

# Clinical Movement Analysis Lab Assignment 2022



Vincent Belpaire  
Supervisor: Prof. Malcolm Forward

University of Ghent  
Faculty of Engineering and Architecture  
Bachelor of Science: Biomedical Engineering

## 1 Introduction

This small report consists out of three parts - 3D marker trajectory, balance assessment and EMG - which are not related to each other in the sense that one part does not depend on the result from another, i.e. each part is a stand alone.

In each section data is used obtained from the *Gait & Movement Lab* at UZ Ghent. This data is stored in an excel file and is manipulated with python to create the necessary results. The used python code is based on the Jupyter Notebook file created by *Juul Van Derbeken* - for this a special thanks to him - and is further edited to match the style of this report.

```
1 import numpy as np # data manipulation
2 import openpyxl # importing data from excel
3 import matplotlib.pyplot as plt # data plotting
4 import scipy.signal as sg # signal analysis
5
```

Code block 1: imported python modules

## 2 3D marker trajectory

Three walk trials were recorded in the Lab.

Using the *openpyxl* module data can be easily imported in python for further manipulation. Remark that in code block 2 on the end of line 5 the first range starts from 14 instead of 5 this due to NoneType elements contained in the range 5:14. Because it is just a small range compared to the full dataset - consisting from  $\pm 500$  rows - it is assumed that this will not heavily impact the results.

```
1 input_file = './Group 3B.xlsx'
2 workbook = openpyxl.load_workbook(input_file)
3 Sheets = workbook.sheetnames # Excel sheet Names
4
5 T1 = np.array([[el.value for el in rij] for rij in workbook[Sheets[0]].rows])[14:,2:].T #
    First Trial, None values for rows < 14
6 T2 = np.array([[el.value for el in rij] for rij in workbook[Sheets[1]].rows])[5:,2:].T #
    Second Trial
7 T3 = np.array([[el.value for el in rij] for rij in workbook[Sheets[2]].rows])[5:,2:].T #
    Third Trial
8 T = [T1, T2, T3]
9
```

Code block 2: importing 3D marker trajectory data

### 2.1 Walking speed

For each trial the forward propagation direction is plotted in figure 1. Note that on each plot multiple trajectory lines occur. Each line corresponds to a marker placed on the subject who executed these trials. The global

(average) walking speed is then calculated as the mean of the different marker speeds. The results are placed in table 1.

```

1 fig, ax = plt.subplots(1,3, figsize = (18,6),sharey=True)
2 ax[0].set_ylabel('distance in mm', fontsize=15)
3
4 for i, t in enumerate(T):
5     AS = [] # average speeds
6     for y in t[1::3]:
7         v = np.diff(y) # instantanious speeds
8         ASL = np.mean(v) # averaged speed
9         AS.append(ASL)
10        ax[i].plot(y), ax[i].set_title(f'trial {i+1}', fontsize=15)
11
12    AS = np.array(AS)
13    speed = np.mean(AS) # Averaged speed for all markers
14    std = np.std(AS)
15    print(f'Average speed for Trial {i} = ',speed, ' and std = ', std)
16 plt.show()
17

