e_rr

March 14, 2023

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import time
import sys
import os
from scipy.signal import savgol_filter
from scipy.interpolate import interp1d
from tqdm import tqdm
sys.path.append(os.path.join(os.getcwd(), "..", ".."))
from support.omniwheel_calculation import *
from datetime import datetime
from support.pd_support import *
from support.calculations_support import *
from support.ar_calculations import *
from numba import njit
import polars as pl
from scipy.signal import find_peaks
from scipy.signal import peak_widths
from scipy.signal import peak_prominences
from support.imu_calculations import *
```

[ ]:
```
_parent_folder = "encoder_validation"
_folder_name = "sk28_err_0"
_base_pth = os.path.dirname(os.getcwd())
_base_pth = os.path.join(_base_pth, "..","recording_programs", "test_data",␣
 ↪_parent_folder)
_base_pth
```

[ ]:
```
'c:\\Users\\CMC\\Documents\\openposelibs\\pose\\armbo\\simulation\\..\\recording
_programs\\test_data\\encoder_validation'
```

[ ]:
```
_sk_df = pd.read_csv(os.path.join(_base_pth, _folder_name, "imu01.csv"))
_sk_df["rust_time"] = _sk_df["rust_time"].apply(lambda x: datetime.
 ↪fromtimestamp(x))
# set zero
_sk_df.rename(columns={"rust_time": "time", "e_fr":"e_t"}, inplace=True)
```

```
_sk_df = set_zero(_sk_df, column_name = ["e_t", "e_rr", "e_rl"])
# rename columns
_sk_df["e_t"] =  -_sk_df["e_t"]

_sk_df
```

[ ]:

```
                      sys_time                      time  e_t  e_fl   e_rr  \
0     2023-03-14 11:41:57.514565 2023-03-14 11:41:57.515051    0     0      0
1     2023-03-14 11:41:57.701557 2023-03-14 11:41:57.701884    0     0      0
2     2023-03-14 11:41:57.702064 2023-03-14 11:41:57.702427    0     0      0
3     2023-03-14 11:41:57.702549 2023-03-14 11:41:57.702755    0     0      0
4     2023-03-14 11:41:57.702549 2023-03-14 11:41:57.703026    0     0      0
…                            …                          …  …   …    …
5565  2023-03-14 11:43:07.753226 2023-03-14 11:43:07.753271    0     0 -40973
5566  2023-03-14 11:43:07.765638 2023-03-14 11:43:07.765889    0     0 -40973
5567  2023-03-14 11:43:07.778132 2023-03-14 11:43:07.778432    0     0 -40973
5568  2023-03-14 11:43:07.790914 2023-03-14 11:43:07.791362    0     0 -40973
5569  2023-03-14 11:43:07.803645 2023-03-14 11:43:07.803756    0     0 -40973

      e_rl                          rtc      mils  sync         ax  \
0        0  2019-01-01 08.27.07.000000 AM  10627261     0 -10.140318
1        0  2019-01-01 08.27.07.000000 AM  10627274     0 -10.128355
2        0  2019-01-01 08.27.07.000000 AM  10627286     0 -10.143907
3        0  2019-01-01 08.27.07.000000 AM  10627299     0 -10.142112
4        0  2019-01-01 08.27.07.000000 AM  10627312     0 -10.131944
…    …                              …       …   …         …
5565     0  2019-01-01 08.28.17.000000 AM  10697497     0 -10.140318
5566     0  2019-01-01 08.28.17.000000 AM  10697509     0 -10.146897
5567     0  2019-01-01 08.28.17.000000 AM  10697522     0 -10.159458
5568     0  2019-01-01 08.28.17.000000 AM  10697535     0 -10.152281
5569     0  2019-01-01 08.28.17.000000 AM  10697547     0 -10.140916

            ay        az        gx        gy        gz          mx  \
0     0.132190  0.046655 -0.068598 -0.007622  0.030488  235.500015
1     0.148938  0.008374 -0.060976 -0.007622  0.015244  218.700012
2     0.133386  0.024524 -0.060976 -0.007622  0.022866  225.900009
3     0.143554  0.046655 -0.060976 -0.022866  0.015244  225.900009
4     0.153125  0.032300 -0.068598 -0.022866  0.030488  218.700012
…        …       …        …         …         …          …
5565  0.135778  0.026916 -0.068598 -0.015244  0.022866  206.700012
5566  0.127404  0.034094 -0.068598  0.000000  0.038110  230.700012
5567  0.110656  0.004187 -0.060976 -0.015244  0.022866  201.900009
5568  0.140564  0.015552 -0.083842 -0.007622  0.022866  216.300003
5569  0.133984  0.027515 -0.068598 -0.015244  0.022866  221.100006

