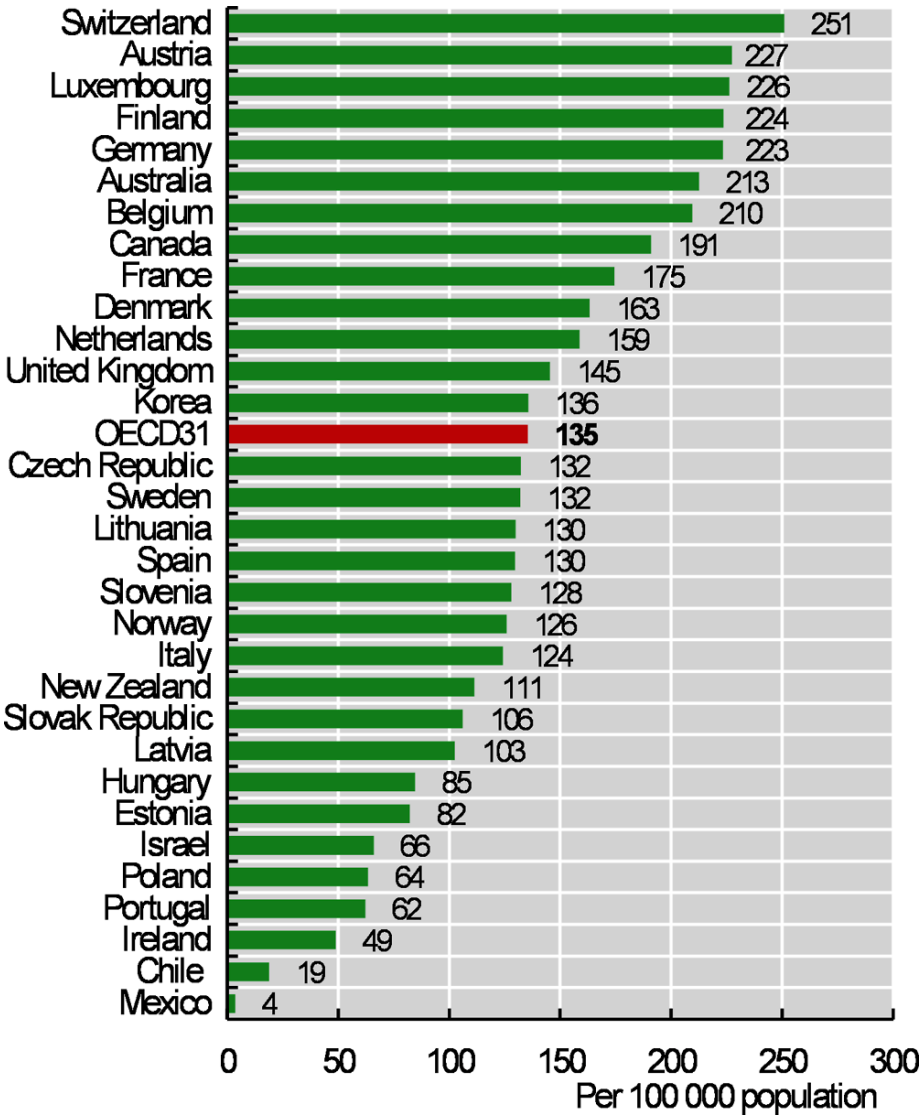

Figure 3. Knee replacement surgery, 2017 (or nearest year). Source: OECD Health Statistics 2019.

The amount of arthroplasty both for knee and hip replacements have been on the rise especially in OECD countries in the 21st century. Research suggests that there is a strong correlation in the amount of arthroplasty operation utilization with GDP, obesity, and health expenditure. (Pabinger, Lothaller, & Geissler, 2015; Pabinger & Geissler 2014.)

According one market research the size of global total knee arthroplasty market in 2021 was US $8 7339,2 Mn and is expected in 2028 to rise to $11 484,8 Mn (Coherent Market Insights 2022).

As the amount of knee replacement operations is increasing, there will be growing market for preventive and therapeutic care to postpone or avoid such an operation.

As machine learning and artificial intelligence (AI) are coming increasingly more common place in everyday life, the technology and resources needed for implementation are now more accessible for anyone than ever before. Previously, for example, the resources for collecting medical data like EMG and ECG has been dependent on often expensive and non-consumer available equipment.

This project was initiated to discover if a data model can be created to recognize and classify knee position and movements using data from sensors and machine learning practices like Support Vector Machine (SVM) algorithm. A functional model created through this study could serve as foundation for future development of intelligent leg support devices where recognized movement would cause support to be distributed accordingly. Additional use case could be to use the model for subject gait analysis.

The sensor devices used in the project were wireless Beat2Phone electrocardiogram (ECG) device built by VitalSignum (2019). Instead of monitoring heart rate, they are repurposed as surface electromyography (EMG) monitors which also provide accelerometer data. The devices are relatively inexpensive when compared to previously available medical professional sensor equipment. The project team generated and collected the data material based on agreed forms of subject movements.

The data processing and modelling was done using Python and open-sourced libraries as the primary development platform. Both types of raw data (ACC and EMG) provided by sensors were pre-processed and combined in effort to achieve better machine learning modelling result via sensor fusion.

2.1 Expected outcomes

The goal of the project was to achieve the following outcomes:

- The method of processing and qualifying valid data.
  The steps required for collecting, validating, and pre-processing raw data to be suitable quality to be used in the selected machine learning model. Valid data means that noise is filtered away and that data is consistent for classification purpose.
- A functioning data model capable of classifying provided sensor data with high probability.
- Reusable code for possible further development of an intelligent leg support device prototype

- A presentable correlation between accelerometer and EMG data. Combined data is expected to yield a more accurate classification result than using just a single data source for model training.

To reach the desired outcomes, these research questions were selected to guide the research.

RQ1 How well is the SVM model suited for leg movement classification?

RQ2 What are the requirements for suitable data for training machine learning model on leg movement recognition?

RQ3 What type of pre-processing of data is required for SVM?

RQ4 How could wellness-monitor type of sensors be repurposed to provide sufficiently detailed data?

2.2 Scope

The model programming was done in Python language with recognized Open-Source libraries. Deep learning neural networks are omitted from the scope since the project outcome is aimed for embedded devices. The ECG sensor devices will be the primary data collecting source. At least a single working data model was expected.

The model is based on a limited number of predefined movements.

Updating and maintenance support of the data model, documentation or code is not part of scope. Any commercial or stakeholder pitching is out of scope. Scaling the model to larger environment or additional sensor devices is out of scope. Additional or comprehensive list of movements are out of scope.

3 Theoretical framework

In this chapter I will introduce the common biomechanical concepts of human body muscle activity monitoring, type of sensors used, signal capturing and processing techniques. What research has been done in this area and how machine learning has been introduced in the modern research. What type of programming tools are used in the research and why. The visual representations of different signal processing methods can be found in chapter 5.1. Signal processing.

3.1 EMG and signal processing

Electromyogram (EMG) is the electric signal associated with contraction of a muscle and its area of study is called electromyography. The signal is affected by the different variables of muscle activity such as shortening or lengthening of the muscle, fatigue, rate of tension build-up and reflex activity. (Winter 2005, p. 229.)

EMG data can be acquired with surface electrodes (sEMG) on skin or invasive electrodes directly to the muscle itself when finer detail signal is needed or for the study of deeper located muscles. The latter type of electrodes is usually used for clinical research and not frequent use due to user discomfort. The signal resolution and accuracy are dependent on signal frequency and the length of the segment. (Spiewak et al. 2018.)

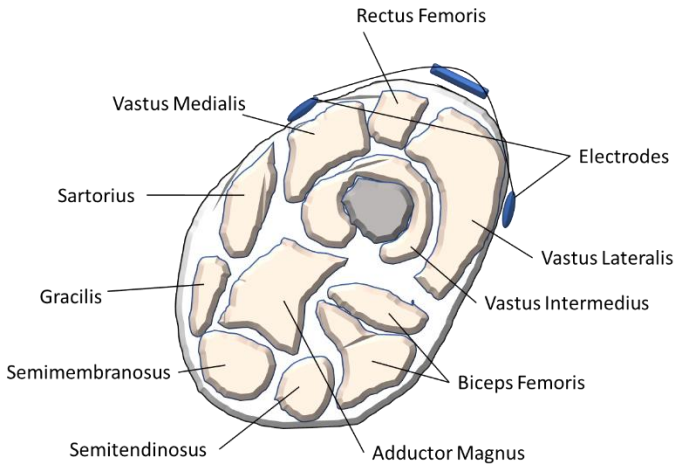

Figure 4. Figure of midthigh underlying muscles and suggested placement of surface electrodes. (Adapted from Winter 2005, p. 245)

The EMG signal (Figure 5 upper part) has typically rapid variation and can contain noise artifacts from other sources such as other muscles, electrode abrasion or outside sources.

Once EMG signal has been recorded it needs to be cleaned and prepare for further analysis using following techniques:

- Full wave rectification
- Bandpass filtering
- Linear envelope

The raw EMG signal has a zero mean value due to recording with biological amplifier with a low-frequency cut off around 10 Hz. Full-wave rectifying generates the absolute value of EMG meaning positive values. The visual examination of rectified signal gives reasonable indication of muscle contraction. Linear envelope is the moving average following the EMG trend and is obtained by applying a low pass filter to the rectified data. The proper amplitude unit for EMG signal is millivolts. (Winter 2005, p. 249.)

The lower part of Figure 5 shows signal frequency distribution. The low frequency 0-10Hz peaks are likely movement artifacts and bandpass filtering with low and high pass filter values would be applied to desired frequency range to remove noise from the signal.