```

Code block 3: walking speed calculation and trajectory plotting

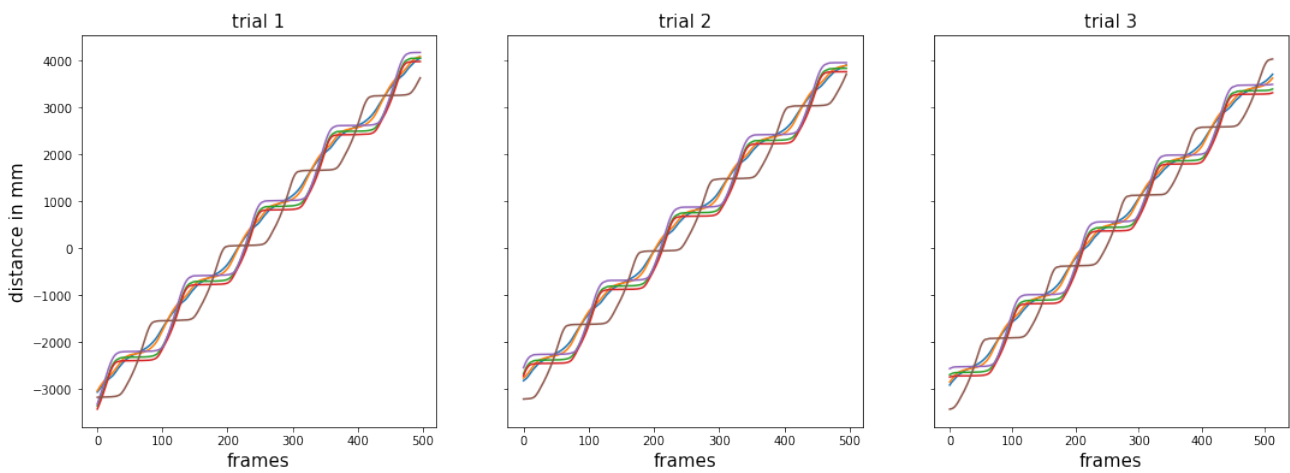


Figure 1: forward propagation direction

trial	walking speed [mm/frame]	std [mm/frame]
1	14.58	0.48
2	13.42	0.32
3	12.61	0.97

Table 1: Walking speed for different trials

## 2.2 Stride length

The stride length is calculated as the difference in foward propagation direction values corresponding to the local minima of the ankle marker height. To approximate these local minima a treshhold is used as displayed in figure 2. The numerical results are listed in table 2.

```

1 fig, ax = plt.subplots(1,3, figsize = (18,6),sharey=True)
2 ax[0].set_ylabel('Height in mm', fontsize=15)
3 AHS = [] # average stride length for each trial.
4 for i, t in enumerate(T):
5     TAZ = t[8] # Trial Ankle Z-coordinate
6     TAY = t[7] # Trial Ankle Y-coordinate (forward movement direction)
7     Tp = np.percentile(TAZ, 30) # Treshold at 30%

```

```

8 TP = np.ones(np.shape(TAZ))*Tp # treshhold for graph
9 P = np.where(TAZ < Tp, 1, 0) # ones where Trial is under treshhold value
10 HS = np.where(np.diff(P) == 1) # locations where values dip under treshhold values.
11 Y = TAY[HS]
12
13 ax[i].plot(TAZ), ax[i].plot(TP)
14 ax[i].legend(['Data', 'Treshhold'], loc="upper right", fontsize=15)
15 ax[i].set_title(f"Trial {i+1}: Ankle", fontsize=15)
16 print(np.diff(Y))
17 print(f'Trial {i+1}: Average stride length is: ', np.mean(np.diff(Y)), " and std = ", np
18 .std(np.diff(Y)))
19 plt.show()

```

Code block 4: plotting marker height with treshhold and calculating stride length