              my         mz
```

```
0     -93.300003 -71.700005
1    -102.900002 -77.700005
2     -98.100006 -72.300003
3    -100.500008 -69.900002
4    -105.300003 -69.300003
…           …          …
5565 -114.900002 -69.300003
5566 -112.500008 -74.100006
5567  -95.700005 -73.500000
5568 -105.300003 -78.900002
5569 -105.300003 -68.700005

[5570 rows x 18 columns]
```

[ ]:
```
# type in marker details
_xm = get_marker_name(3)
_zm = get_marker_name(1)
_om = get_marker_name(2)
```

[ ]:
```
_mocap_df, st_time = read_rigid_body_csv(os.path.join(_base_pth, _folder_name.
 ↪split("_")[0] ,_folder_name + ".csv"))
_mocap_df = add_datetime_col(_mocap_df, st_time, "seconds")
```

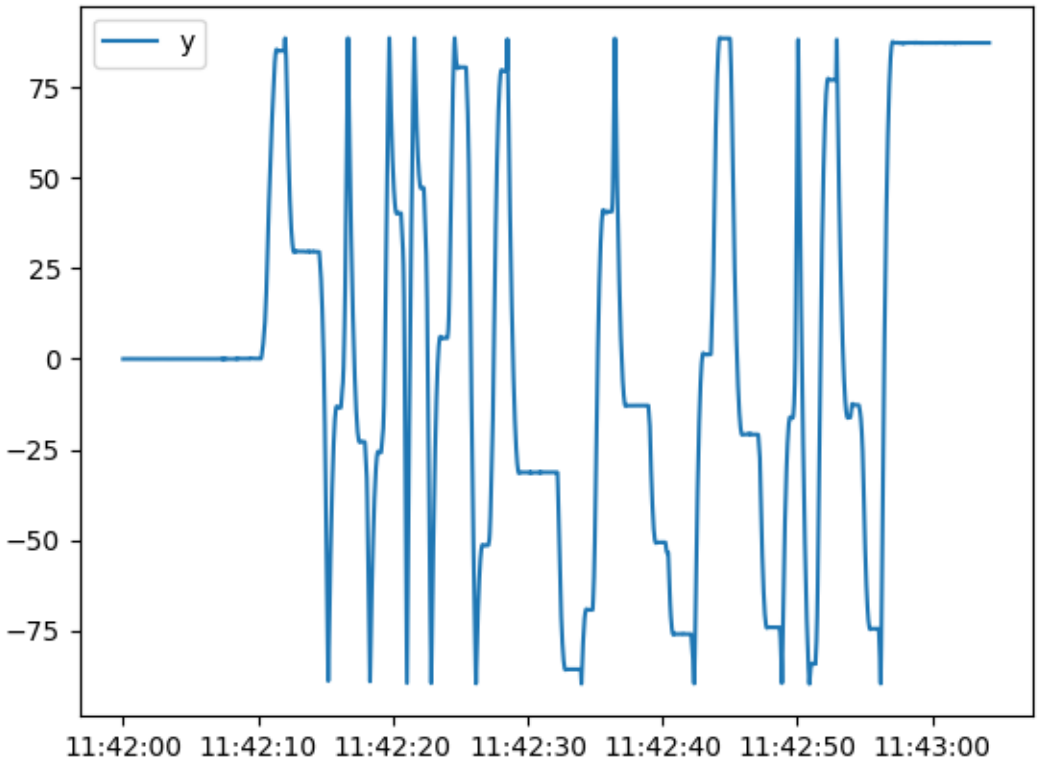
[ ]:
```
# This cell is optimized to run faster using polars

# calculate rotation matrix from xvec, zvec, org
_m_df = _mocap_df.copy()
_m_df = pl.from_pandas(_m_df)

_rotmat_i = []
for i in tqdm(range(len(_m_df))):

