Compare data

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import interp1d

# open data

df = pd.read\_csv('/Users/kairenzheng/Library/CloudStorage/OneDrive-AuburnUniversity/KINE7670\_homeworks/opencap\_study/data\_opencap\_squat\_trc/results.csv')

# pick knee flexion data and max knee flexion angle

def extract\_data(start\_col):

knee\_flexion = df.iloc[:, start\_col].dropna().astype(float).to\_numpy()

max\_flexion = df.iloc[0, start\_col + 1]

return knee\_flexion, max\_flexion

# pick up datas

opencap\_cols = [1, 4, 7]

tracker\_cols = [10, 13, 16]

opencap\_data = [extract\_data(col) for col in opencap\_cols]

tracker\_data = [extract\_data(col) for col in tracker\_cols]

# calculate rms

rms = np.sqrt(np.mean([(o[1] - t[1]) \*\* 2 for o, t in zip(opencap\_data, tracker\_data)]))

print("rms:", rms)

# create function for time normalization -> to 100 frames

def time\_normalize(data, length=100):

return interp1d(np.linspace(0, 1, len(data)), data, axis=0, kind='linear')(np.linspace(0, 1, length))

# run time normalization

data\_100 = [np.vstack((time\_normalize(o[0]), time\_normalize(t[0]))).T for o, t in zip(opencap\_data, tracker\_data)]

# create ICC function

def icc\_calculate(Y, icc\_type):

n, k = Y.shape

mean\_Y, SST = np.mean(Y), ((Y - np.mean(Y)) \*\* 2).sum()

x, x0 = np.kron(np.eye(k), np.ones((n, 1))), np.tile(np.eye(n), (k, 1))

X = np.hstack([x, x0])

predicted\_Y = X @ np.linalg.pinv(X.T @ X) @ X.T @ Y.flatten("F")

SSE, MSE = ((Y.flatten("F") - predicted\_Y) \*\* 2).sum(), ((Y.flatten("F") - predicted\_Y) \*\* 2).sum() / ((n - 1) \* (k - 1))

MSC, MSR = (((np.mean(Y, 0) - mean\_Y) \*\* 2).sum() \* n) / (k - 1), (((np.mean(Y, 1) - mean\_Y) \*\* 2).sum() \* k) / (n - 1)

if icc\_type == "icc(2)":

ICC1 = (MSR - MSE) / (MSR + (k - 1) \* MSE + k \* (MSC - MSE) / n)

return ICC1

# run ICC

for i, data in enumerate(data\_100, 1):

print(f"ICC(2,1) for squat {i}:", icc\_calculate(data, "icc(2)"))

# make graph

for i, data in enumerate(data\_100, 1):

plt.figure(figsize=(8, 5))

plt.plot(data[:, 0], label='OpenCap Knee Flexion')

plt.plot(data[:, 1], label='Tracker Knee Flexion')

plt.title(f'Squat {i}: OpenCap vs Tracker Knee Flexion')

plt.xlabel('Normalized Frame (%)')

plt.ylabel('Knee Flexion Angle (degrees)')

plt.legend()

plt.grid(True)

plt.show()

Deal with RMS for opencap and trakcer data (example for opencap)

import numpy as np

# pick up data

# knee flexion angle of opencap and tracker

squat1\_open = data\_100[0][:, 0] #final number 0 = opencap data; 1 = tracker data

squat2\_open = data\_100[1][:, 0] #final number 0 = opencap data; 1 = tracker data

squat3\_open = data\_100[2][:, 0] #final number 0 = opencap data; 1 = tracker data

# Compute mean per frame

mean\_per\_frame = np.mean([squat1\_open, squat2\_open, squat3\_open], axis=0)

# Compute RMS for each frame

rms\_per\_frame1 = np.sqrt((squat1\_open - mean\_per\_frame) \*\* 2)

# Compute the average RMS difference across all frames

avg\_rms\_diff1 = np.mean(rms\_per\_frame1)

print("RMS per frame:", rms\_per\_frame1)

print("Average RMS difference:", avg\_rms\_diff1)

# Compute RMS for each frame

rms\_per\_frame2 = np.sqrt((squat2\_open - mean\_per\_frame) \*\* 2)

# Compute the average RMS difference across all frames

avg\_rms\_diff2 = np.mean(rms\_per\_frame2)

# Compute RMS for each frame

rms\_per\_frame2= np.sqrt((squat2\_open - mean\_per\_frame) \*\* 2)

# Compute the average RMS difference across all frames

avg\_rms\_diff2 = np.mean(rms\_per\_frame2)

# Compute RMS for each frame

rms\_per\_frame3= np.sqrt((squat3\_open - mean\_per\_frame) \*\* 2)

# Compute the average RMS difference across all frames

avg\_rms\_diff3 = np.mean(rms\_per\_frame3)

# Compute RMS for each frame

rms\_per\_frame3= np.sqrt((squat3\_open - mean\_per\_frame) \*\* 2)

# Compute the average RMS difference across all frames

avg\_rms\_diff3 = np.mean(rms\_per\_frame3)

# average rms of three trails

average\_rms = (avg\_rms\_diff1 + avg\_rms\_diff2 + avg\_rms\_diff3) / 3