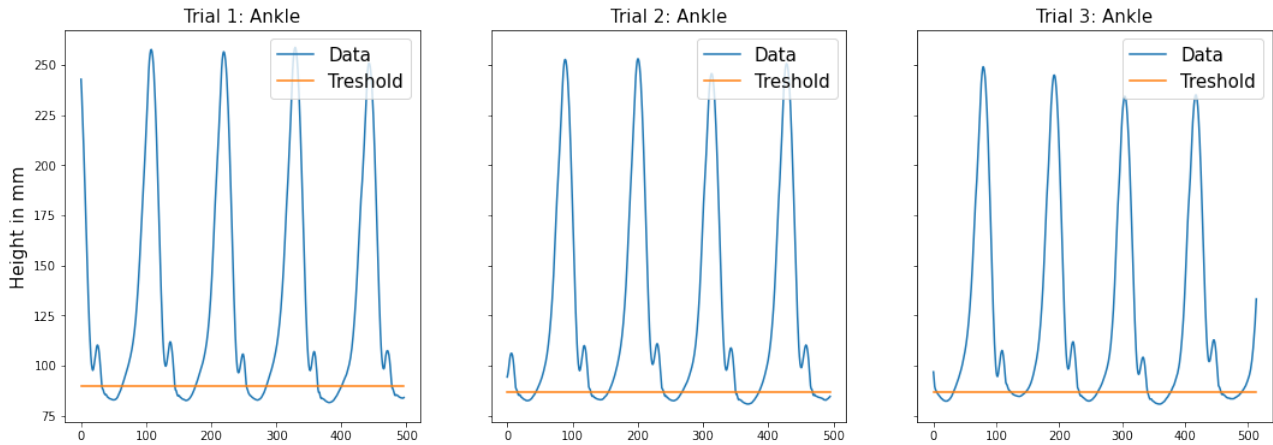


Figure 2: Ankle marker height

trial	walking speed [mm]	std [mm]
1	1597.86	20.73
2	1554.80	19.87
3	1503.08	55.59

Table 2: Stride length

The stride length is in the first place related to the person's height, since taller humans can take larger steps.

### 3 Balance assessment

```

1 input_file = '.\Grp 3B Wk 2.xlsx'
2 workbook = openpyxl.load_workbook(input_file)
3 sheets = G2A_2.sheetnames #Sheet Names
4
5 EO = np.array([[el.value for el in rij] for rij in workbook[sheets[1]].rows])[6:15006,2:].T
6   #First Trial #15006
7
8 EC = np.array([[el.value for el in rij] for rij in workbook[sheets[2]].rows])[6:12976,2:].T
9   #Second Trial #12976
10
11 EOFX, EOFY, EOFZ = EO[0], EO[1], EO[2] # Eyes Open Force Vectors of x,y,z coordinates
12 ECFX, ECFY, ECFZ = EC[0], EC[1], EC[2] # Eyes Closed Force Vectors of x,y,z coordinates

```

Code block 5: importing balance assessment data

First it is recommended to plot the overall xy-plane with the center of pressures (CoP) on it. This is displayed in figure 3. This gives a first impression between the two cases.

```

1 fig, (ax0, ax1) = plt.subplots(1,2, figsize = (18,6), sharey=True)
2 ax0.scatter(EOFX, EOFY, s=1), ax0.set_title('eyes open (EO)', fontsize=15), ax0.set_xlabel('
    x-coordinate in mm', fontsize=15), ax0.set_ylabel('y-coordinate in mm', fontsize=15)
3 ax1.scatter(ECFX, ECFY, s=1), ax1.set_title('eyes closed (EC)', fontsize=15), ax1.set_xlabel
    ('x-coordinate in mm', fontsize=15)
4 plt.show()
5

