Deal with opencap data

# input tool box

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 the process function

def process(file\_path, output\_path):

#open data

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

# pick up frequency

fs = int(float(df.iloc[2, 1]))

# time interrval = 1 / frequenccy

sampling\_interval = 1 / fs

# pick up points

segment\_names = ['mid\_hip', 'Rhip', 'RKnee', 'RAnkle', 'Lhip', 'LKnee', 'LAnkle']

segment\_columns = [(23, 26), (26, 29), (29, 32), (32, 35), (35, 38), (38, 41), (41, 44)]

# change variable types

segments = {name: df.iloc[6:, start:end].astype(float).to\_numpy() for name, (start, end) in zip(segment\_names, segment\_columns)}

# create butterworth\_filter function

def butterworth\_filter(data, cutoff, fs, order=4, filter\_type='low'):

nyquist = 0.5 \* fs

return filtfilt(\*butter(order, cutoff / nyquist, btype=filter\_type), data, axis=0)

# set up elements for functions

cutoff = 6

# run butterworth\_filter function for each points

filtered\_segments = {name: butterworth\_filter(seg, cutoff, fs) for name, seg in segments.items()}

# set up events -> standard it 5 cm downward

mid\_hip\_y\_pos = filtered\_segments["mid\_hip"][:, 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

# set up elements to get relative angles

Lhip\_xy\_position = filtered\_segments['Lhip'][:, 0:2]

LKnee\_xy\_position = filtered\_segments['LKnee'][:, 0:2]

LAnkle\_xy\_position = filtered\_segments['LAnkle'][:, 0:2]

# create two vectors

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

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

# create function of relative angles

# data1 and data2 are the two vectors that we created

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)

theta[i] = np.arctan2(np.cross(nor\_A, nor\_B), np.dot(nor\_A, nor\_B))

return theta

# run function of getting relative angles

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

# create function for gimbal lock problem

def unwrap\_deg(data):

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

# 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]

# get the knee angle

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

# pick up the minimum knee angle

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

# export data

export = {

"Rknee\_sagital\_angle": Lknee\_sagital\_angle[event1:event2],

"Max\_knee\_flexion\_angle": ([max\_flexion\_knee\_angle] \* (event2 - event1))

}

output\_df = pd.DataFrame(export)

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

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

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

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

output\_path = '/Users/kairenzheng/Desktop'

process(file\_path, output\_path)