    _x_vec = _m_df[[_xm["x"], _xm["y"], _xm["z"]]][i, :].to_numpy().T
    _z_vec = _m_df[[_zm["x"], _zm["y"], _zm["z"]]][i, :].to_numpy().T
    _org = _m_df[[_om["x"], _om["y"], _om["z"]]][i, :].to_numpy().T

    _rotmat_i.append(calculate_rotmat(_x_vec, _z_vec, _org))
# calculating del rotmat for mc
_del_r = []
for i in tqdm(range(len(_rotmat_i))):
    _del_r.append(_rotmat_i[i].T@_rotmat_i[0])

# calculating angle for mc
_theta_x = []
_theta_y = []
_theta_z = []
```

```
for i in tqdm(_del_r):
    _theta_x.append(np.arctan2(i[2,1], i[2,2]))
    _theta_y.append(np.arctan2(-i[2,0], np.sqrt(i[2,1]**2 + i[2,2]**2)))
    _theta_z.append(np.arctan2(i[1,0], i[0,0]))

_theta_x = np.array(_theta_x)
_theta_y = np.array(_theta_y)
_theta_z = np.array(_theta_z)

# converting them to degrees
_theta_x = np.rad2deg(_theta_x)
_theta_y = np.rad2deg(_theta_y)
_theta_z = np.rad2deg(_theta_z)
```

```
100%|      | 6423/6423 [00:03<00:00, 1891.28it/s]
100%|      | 6423/6423 [00:00<00:00, 349988.50it/s]
100%|      | 6423/6423 [00:00<00:00, 107019.91it/s]
```

[ ]:
```
# plt.plot(_m_df["time"][1000:2000], _theta_x[1000:2000], label="x")