```

Code block 6: plotting CoP

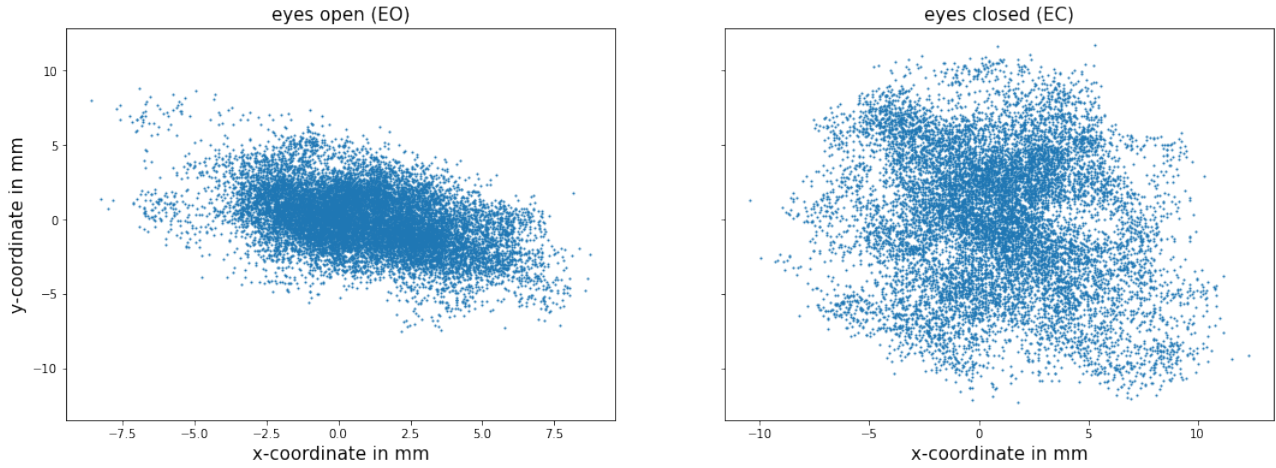


Figure 3: Center of pressure

Different parameters such as path length, swept area and mean radius suffice to make a distinguishment between EO and EC. Before one of these parameters can be calculated one must calculate the Arithmetic Mean Point (AMP) and shift the axis to this point in space. The results of the three mentioned parameters are listed in table 3.

trial	path length [mm]	swept area [mm <sup>2</sup> ]	mean radius [mm]
EO	15508.36	13936.26	2.879
EC	13462.69	22494.91	5.248

Table 3: Calculated parameters

```

1 def AMP(x): # calculating AMP
2     return np.mean(x)
3
4 EOFXs, EOFYs = EOFX - AMP(EOFX), EOFY - AMP(EOFY) # shifting to AMP
5 ECFXs, ECFYs = ECFX - AMP(ECFX), ECFY - AMP(ECFY) # shifting to AMP
6
7 def Mean_radius(x, y):
8     r = (x**2 + y**2)**(1/2) # radius
9     return np.mean(r), np.std(r)
10
11 def Path_length(x, y):
12     L = (np.diff(x)**2+np.diff(y)**2)**(1/2) # length
13     return np.sum(L)
14
15 def Swept_area(x, y):
16     r = (x**2 + y**2)**(1/2) # radius
17     L = (np.diff(x)**2+np.diff(y)**2)**(1/2) # length
18     a, b, c = L, r[:-1], r[1:]
19     S = 1/2*(a+b+c)
20     A = (S*(S-a)*(S-b)*(S-c))**(1/2)
21     return np.sum(A)
22

```

Code block 7: calculating the parameters

In table 3 one might expect every parameter to be larger in the EC scenario but instead the path length reaches the highest value. Probably this is due to the difference in data points - more data points imply a longer path that can be passed. Looking to code block 5 one can observe on line 5 and 6 a great difference in data points between EO and EC ( $15006 > 12976$ ).

Some practical notes have to be taken into account for these measurements. The force plate exists out of a collection of sensors. There can be small errors on the linearity of these sensors, i.e. the amount of displacement measured is not fully proportional with the measured voltage, which leads to small errors in the data. It is crucial to minimize these errors for accurate measurements. Moreover do the number of sensors also influence the accuracy by which there can be measured.

Besides these practical limitations, there is also a choice to be made by the executor of the measurement, namely which sampling rate to use. There are two factors to take into account: what is the maximal sampling rate the force plate can physically handle and by which frequencies does the subject on the force plate move. Since the subject on the force plate is human it wise to start by looking at the human movement frequency and to aim the sampling rate somewhere in between this human movement frequency and the maximal sampling rate. A higher sampling rate would cause more noise in the data, hence aiming to lower sampling rates that still would capture all the human movement is the goal.

## 4 EMG

```
1 G2A_3_input = '.\G3B_EMG Lab3.xlsx'
2 G2A_3 = openpyxl.load_workbook(G2A_3_input)
3 SN3 = G2A_3.sheetnames #Sheet Names
4
5 EN = np.array([[el.value for el in rij] for rij in G2A_3[SN3[0]].rows])[11:10951,2:].T #EMG
    Normal walking #10951
6 ET = np.array([[el.value for el in rij] for rij in G2A_3[SN3[1]].rows])[11:11891,2:].T #EMG
    Toe walking #11891
7
8
```

Code block 8: input EMG data

```
1 ENG = EN[0] - EN[1] #EMG Normal-walking Gastrocnemius
2 ENT = EN[2] - EN[3] #EMG Normal-walking Tibialis-anterior
3 ETG = ET[0] - ET[1] #EMG Toe-walking Gastrocnemius
4 ETT = ET[2] - ET[3] #EMG Toe-walking Tibialis-anterior
5
6 fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(2,2, figsize = (14,8), sharey=True)
7 ax0.plot(ENG), ax0.set_title('Gastrocnemius Normal', fontsize=15), ax0.set_xlim(0,10000),
    ax0.set_ylabel('Voltage [V]', fontsize=15)
8 ax1.plot(ETG), ax1.set_title('Gastrocnemius Toe', fontsize=15), ax1.set_xlim(0,10000)
9 ax2.plot(ENT), ax2.set_title('Tibialis Ant Normal', fontsize=15), ax2.set_xlim(0,10000), ax2
    .set_ylabel('Voltage [V]', fontsize=15)
10 ax3.plot(ETT), ax3.set_title('Tibialis Ant Toe', fontsize=15), ax3.set_xlim(0,10000)
11
```

Code block 9: raw EMG plotting

Looking at figure 4, capturing the raw EMG data, the following observations can be made: the separation between voltage bursts is shorter and the amplitudes reach higher values in the toe walking case in comparison with the normal walking case.

```
1 ENG = ENG - np.mean(ENG)
2 ENT = ENT - np.mean(ENT)
3 ETG = ETG - np.mean(ETG)
4 ETT = ETT - np.mean(ETT)
5 ENG, ENT, ETG, ETT = abs(ENG), abs(ENT), abs(ETG), abs(ETT)
6
7 fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(2,2, figsize = (14,8), sharey=True)
8 ax0.plot(ENG), ax0.set_title('Gastrocnemius Normal', fontsize=15), ax0.set_xlim(0,10000),
    ax0.set_ylabel('Voltage [V]', fontsize=15)
9 ax1.plot(ETG), ax1.set_title('Gastrocnemius Toe', fontsize=15), ax1.set_xlim(0,10000)
```