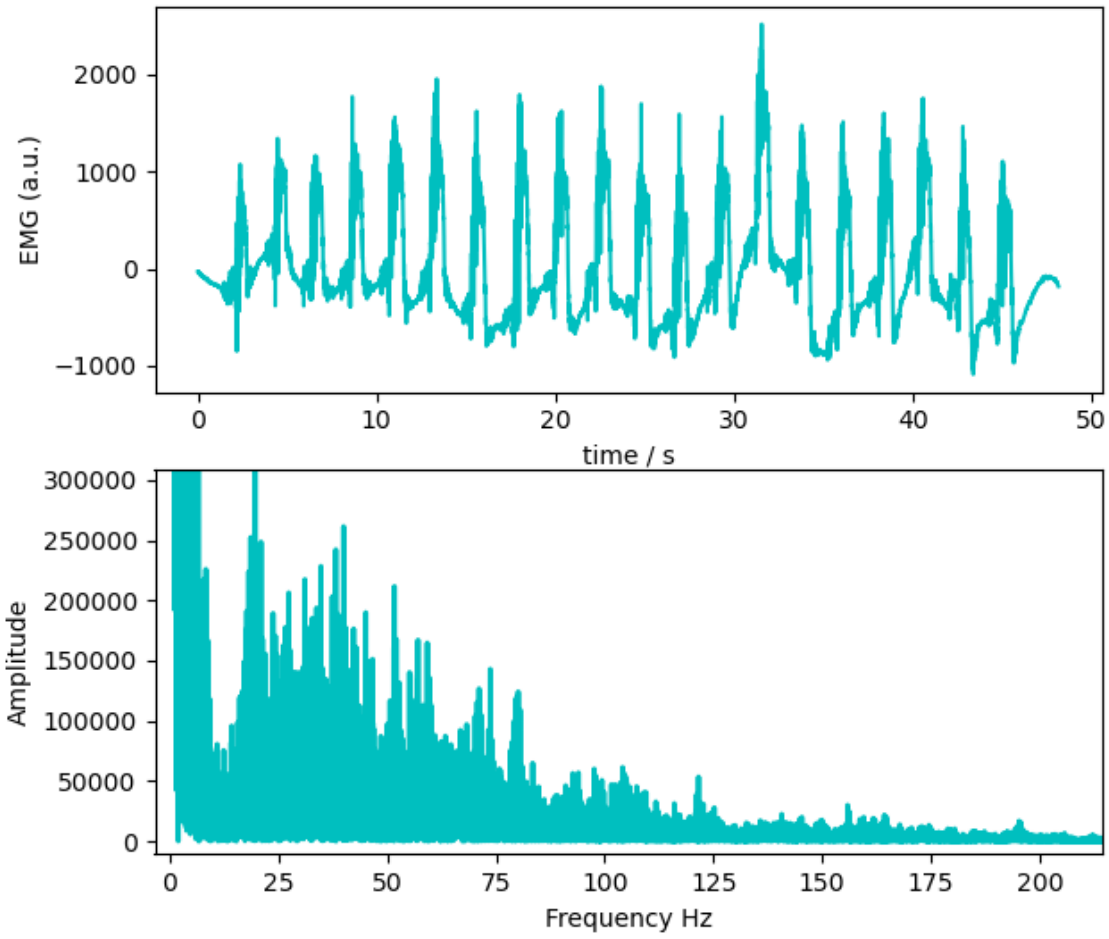

Figure 5. EMG signal portrayed in time and frequency domains.

The cleaning of data and highlighting relevant information into data structures is commonly called Feature extraction. The three main categories of feature extraction are time domain, frequency domain and time-frequency domain. (Spiewak et al. 2018.)

The time domain features do not require additional transformation and are commonly used in EMG pattern recognition. The computation of time domain features is based on the amplitude of the input signal producing frequency, duration, and the amplitude of waveform. Common methods include integrated EMG, which is similar to linear envelope, and Root Mean Square (RMS). (Spiewak et al. 2018.)

Frequency domain feature extraction methods are based on estimated signal power spectral density and require more computing resources and time (Spiewak et al. 2018).

After pre-processing the data may need to be normalized if the EMG activity is to be compared with sessions, different individuals or between muscles. According to Halaki and Ginn (2012), there are four common methods for EMG signal normalization:

1. Maximum (peak) activation levels during maximum contractions

Maximum voluntary isometric contraction (MVIC) is measured using a reference test, where monitored individual repeats a manual muscle test to produce a maximum contraction for reference. The test is proposed to be repeated at least three times separated at least two minutes to reduce the effect of fatigue. The obtained maximum value from the processed signal of the muscle test is then used as a reference for EMG signal normalization.

The challenge with this method is how to decide which test will actually yield the MVIC result, and the results would vary in different muscles. If there is no standardized set of tests for MVIC there is a possibility of getting results that exceed 100% MVIC EMG levels, especially if movements contain rapid or forceful contractions. Incorrectly measuring the MVIC can cause systematic error in the normalization process. The MVIC is individual and cannot be generalized between different subjects. (Halaki & Ginn 2012.)

2. Peak or mean activation levels obtained during the task under investigation

A subject is monitored during the activity and observed peak or mean levels are used

for the EMG normalization. This method is shown to reduce variability between individuals as the reference value is relative to the task and not the maximum capacity of the muscle. The value cannot be used to compare with other tasks, muscles, or individuals. (Halaki & Ginn 2012.)

3. Activation levels during submaximal isometric contractions

In cases where MVIC cannot be obtained if the subject has physical limitations or restrictions such as pain or risk of injury in performing the activity, a submaximal contraction level can be used. Common tests for obtaining submaximal contraction levels are holding limb against gravity or holding different loads. This method is also subject specific and does not fit for comparing different individuals or muscles. (Halaki & Ginn 2012.)

4. Peak to peak amplitude of the maximum M-wave (M-max)

M-wave means the signal of stimulating a motor nerve proximal to a muscle causing it to contract. M-max is the maximum of M-wave achieved by increasing the amplitude of stimulation until the peak amplitude does not increase anymore. The amplitude of M-max is used for muscle specific normalization. The problem using this method is the unreliability of M-max repeatability due to among other things tasks performed, time and muscle length. (Halaki & Ginn 2012.)

3.2 Machine learning and classification

Machine learning (ML) is a type of artificial intelligence (AI) where algorithms using historical data as input allow software applications to become better at predicting outcomes. Typically the types of machine learning is classified in four categories: supervised learning, unsupervised learning, semi-supervised learning and reinforced learning. (Burns 2021.)

The classification of processed signal can be done with the help of machine learning. There are several options and similarly to feature extraction, different classification techniques apply better to specific type of data.

Some common classifiers in this type of research are Bayesian classifiers, decision tree, random forest, neural networks, and support vector machines (SVM).

Bayesian classifier, where system can predict the values of a class of features if a class is known and if the class is not known it can be predicted using Bayes' rule. Bayesian classifiers are often used because of speed, and it can be very useful when used in combination. (Spiewak et al. 2018.)

A neural network is a supervised machine learning model that imitates brain by using a network of neurons. There are three classes of nodes, input, output and hidden in different layers and connections between the type of nodes. The model calculations are based on the nodes which are weighted, and the model has algorithms to adjust the weights. The input nodes receive an activation pattern which moves toward the output layer through hidden layers in between and the input activation is multiplied by the links it spreads. In the end the input activation is summed and if it exceeds the nodes threshold the node becomes activated. The neural network model is efficient with large datasets with nonlinear features but needs to be monitored during model training not to over or underfit. The model is computationally heavy and usually requires a lot of storage. (Spiewak et al. 2018.)

Support Vector Machines (SVM) are a supervised learning method in machine learning, and they are used in classification, regression, and outlier detection. (Scikit-learn 2022a)

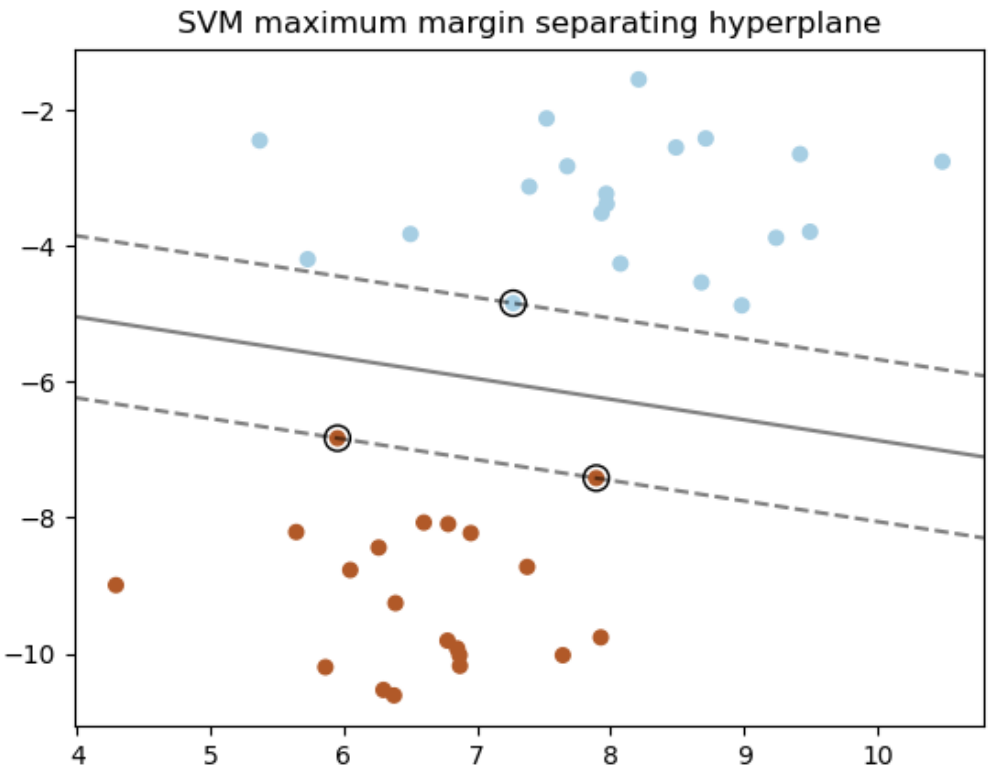

Figure 6. SVM classifier with linear kernel hyperplane example. (Scikit-learn 2022a)

An example of an SVM (Figure 6) where a linear line, hyperplane, separates two classes of data points and the dotted lines (support vectors) indicate the maximum margin between hyperplane and any class of data points. There are different types of Support Vector Classifiers (SVC) that can be used to for non-linear and multidimensional data with help of kernel trick, where the classification is based on different kernel function for calculation. Some common kernels used in SVM are polynomial, Radial Basis Function (rbf) and sigmoid. The best choice depends on the data and other parameters applied in the model.