plt.plot(_m_df["time"], _theta_y, label="y")
# plt.plot(_m_df["time"][1000:2000], _theta_z[1000:2000], label="z")
# change angle to 0 to 360
plt.legend()
# max(_theta_z)
```

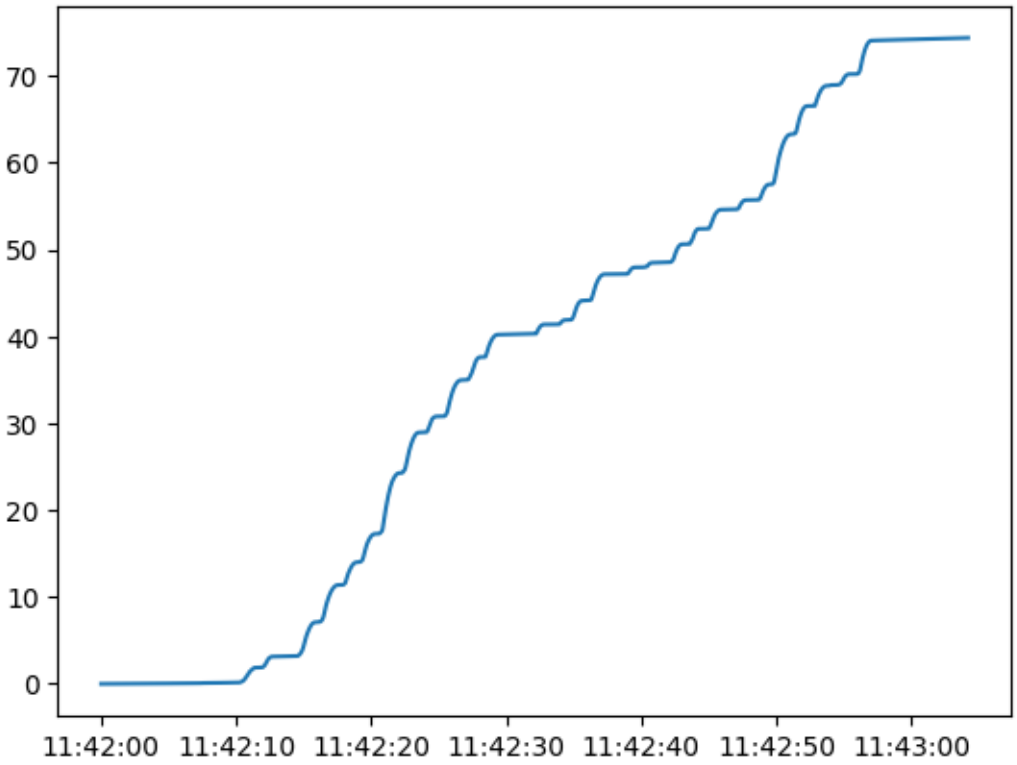
[ ]:
```
<matplotlib.legend.Legend at 0x1fabfe9c7c8>
```



[ ]:
```
theta_df = pd.DataFrame({"time": _m_df["time"], "theta_y": _theta_y})
theta_df["diff"] = abs(theta_df["theta_y"].diff())
# replace nan with 0
theta_df["diff"].fillna(0, inplace=True)
# integrate angle
df, _ = get_orientation(theta_df, "diff")
```

[ ]:
```
plt.plot(df["time"], df["theta"], label="y")
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x1fabf3fe0c8>]
```

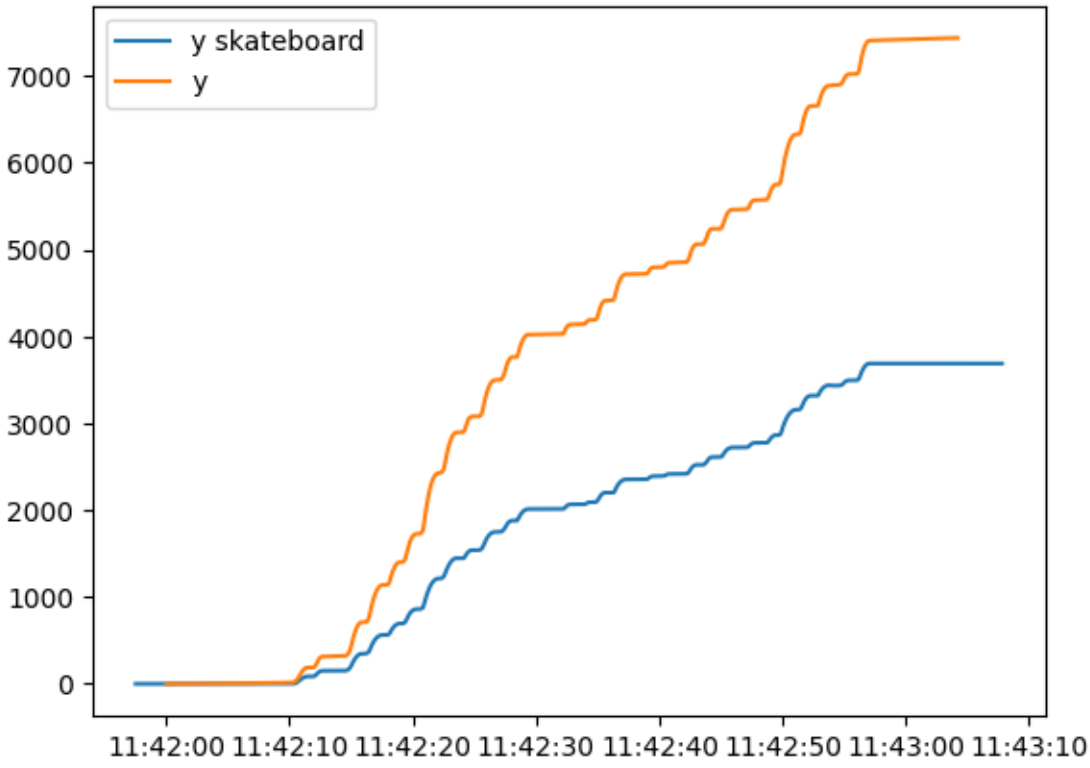


[ ]:
```
_ang_df = _sk_df[['time', 'e_rr']].copy()
```

[ ]:
```
_ang_df["ang_y"] = _ang_df["e_rr"].apply(lambda x: x*0.09)
```

[ ]:
```
plt.plot(_ang_df["time"], -_ang_df["ang_y"], label="y skateboard")
plt.plot(df["time"], df["theta"]*100, label="y")
plt.legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x1faba05d2c8>
```


y skateboard
y
7000
6000
5000
4000
3000
2000
1000
0
11:42:00
11:42:10
11:42:20
11:42:30
11:42:40
11:42:50
11:43:00
11:43:10

[ ]: