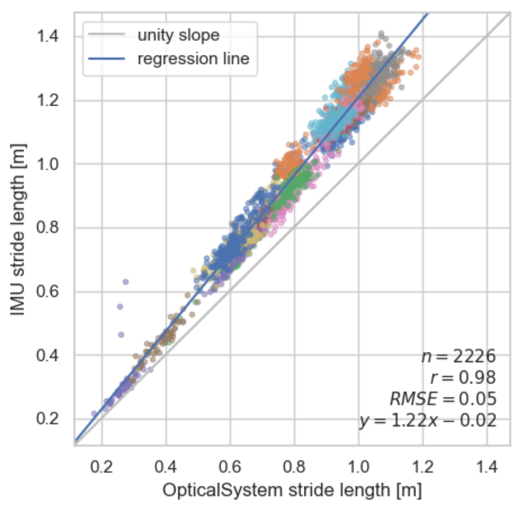
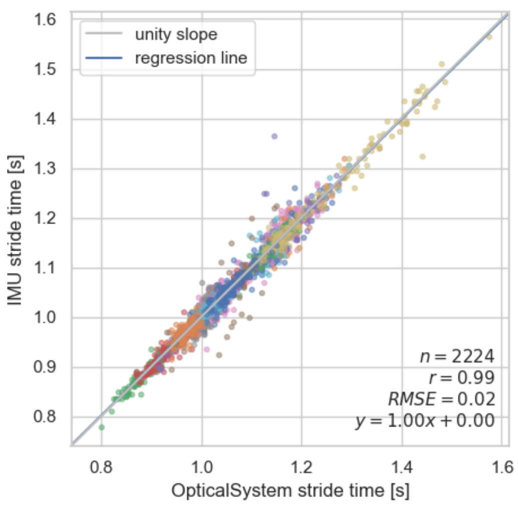
Troubleshooting for the scaling factor in stride length correlation plot.

Eliminating the effects of changing treadmill speed

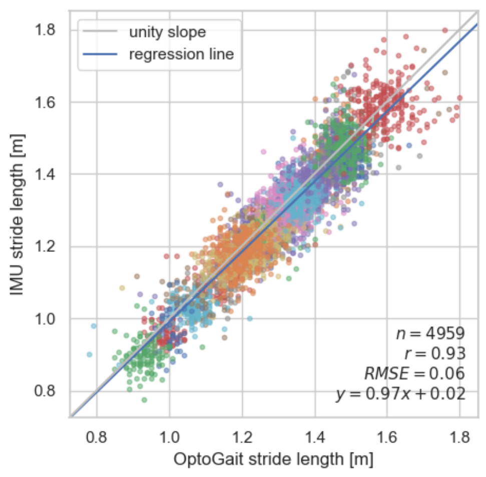
We suspected that the changing treadmill speed during recording might have caused IMU stride length estimation problems. However, the scaling factor of ~1.2 still exists after analyzing only strides from the constant-treadmill-speed segments. The current plots for stride length and stride time indicate that the temporal component was accurately measured, thus excluding the possibility of an inaccurate sampling rate.





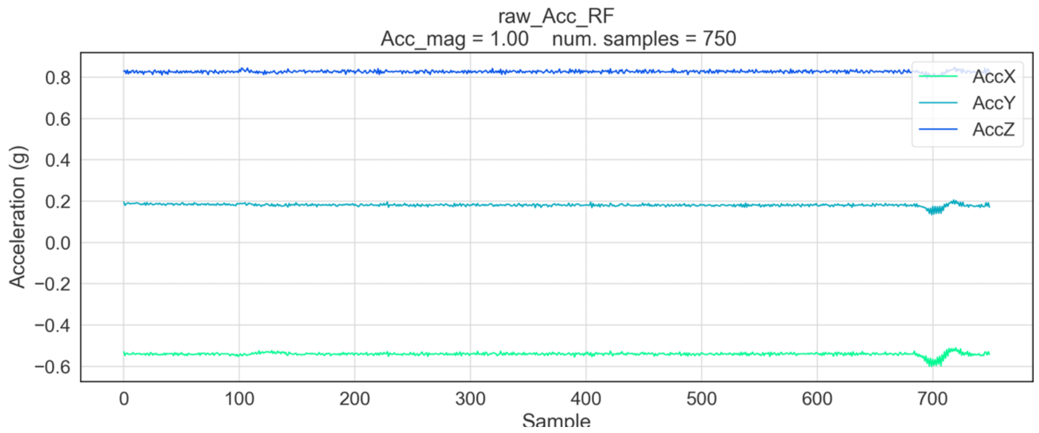
Check the gait analysis pipeline

When using another dataset (the TRIPOD dataset we recorded previously on the treadmill: https://doi.org/10.3390/data6090095), the pipeline produces a normal regression line without overestimating the stride length. This indicates that the data processing pipeline did not cause the scaling issue. (The fit from this plot is not as good as in the original publication because the IMU and the reference system were not synched properly.)