3.3 Wearable sensors and related research

Additional sensors can be used to measure human movement. Traditionally movement has been captured with camera. More recently gyroscope and accelerometer type of sensors have become common in consumer electronics such as smartphones and fitness monitors. Wearable sensors, that are worn on body, can also be mobile units with integrated gyroscope and accelerometer sensors (Badawi et al. 2020). The additional positional data from these sensors should allow better movement recognition when combined with EMG data.

Inertial sensors with accelerometer and gyroscope were used in a study by Kobsar and Ferber (2018) where these wearable sensors and machine learning were used to track movement patterns of subjects with osteoarthritis to identify changes in gait patterns. Using four sensors placed on subject lower back and leg, accelerometer data was gathered, processed, and then machine learning model one-class support vector machine (OCSVM) was used for the gait pattern recognition of the subject and help to compare changes after physical exercise therapy intervention.

In another research article by Badawi, Al-Kabbanyh & Shaban (2020) the human movement activity recognition and different methods using sensor fusion of different wearable surface EMG and inertial sensors. The study used a recently created large database The Human Gait Database (HuGaDB) by Roman Chereshnev (2017) with various gesture and activity data and applied several different machine learning methods to compare and find the best fit for each purpose and claimed to reach very high results.

3.4 Programming languages

In general, the programming language of choice in terms of data science and software engineering is often Python due to its open-source nature, general versatility, proficiency in machine learning tasks and overall popularity. Python has a wide array of free opensource libraries and tools built for specific tasks and can be easily extended. (Python 2022) Another increasingly popular choice for data science and analytics is R language which is specifically developed for data statistics and analysis as well as representation.

For strictly numerical computation the MATLAB programming language is preferred by scholars, but this is a proprietary programming language meaning licensing costs for users. Python has open-source libraries that replicate the functionality of the proprietary MATLAB. Other open-source options to MATLAB as programming language with similar functionality are Julia and Octave (Popuri & Gabbert 2017).

The main open-source Python library used for machine learning is scikit-learn which contains tools for predictive data analysis and is based on other necessary libraries such as NumPy (Python scientific computation), SciPy (Python fundamental scientific computing algorithms) and matplotlib (data visualization).

An example of specific Python advanced open-source software package called MNE-Python, which is specifically designed for brain electrical activity magnetoencephalogram (MEG) and electroencephalogram (EEG) data analysis tasks. In their article published in

Methods, Alexandre Gramford et al (2013) provide a thorough description of the algorithms, capabilities such as preprocessing, visualization and integrations with Python libraries. The MNE-Python software package is developed with support of from various laboratories to allow the use of best practices in analysis.

4 Methodology

This chapter describes the project team and selected methodology and processes used.

The core team consisted of project lead with high subject matter expertise on integrated circuits and signal processing, and two junior programmers, one using Python as the programming platform and the other using MATLAB. Using two different programming languages and algorithms the team aimed to achieve validation of the applicability of data as similar outcomes were expected and obvious errors could be discovered and remedied quickly.

Team worked using light agile software development framework using components from successful agile SCRUM methodology applied to fit the small size of the team. Agile software development methods allow faster development with iterative and incremental practices instead of a traditional monolithic waterfall model (Ambily & Judeth Malliga 2011). During the six months development period the team had biweekly status meetings where targets were set, presented, assessed, revised, or concluded. While setting new weekly goals the effort was also measured towards the overall goal, whether there were sufficient resources to complete those.

The movement data was gathered using VitalSignum's wireless heart rate monitor device Beat2Phone, which has a tri-axis accelerometer in addition to the ECG monitor functionality. The wireless functionality is using Bluetooth version 3 and the data is collected with an accompanying phone application Beat2Phone ECG (VitalSignum 2022) with both iOs and android varieties. The application allows real-time recording and monitoring of ECG signal during exercise.

Sensor placement was decided to be on thigh to register the large vastus lateralis muscle activity (Figure 4).

At least two movements were defined, leg extension while sitting (Picture 1) and moving knee to the side while sitting (Picture 2). The movements were selected to cover focus on both electromyographic and accelerometer data.

Picture 1. Demonstration of leg extension movement.

Picture 2. Demonstration of knee to the side movement.

The subject repeated a single movement for 20 repetitions after starting recording from a smart phone where the Beat2Phone app was activated. Sets were recorded throughout the development.

After recording was completed, the data was transferred from the phone to computer for processing. The two project team members processed the collected data, and each used a different programming platform, MATLAB and Python with their corresponding functions from acknowledged open-source libraries. The reason for two different programming languages was to simultaneously analyse the data and then compare the results to validate the reliability of the used functions and algorithms. As the project progressed, each member had their own focus on data analysis, in Python the focus was on signal

processing and an initial model to use with EMG data and on MATLAB the analysis of the combined accelerometer data. This thesis covers the Python part of the project. The Python development was done on Spyder development environment provided by Anaconda open-source Python distribution (Anaconda 2022).

There were four stages of data processing (Figure 7):

1. Initially the collected EMG and ACC data was transferred from the phone device stored in raw format to PC
2. pre-processing tasks including Fast Fourier Transition analysis, signal filtering, rectifying, windowing, and normalization were applied to prepare the data into valid datasets
3. The pre-processed EMG and ACC data was joined and organized into datasets which were further split into training and test sets. The training datasets were fitted into the SVM model machine learning algorithm
4. The SVM model is fed test data and the algorithm identifies and classifies the type of movement indicated by data

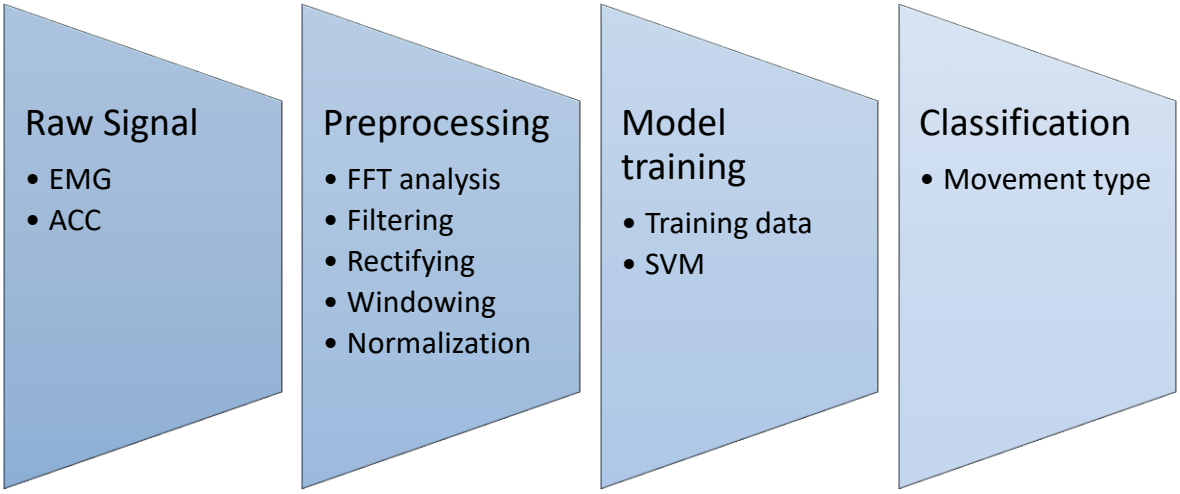

Figure 7. Data processing phases

After the capturing of movement sequence, the accelerometer and electromyographic data was collected from the device in raw format to allow more options for processing and transformation. The python libraries used for signal data processing were NumPy and SciPy.

The accelerometer data has three axis X, Y and Z. The Z is also affected by standard gravity. For processing, the standard gravity was subtracted from the Z-axis data and to reduce complexity all three axis were normalized and combined into a single vector.

The raw electromyographic data is noisy (Figure 10) and requires multiple processing steps. First the data is corrected to zero mean level. Fast Fourier Transformation is used to transform the time domain data into frequency domain and allow to visually evaluate the frequencies and amplitudes in the signal. Based on the frequency analysis the signal is stripped of unwanted frequencies by applying a butterworth high and low band pass filter.

The filtered signal data is full wave rectified to absolute, positive values.

Windowing or EMG enveloping was done by applying a low band pass filter of 10Hz to the rectified signal data.

Normalization was done based on peak activation levels in data using peak detection function.

After peak location data was registered, the signal data was split before and after each peak into datasets consisting of extracted individual moves. The accelerometer data was also split similarly using synchronisation and interpolation to match the time stamps of EMG data.

The python libraries used for machine learning were from Scikit-Learn (Pedregosa et al. 2011) SVM, and oneclass SVM. The SVM with radial based function kernel was selected for classification because of reported good results in similar research and a compromise between simple classification and more complex neural network classifiers.

