Deal with tracker data

#%% deal with tracker data

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.signal import butter, filtfilt

from scipy.interpolate import interp1d

# create function of relative angle

def calculate\_theta(data1, data2):

m = np.shape(data1)[0]

theta = np.zeros(m)

for i in range(m):

A = data1[i, :]

B = data2[i, :]

nor\_A = A / np.linalg.norm(A)

nor\_B = B / np.linalg.norm(B)

R = np.array([[nor\_A[0], nor\_A[1]], [-nor\_A[1], nor\_A[0]]])

c = np.dot(R, nor\_B)

theta[i] = np.arctan2(c[1], c[0])

return theta

# create function for gimbal lock problem

def unwrap\_deg(data):

data = data.copy()

dp = np.diff(data)

dps = np.mod(dp + np.pi, 2 \* np.pi) - np.pi

dps[np.logical\_and(dps == -np.pi, dp > 0)] = np.pi

dp\_corr = dps - dp

dp\_corr[np.abs(dp) < np.pi] = 0

data[1:] += np.cumsum(dp\_corr)

return data

# if you need to use -> change the input file name

# input file

file\_path = '/Users/kairenzheng/Library/CloudStorage/OneDrive-AuburnUniversity/KINE7670\_homeworks/opencap\_study/data\_opencap\_squat\_trc/squat3＿tracker.csv'

#open data

df = pd.read\_csv(file\_path, header=None)

# get sampling interval

sampling\_interval = float(df.iloc[3, 0])

# get frequency = 1 / sampling\_interval

fs = 1 / sampling\_interval

# pick up positions

Lhip\_xy\_position = df.iloc[2:, 1:3].astype(float).to\_numpy()

LKnee\_xy\_position = df.iloc[2:, 4:6].astype(float).to\_numpy()

Lankle\_xy\_position = df.iloc[2:, 7:9].astype(float).to\_numpy()

# create vectors

Lankle\_knee\_line = Lankle\_xy\_position - LKnee\_xy\_position

Lhip\_knee\_line = Lhip\_xy\_position - LKnee\_xy\_position

# get relative angles

Lknee\_sagital\_angle = calculate\_theta(Lhip\_knee\_line, Lankle\_knee\_line)

# check if it has gimbal lock and change unit to degree

Lknee\_sagital\_angle = (unwrap\_deg(Lknee\_sagital\_angle)) \* (180 / np.pi)

# set up the original angle is the first frame

Lknee\_sagital\_angle = Lknee\_sagital\_angle - Lknee\_sagital\_angle[0]

# elements for events

mid\_hip\_y\_pos = Lhip\_xy\_position[:, 1]

threshold = np.mean(mid\_hip\_y\_pos[:5]) - 0.05

event1 = np.where(mid\_hip\_y\_pos < threshold)[0][0] if np.any(mid\_hip\_y\_pos < threshold) else None

event2 = np.where(mid\_hip\_y\_pos[event1:] >= threshold)[0][0] + event1 if event1 is not None and np.any(mid\_hip\_y\_pos[event1:] >= threshold) else None

# cut data -> data during from event1 to event2

Lknee\_sagital\_angle = Lknee\_sagital\_angle[event1: event2]

# find the knee flexion angle

Lknee\_sagital\_angle = 180 - Lknee\_sagital\_angle

# find the minimum knee flexion angle

max\_flexion\_knee\_angle = np.min(Lknee\_sagital\_angle)

# export data

export = {

"Rknee\_sagital\_angle": Lknee\_sagital\_angle,

"Max\_knee\_flexion\_angle": ([max\_flexion\_knee\_angle] \* len(Lknee\_sagital\_angle))

}

output\_df = pd.DataFrame(export)

# output to csv file

# if you need to use -> change the output file name

output\_df.to\_csv('/Users/kairenzheng/Desktop/squat3\_tracker.csv', index=False)