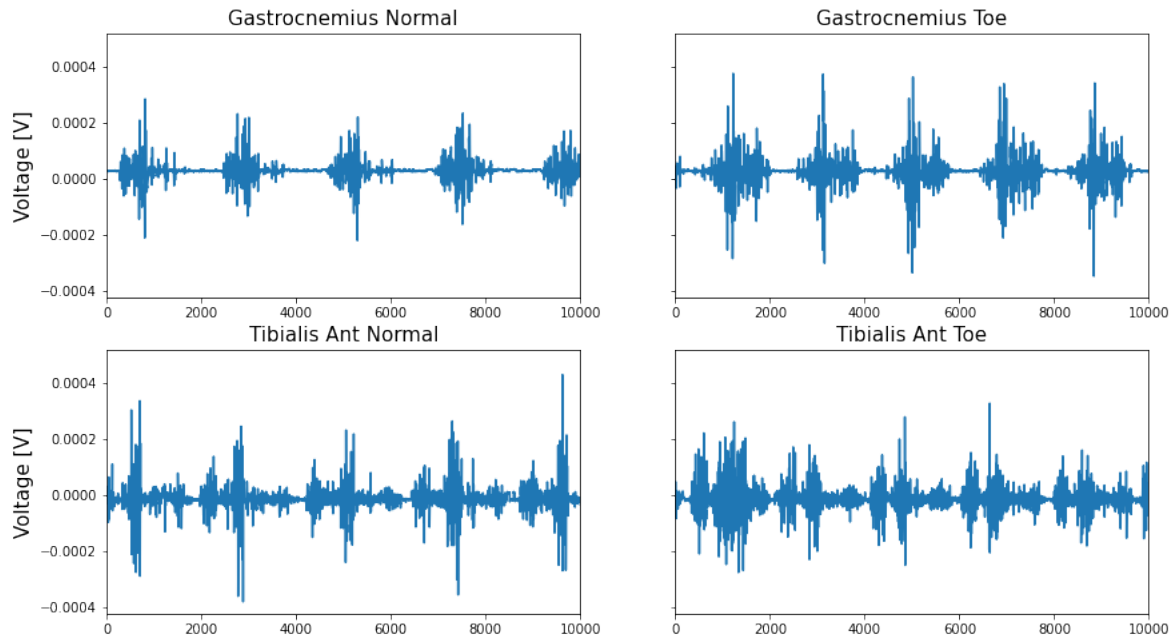


Figure 4: raw EMG signals

```

10 ax2.plot(ENT), ax2.set_title('Tibialis Ant Normal', fontsize=15), ax2.set_xlim(0,10000), ax2
    .set_ylabel('Voltage [V]', fontsize=15)
11 ax3.plot(ETT), ax3.set_title('Tibialis Ant Toe', fontsize=15), ax3.set_xlim(0,10000)
12 plt.show()
13

```

Code block 10: rectified EMG calculation and plotting

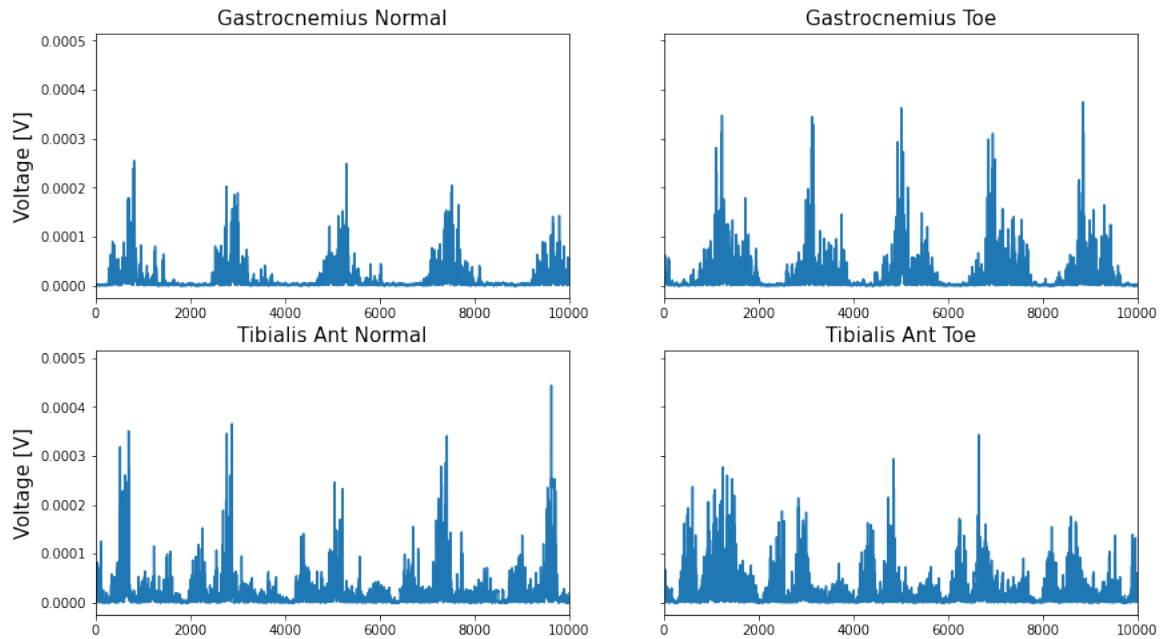


Figure 5: rectified EMG signals

Starting from the rectified EMG signals, i.e. the absolute value taken from the original signals, one could produce smooth rectified EMG by filtering out the high frequencies in the signal. This can be realized with a low-band pass filter.

At last if one would want to deduce the resting EMG level, i.e. the measured voltage without the muscle actually being contracted, one should establish a threshold of EMG on/off levels. This can be derived from the smoothed out rectified EMG signal. The voltages below this threshold can be considered as the resting EMG level.