The model was trained with 80% of the randomized and normalized dataset and tested with the remaining 20%.

5 Implementations and outcomes

The VitalSignum Beat2Phone sensor device was repurposed to be used for registering EMG signal by placing it on subject's thigh to monitor activity in vastus lateralis muscle instead of attaching the device to chest. Since the device sensor belt did not cross the heart, the main primary signal was from the leg muscles.

When activated the Beat2Phone sensor device sends data to Beat2Phone ECG application on a Bluetooth enabled phone where the recording settings can be modified. After the activity recording is completed, the data is transferred to a pc for further processing.

The file format natively supported by the application is European Data Format (EDF) which turned out not to be applicable to this project requirements and instead a special developer version of the application was requested and then the data files could be read in raw format. In the beginning the EDF file format use was attempted with Python MNE library (Gramfort et al. 2013) which had an advanced processing toolset but as it was specifically developed for different types electrogram signals it was fairly soon phased out from the project.

Reading and processing the raw data required writing supporting code functions in both Python and MATLAB. Developing the functions on different programming languages allowed crosschecking to validate the code and ensure the that the raw data was processed correctly and keeping in sync with timestamps.

Once the data input read functions were working the work resumed on pre-processing functions.

5.1 Signal processing

Initial idea was to analyse the entire raw EMG signal, find peaks and then isolate smaller portions of the signal as sections based on time before and after each peak. This section would represent a single gesture, such as a kick. The corresponding time markers of the sections would be applied also to the accelerometer raw data and the same sections would be isolated as well.

First the raw data had to be pre-processed.

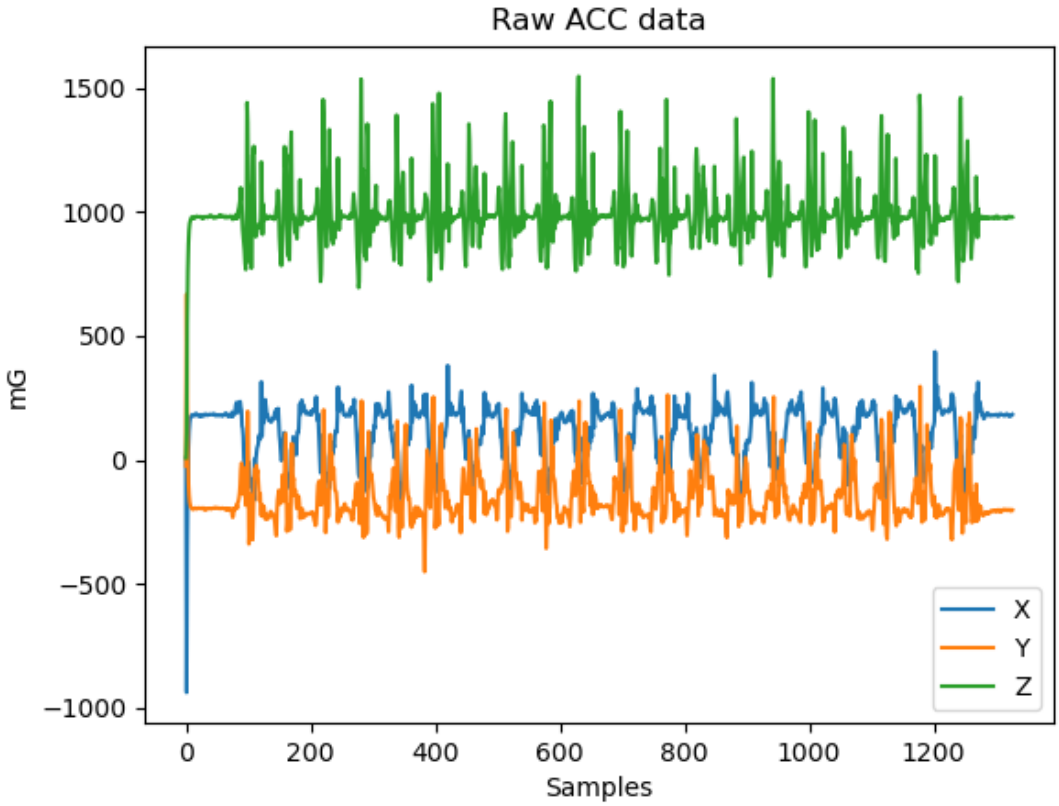

Figure 8. Raw ACC data from movement set

The raw ACC data from single recording in Figure 8 shows the three axes X, Y and Z, and how the standard gravity affects the Z axis of the data. To balance the axis values, the standard gravity value (981 mG) is subtracted from the Z axis values (Figure 9).

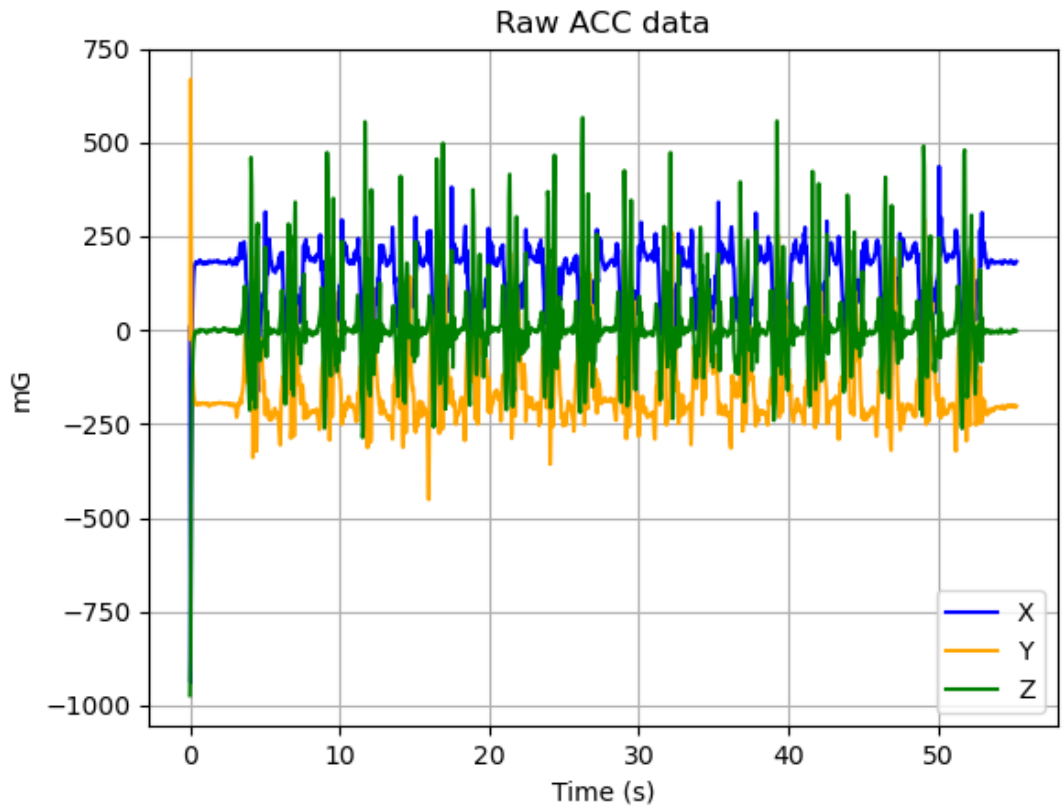

Figure 9. Raw ACC data with Z-axis corrected.

The ACC data is then normalized and the axis combined into single vector:

$$ACC = \sqrt{x^2 + y^2 + z^2}$$

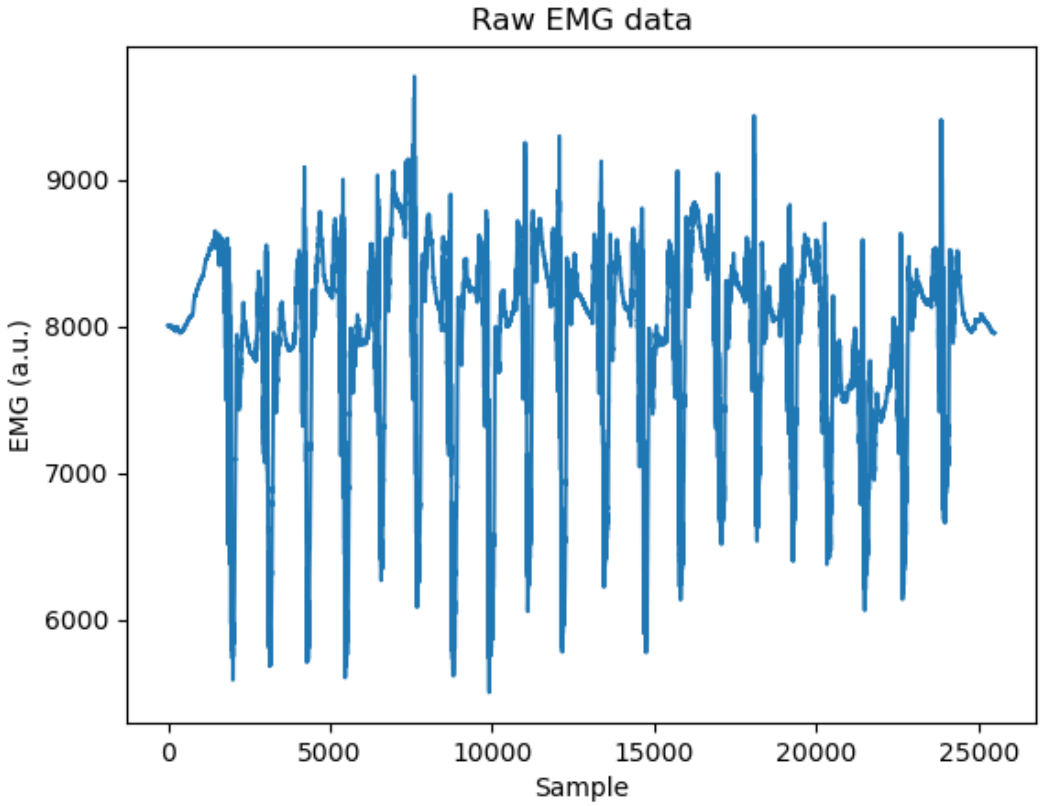

Figure 10. Raw EMG data from movement set.

The EMG data from same activity in raw form is seen in Figure 10. The data shows noise and artifacts from possible sensor belt movement. The raw EMG data processing starts by averaging the signal on zero. The zero level is achieved by calculating the signal average and subtracting it from the signal, see Figure 11.

Python code:

```
emg_correctmean = temp_emg - np.mean(temp_emg)
```

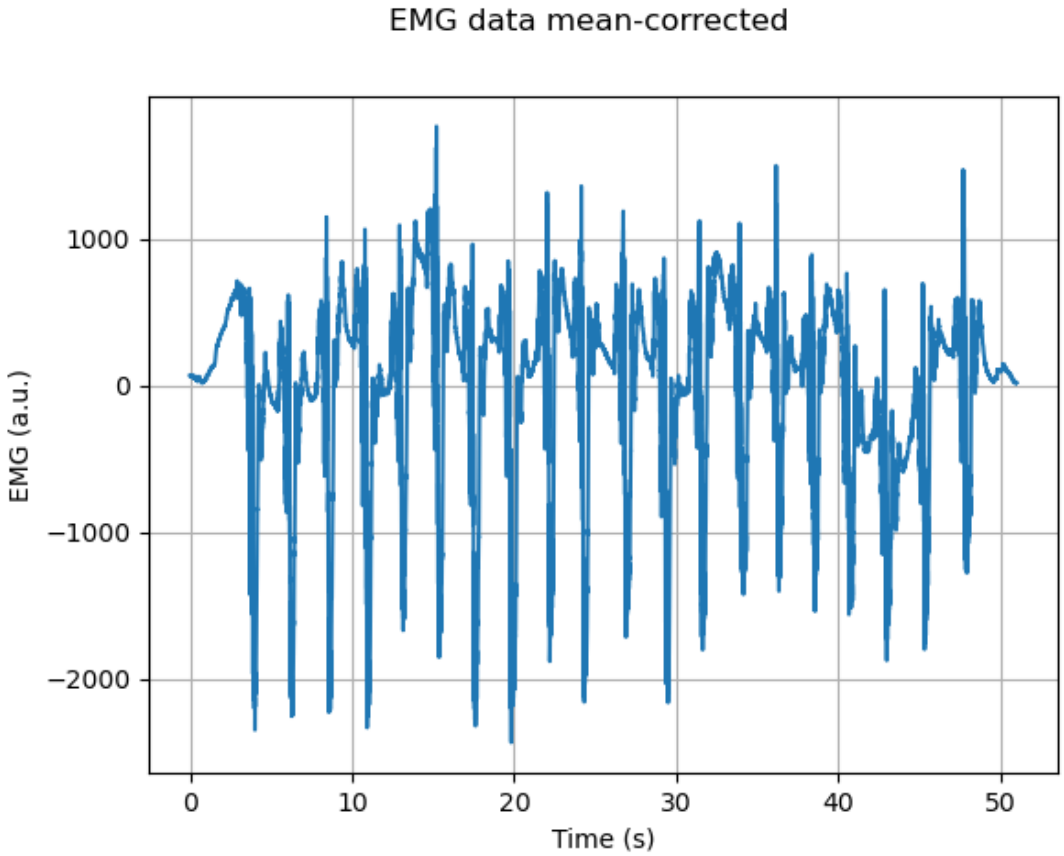

Figure 11. EMG data mean corrected.

Next step is to perform full wave rectification of the signal to have only positive values, see Figure 12.

Python code:

```
# rectify the signal by using np.abs to remove negative values
  emg_rectified=np.abs(emg_filtered)
```

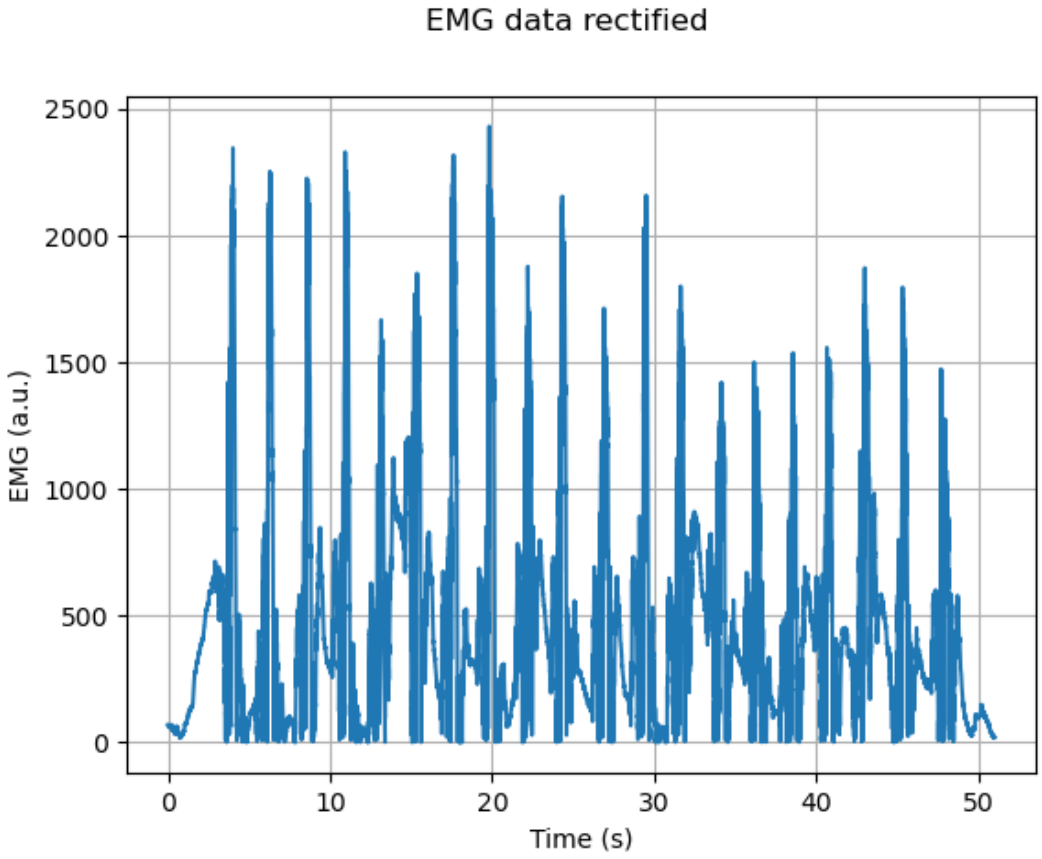

Figure 12. Rectified EMG data.

For feature extraction in the frequency domain and to smoothen the signal, it needs to be filtered. To decide which type of filtering is needed based on the amplitude, a further visual analysis using Fast Fourier transformation is needed.

Python code:

```
def fft_sample(emg_np, dt):
  #runs Fast Fourier Transformation to signal data and plots the amplitudes along with signal
  #function inputs: emg_np is the source signal, dt is the timestep, 0.002 is the value for 500Hz

 from scipy import fftpack
  f = emg_np - np.mean(emg_np) #mean corrected signal
  t = np.array([i/500 for i in range(0, len(emg_np),1)]) # signal samples to time
  n = len(t)
  emg_fft = fftpack.fft(f, n) #signal Fast Fourier Transformation
  psd=emg_fft * np.conj(emg_fft) / n #Powers Spectral Density
  freq = (1/(dt*n)) * np.arange(n) #Frequencies array
  L = np.arange(1,np.floor(n/2), dtype='int')

  #plot the signal and the fft frequency amplitudes
  fig,axs = plt.subplots(2,1)
  plt.sca(axs[0])
```

```
plt.plot(t,f,color='c',LineWidth=1.5, label='Signal')
plt.ylabel('EMG (a.u.)')
plt.xlabel('time / s')
plt.sca(axs[1])
plt.plot(freq[L],psd[L],color='c',LineWidth=2,label='Signal')
plt.xlabel('Frequency Hz')
plt.ylabel('Amplitude')
```

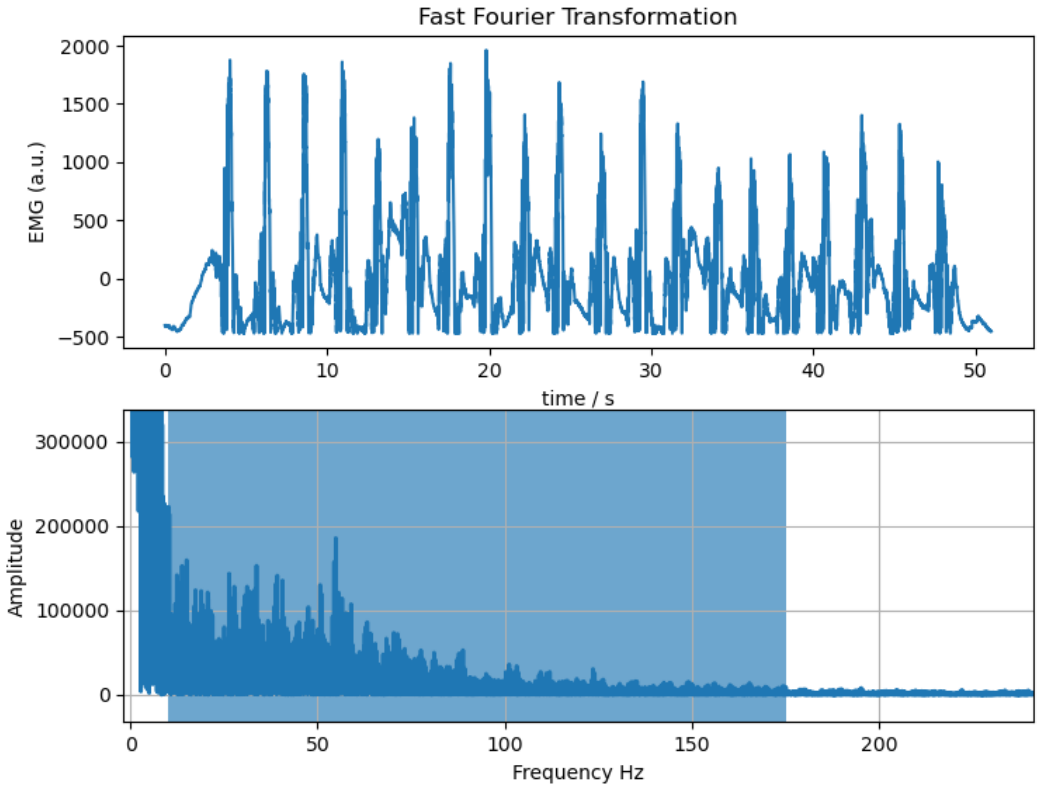

Figure 13. Fast Fourier Transformation of the full wave rectified EMG data with highlighted frequency bandwidth.

In Figure 13 the highest amplitudes are in the low frequency area and higher frequency range is mostly noise. For processing a bandpass filtering will be applied with cut-off values of 10 Hz for high and 175 Hz for low. This frequency range is then applied with an additional 10 Hz low pass filter to smoothen the signal and create the EMG envelope.

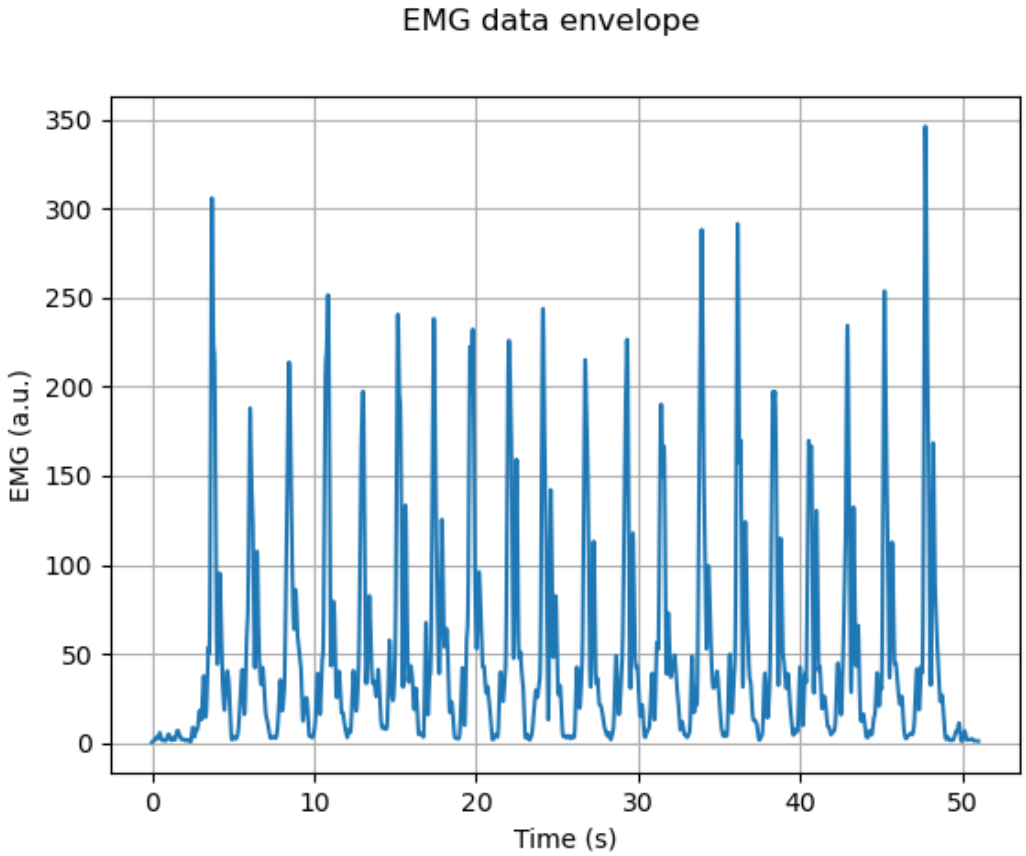

Figure 14. EMG data envelope.

In Figure 14 the EMG signal has been smoothed using bandpass filter between 10 – 175 Hz and a low pass filter of 10Hz which has removed most of the noise from the signal creating a linear envelope.

Python code:

```
# low and highpass bandpass Butterworth filter using mean corrected signal
  high = 10/(1000/2)
  low= 175/(1000/2)
  b, a=sp.signal.butter(4, [high, low], btype='bandpass'
  emg_filtered=sp.signal.filtfilt(b, a, emg_correctmean)

# create lowpass filter and apply to rectified signal to get EMG envelope
low_pass = 10/(1000/2)
b2, a2 = sp.signal.butter(4, low_pass, btype='lowpass')
emg_envelope = sp.signal.filtfilt(b2, a2, emg_filtered)
```

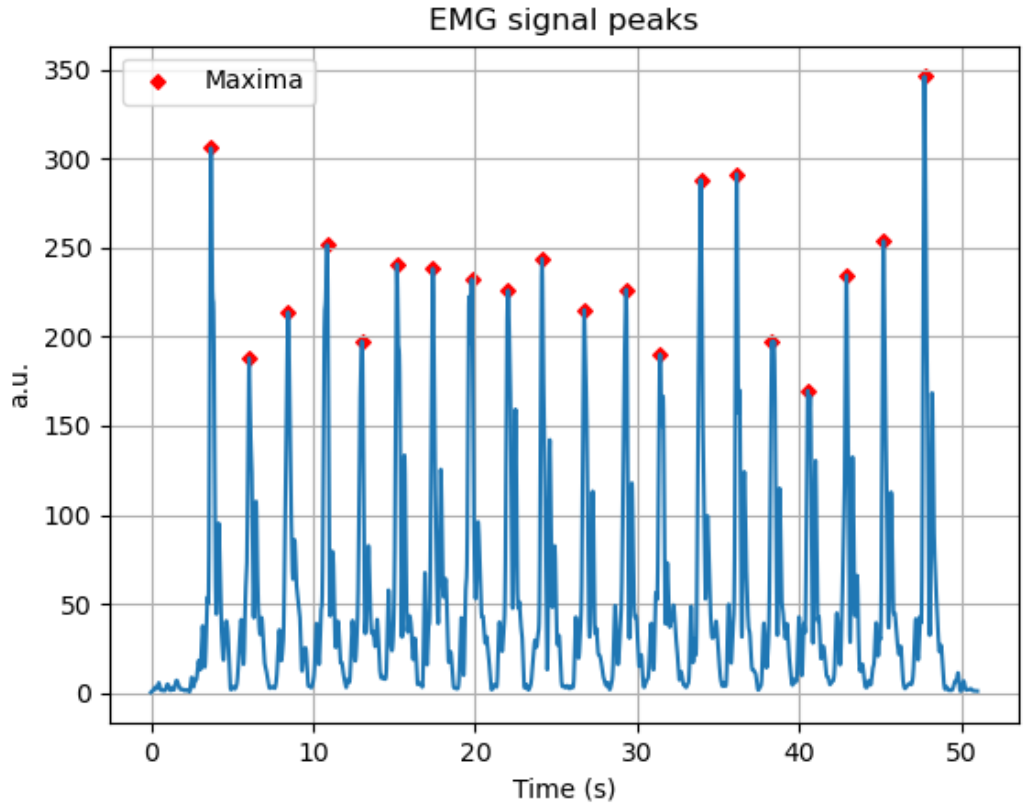

Figure 15. EMG signal peak detection.

Once the signal data has been cleaned, it is ready for peak analysis, see Figure 15.

Before moving to self-written code, the peak detection was done manually using a standalone reader software for EDF files meant for EEG and ECG use. Through iterative development, the peak detection function was successfully implemented in Python (Figure 15). Reliable peak detection still requires manually finetuning the parameters of the function depending on the type of data. When recorded EMG signal is a repetition of a single movement the timeframe before and after the peak value can differ considerably. Also, for certain movements the peak finding method is not the best option for feature detection.

In the function the peak analysis is done against the mean average of the signal and can require adjusting of the different parameters depending on the characteristics of the signal data which is why visual verification of the plot with suggested peaks was usually required.

Python code:

```
def find_peaks(data,time, *args):
    #find peaks from preferably preprocessed data, plots the peaks and returns the peak
location indexes of input data
    from scipy.signal import find_peaks
    emgdata=data
    filename=args[0]
    mean=emgdata.mean()
    peaks = find_peaks(emgdata, height =mean, threshold =0.0001, distance =500,
prominence = 1.2*mean, width = 30, wlen=500, rel_height=1)
    #slight change to function arguments if result is 0
    if len(peaks[0])==0:
        peaks = find_peaks(emgdata, height =mean, threshold =0.0001, distance =500,
prominence = 1.2*mean, width = 30)

    height = peaks[1]['peak_heights']
    peak_pos = time[peaks[0]]

    fig = plt.figure(filename)
    ax = fig.subplots()
    ax.plot(time, emgdata)
    ax.scatter(peak_pos, height, color = 'r', s = 15, marker = 'D', label = 'Maxima')
    ax.legend()
    ax.grid()
    plt.show()
    return peaks[0]
```

The peak location data is used to extract the individual moves from the set and to create the data set used for machine learning model.

Here a single movement is considered starting 0.2 second before the registered peak and continue another 0.5 s after the peak. If the movement was slower or entirely different, the parameters need to be adjusted.

Python code:

```
def collect_peak_samples(time_emg, emg_proc, time_acc, acc_proc, locs):
  emg_samplelist=list(range(len(locs)))
  acc_samplelist=list(range(len(locs)))
  time_loc=list()

  #setting sample frequency, for ECG its 500 and for ACC it's 24
  sfreq=500


  #getting time signatures from location data
  if len(time_emg) == len(time_acc):
      for i, loc in enumerate(locs):
          time_loc.append(time_emg[loc])
          #defining start and end time of sample: -0.2s and 0.5s from peak
```

```
        start = loc - (sfreq*0.2)
        end = loc + (sfreq*0.5)
        emg_samplelist[i]=emg_proc[int(start):int(end)]
        acc_samplelist[i]=acc_proc[int(start):int(end)]

    else:
        for i, loc in enumerate(locs):
            time_loc.append(time_emg[loc])
            #defining start and end time of sample: -0.2s and 0.5s from peak
            start = loc - (sfreq*0.2)
            end = loc + (sfreq*0.5)
            emg_samplelist[i]=emg_proc[int(start):int(end)]
            #getting time signatures from location data
            sfreq=24
            timelocs_array=np.array(time_loc)

            for i, loc in enumerate(timelocs_array):
                loc_in_acc=np.where(time_acc==round(loc))

                #defining start and end time of sample: -0.2s and 0.5s from peak

                start = loc_in_acc[0] - (sfreq*0.2)
                end = loc_in_acc[0] + (sfreq*0.5)
                acc_samplelist[i]=acc_proc[int(start):int(end)]

    return emg_samplelist, acc_samplelist
```

5.2 Datasets and SVM

Once the peak values are discovered from the data, the wave signal is then separated into 0.7s snippets and a dataset is created. The dataset is normalized using the methods provided by Scikit-learn python library. The data set is then labelled based on the recorded gesture and randomized to be split in training and testing data.

The Support Vector Machine model is trained with the training dataset and then fitted with testing data.

For sensor fusion machine learning, both ACC and EMG corresponding data are concatenated before creating the datasets. Due to different resolution of both data types, the lower resolution (ACC) data will be interpolated (Figure 16) according to the higher resolution data (EMG), this way the time sync should be more accurate.

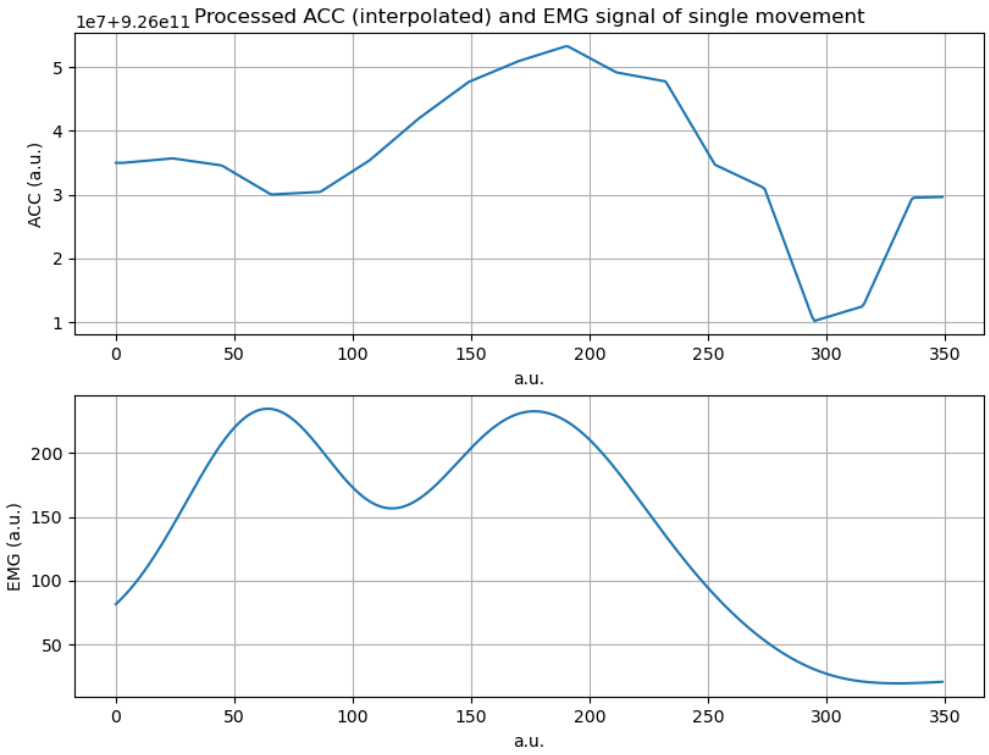

Figure 16. Pre-processed ACC (interpolated) and EMG data of a kick.

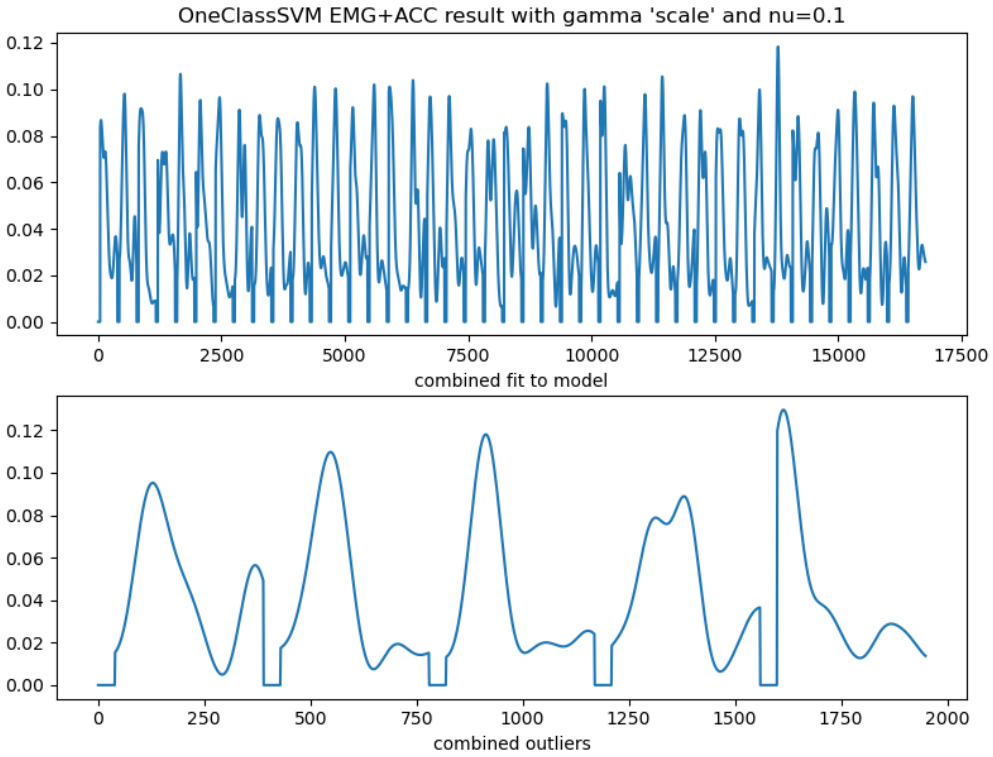

Figure 17. Visualized OCSVM results of fitting and outliers of combined EMG and ACC movement data.

First machine learning algorithm was tested as single class support machine vector model. The single class SVM is primarily meant for anomaly detection and here (Figure 17) the logic was to test and see if the outliers in the testing dataset would be identified as anomalies. The training set consisted of 191 samples and the testing set was 48 samples (20% of the total data). The size of the datasets is not sufficient for proper testing but enough to validate the proof of concept. A single class SVM was not optimal for this type of activity as the algorithm outcome is modified by adjusting the OneClassSVM classifier nu value which by default gives about 50% result and here 89,5%.

Python code:

```
from sklearn.svm import OneClassSVM
from sklearn import preprocessing


combined=[]
# normalizing the sample data and then combining lists for dataset
for i, item in enumerate(list_acc):
  normalized_acc=preprocessing.normalize(item)
  normalized_emg=preprocessing.normalize(list_emg[i])
```

```
    for j in range(0,len(normalized_acc)):
        combined.append(np.concatenate((normalized_emg[j], normalized_acc[j])))

    #randomize data set and split to training and data sets
    np.random.shuffle(combined)
    X_train=combined[:int(len(combined)*0.8)]
    X_test=combined[int(len(combined)*0.8):]

    clf = OneClassSVM(gamma='scale', nu=0.1).fit(X_train)
    fitted=clf.fit_predict(X_test)
```

A multiclass SVM was the main goal and at first the datasets were built with only EMG data to establish a baseline. Using multiclass also meant that additional movements were needed as source data for test and comparison.

Python code:

```
    from sklearn import svm
        emg_list, acc_list, loc_list=populate_data(emg_src_files, acc_src_files, loc_src_files,
folder)
    list_emg, list_acc=process_data(emg_list, acc_list, loc_list)
    combined=[]
    # combining lists for 2nd dataset
    for i, item in enumerate(list_acc):
      normalized_acc=preprocessing.normalize(item)
      normalized_emg=preprocessing.normalize(list_emg[i])
      for j in range(0,len(normalized_acc)):
          combined.append(np.concatenate((normalized_emg[j], normalized_acc[j])))

    np.random.shuffle(combined)
    X2_train=combined[:int(len(combined)*0.8)]
    X2_test=combined[int(len(combined)*0.8):]
    X = np.concatenate((X_train, X2_train))
    y = np.arange(len(X))
    y[:len(X_train)] = 0
    y[len(X_train):] = 1
    clf = svm.SVC()
    clf.fit(X, y)
    clf.predict(X2_test)
    fitted_a = clf.predict(X_test)
    fitted_b = clf.predict(X2_test)
    print(f'SVM multiclass model training size is: {len(X)}, and testing size for class 0:
{len(X_test)} and the % fitting the model from test set is:
{((fitted_a==0).sum()/len(X_test))*100}\n'
          f'testing size for class 1: {len(X2_test)} and the % fitting the model from test set is:
{((fitted_b==1).sum()/len(X2_test))*100}'))
```

Based on the movements focused on this study in Table 1, the corresponding accelerometer data to the EMG data doesn't considerably improve the recognition percentage. This is likely due to lower sample resolution 24Hz of the accelerometer sensor vs 500Hz of the ECG, meaning that there are less samples for accelerometer data than in EMG data in the same time frame of the movement and the missing data was interpolated resulting in synchronization issues between peak values of the two data types.

Table 1. SVM fitting results when run 10 times against the sample set

Movement	**Sensor**	**Average %**
Kick	EMG	98,75
	EMG + ACC	97,06
Knee to the side	EMG	93,00
	EMG + ACC	91,67

Sample set	**Training**	**Testing**	**Total**
Kick	191	48	239
Knee to the side	92	24	116

5.3 Outcomes

Here I will review how the research questions were answered based on the implementation outcomes during the project.

RQ1 How well is the SVM model suited for leg movement classification?

The SVM SVC model is a good fit for this type of classification, provided that data is valid and well processed.

RQ2 What are the requirements for suitable data for training machine learning model on leg movement recognition?

The data used for the training needs to be consistent and each type of move clearly instructed. Attention needs to be paid on the placement of the sensors, recording environment and the repeatability of each movement. The recorded data should be validated to remove erroneous recordings before proceeding.

RQ3 What type of pre-processing of data is required for SVM?

The signal data needs to be:

- rectified
- filtered with bandpass adjusted for muscle movement frequency like 10–150Hz
- EMG enveloped with for example low pass filter
- after pre-processing the data should be normalized and here it was done based on peak values in the data
- when using additional sensors or type of sensor data, time synchronization is essential

RQ4 How could wellness-monitor type of sensors be repurposed to provide sufficiently detailed data?

The suitability of the device sensors depends on the device features such as sensor sensitivity and resolution and whether the recorded data can be processed with open-source tools or if the data is encrypted and can be used only on vendor proprietary platforms. The Beat2Phone device used by this project offered good data resolution and with developer tools the collected raw data could be further processed even when it was not originally meant for this purpose.

6 Conclusions

The quality of data is paramount. There are many variables that affect the quality of data. The sensor electrode might not be properly in place, or the sensors are not picking the signal properly. The amount of body hair or moisture between skin and the sensor will affect the signal data quality. Overly noisy data must be discarded, and analysis can only be done to a properly pre-processed sample data.

After pre-processing, the detection of an individual movement from the data requires different parameters depending on the movement duration and type, the form of a kick movement is different from a move from sitting to standing.

When recording a reference training set for a movement it's important to also consider the muscle fatigue how it will affect the results as events recorded beginning of the set are likely to be different when comparing to the latter ones in the set.

The achieved SVM model results with combined ACC and EMG samples did not outperform the model with plain EMG data. If the synchronization issue is solved with improved programming, only then it could be verified that vectorized accelerometer data would improve precision on sensor data fusion recognition. It is also possible that there could be better methods on combining the EMG and ACC data than concatenating the samples.

The Support Vector Machine SVC algorithm used works well for training the machine learning model with this type of data. The success rate of test results for selected material with a multiclass model was consistently above and at higher end of 90%. However, it should be noted that the sample datasets used here are very small and the danger of overfitting the model must be taken into account. The SVC reportedly scales to samples up to tens of thousands after which it becomes impractical (Scikit 2022b).

The devices used for data recording proved to be sufficiently accurate for required level of data clarity. Proper implementation would require additional sensors to be incorporated in the build to get data also from other related muscles of the limb. Additional sensor data needs to be synced and additional proprietary interface programming is required. The developer app version used during this project had stability issues and often the data recording session had to be restarted or afterwards it was discovered that some of the data had been lost or missing. Also, while there was a limited amount of movement types, the data processing methods applied here still required considerable amount of manual work and adjustments.

For further research additional machine learning methods could be applied to improve classification results like Bayesian classifiers and random forest algorithms. Neural network solutions would likely be efficient with large datasets but in addition to increased model training requirements also require computational power. The freely available large HuGaDB (Chereshnev 2017) gait database can likely be very useful for additional classification research. In terms of a proof-of-concept type of physical therapeutic support device with sensors and computation functionality there would be a lot more to consider in addition to gathering enough valid movement data and to be able to process it in real-time.

References

Ambily O.A & Judeth Malliga, T. 2011. Agile Software Development an Approach to Light Weight from Heavy Weight. Available at: https://www.researchgate.net/publication/50392056_AGILE_SOFTWARE_DEVELOPMENTAN_APPROACH_TO_LIGHT_WEIGHT_FROM_HEAVY_WEIGHT.

Anaconda Inc. 2022. Anaconda Distribution. Available at: https://www.anaconda.com/products/distribution (Accessed 30 October 2022).

Badawi A.A., Al-Kabbany A & Shaban H.A., 2020. "Sensor Type, Axis, and Position-Based Fusion and Feature Selection for Multimodal Human Daily Activity Recognition in Wearable Body Sensor Networks", Journal of Healthcare Engineering, vol. 2020, Article ID 7914649, 14 pages. Available at: https://doi.org/10.1155/2020/7914649.

Burns E. 2021. Machine Learning. Available at: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML (Accessed 28 October 2022)

Chereshnev R. & Kert'esz-Farkas, A., 2017. Hugadb, "Human gait database for activity recognition from wearable inertial sensor networks," in Proceedings of the International Conference on Analysis of Images, Social Networks and Texts, pp. 131–141, Moscow, Russia, July 2017.

Coherent Market Insights 2022. Total Knee Arthroplasty Market Analysis. https://www.coherentmarketinsights.com/market-insight/total-knee-arthroplasty-market-4635 (Accessed: 18 September 2022).

Eurostat 2022a. Mortality and life expectancy statistics. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Mortality_and_life_expectancy_statistics (Accessed: 14 September 2022).

Eurostat 2022b. Population structure and Aging. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Population_structure_and_ageing#The_share_of_elderly_people_continues_to_increase (Accessed: 14 September 2022).

Gramfort A., Luessi M., Larson E., Engemann DA., Strohmeier D., Brodbeck C., Goj R., Jas M., Brooks T., Parkkonen L. & Hämäläinen M. 2013. MEG and EEG data analysis with MNE-Python. Front. Neurosci. 7:267. doi: 10.3389/fnins.2013.00267.

Halaki M., & Ginn K. 2012. “Normalization of EMG Signals: To Normalize or Not to Normalize and What to Normalize to?”. Computational Intelligence in Electromyography Analysis – A Perspective on Current Applications and Future Challenges. *IntechOpen.* Available at: http://dx.doi.org/10.5772/49957.

Kobsar, D., & Ferber, R. 2018. Wearable Sensor Data to Track Subject-Specific Movement Patterns Related to Clinical Outcomes Using a Machine Learning Approach. *Sensors*, 18(9), 2828. https://doi.org/10.3390/s18092828.

OECD (2019), “Hip and knee surgery”, in Health at a Glance 2019: OECD Indicators, OECD Publishing, Paris. https://doi.org/10.1787/e7af69ca-en.

Pabinger, C., Lothaller, H., & Geissler, A. 2015. Utilization rates of knee-arthroplasty in OECD countries. *Osteoarthritis and cartilage*, *23*(10), 1664–1673. https://doi.org/10.1016/j.joca.2015.05.008.

Pabinger, C., Geissler, A. 2014. Utilization rates of hip arthroplasty in OECD countries. *Osteoarthritis and Cartilage*,22(6), 734-741. https://doi.org/10.1016/j.joca.2014.04.009.

Pedregosa, F., Varoquaux, G., Gramfort, A., Vincent, M. & Thirion, B.2011. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011. Available at: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

Popuri, S. K., Gobbert M. K. 2017. A Comparative Evaluation of Matlab, Octave, R, and Julia on Maya. Available at: https://mdsoar.org/handle/11603/11302.

The Python Software Foundation, 2022. General Python FAQ. Available at: https://docs.python.org/3/faq/general.html (Accessed: 14 September 2022).

Scikit-learn 2022a. Support Vector Machines. Available at: https://scikit-learn.org/stable/modules/svm.html (Accessed: 1 September 2022).

Scikit-Learn 2022b. C-Support Vector Classification. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html (Accessed: 1 September 2022).

Spiewak C., Islam R., Zaman A-U., Habibur Rahman M. 2018. A Comprehensive Study on EMG Feature Extraction and Classifiers. *Op Acc J Bio Eng & Bio Sci* 1(1)- 2018. Available at: http://dx.doi.org/10.32474/OAJBEB.2018.01.000104.

VitalSignum 2019. Beat2Phone ECG instruction manual in Finnish. Available at: https://www.vitalsignum.com/wp-content/uploads/2021/05/VS-2019-008-rev2.4-Beat2Phone-ECG-Instruction-Manual_FI.pdf.

VitalSignum 2022. Beat2Phone ECG Sensor, Mobile Application and Cloud Service. Available at: https://www.vitalsignum.com/en/beat2phone/ (Accessed: 30 October 2022).

Winter, D. 2005. Biomechanics and motor control of human movement. John Wiley & Sons Inc.