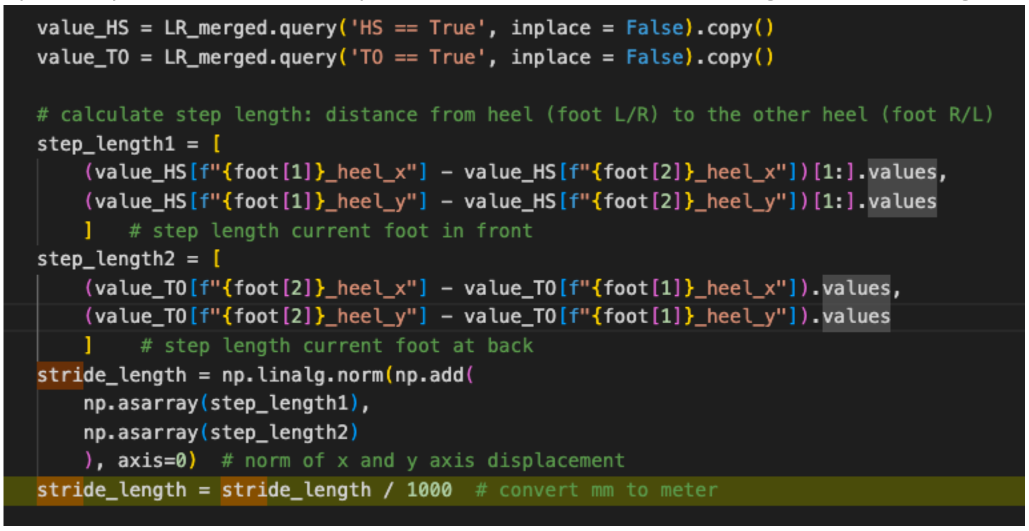
Check if the IMU data is calibrated

Acceleration magnitude for all participants at the beginning of the recording (when they were not moving) is 1 g, i.e., gravity was correctly measured. This indicates that there are no scaling problems with the raw IMU data.



Double-check the stride length calculation from the optical reference system

I did not spot any errors in the stride length calculation. I calculated the stride length from the optical data by adding the lengths of two adjacent steps. This method uses the step lengths relative to the treadmill's surface, instead of the distance traveled by one foot in the optical system coordinate system, to avoid underestimating the stride length.

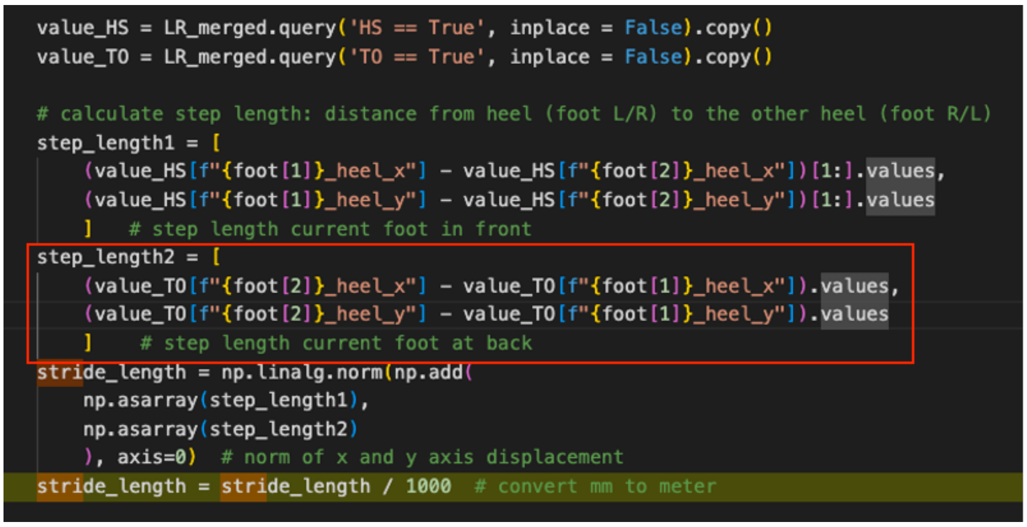


```
value_HS = LR_merged.query('HS == True', inplace = False).copy()
value_TO = LR_merged.query('TO == True', inplace = False).copy()

# calculate step length: distance from heel (foot L/R) to the other heel (foot R/L)
step_length1 = [
    (value_HS[f"{foot[1]}_heel_x"] - value_HS[f"{foot[2]}_heel_x"])[1:].values,
    (value_HS[f"{foot[1]}_heel_y"] - value_HS[f"{foot[2]}_heel_y"])[1:].values
    ]  # step length current foot in front
step_length2 = [
    (value_TO[f"{foot[2]}_heel_x"] - value_TO[f"{foot[1]}_heel_x"]).values,
    (value_TO[f"{foot[2]}_heel_y"] - value_TO[f"{foot[1]}_heel_y"]).values
    ]   # step length current foot at back
stride_length = np.linalg.norm(np.add(
    np.asarray(step_length1),
    np.asarray(step_length2)
    ), axis=0)  # norm of x and y axis displacement
stride_length = stride_length / 1000  # convert mm to meter
```

Update on 2022-10-17

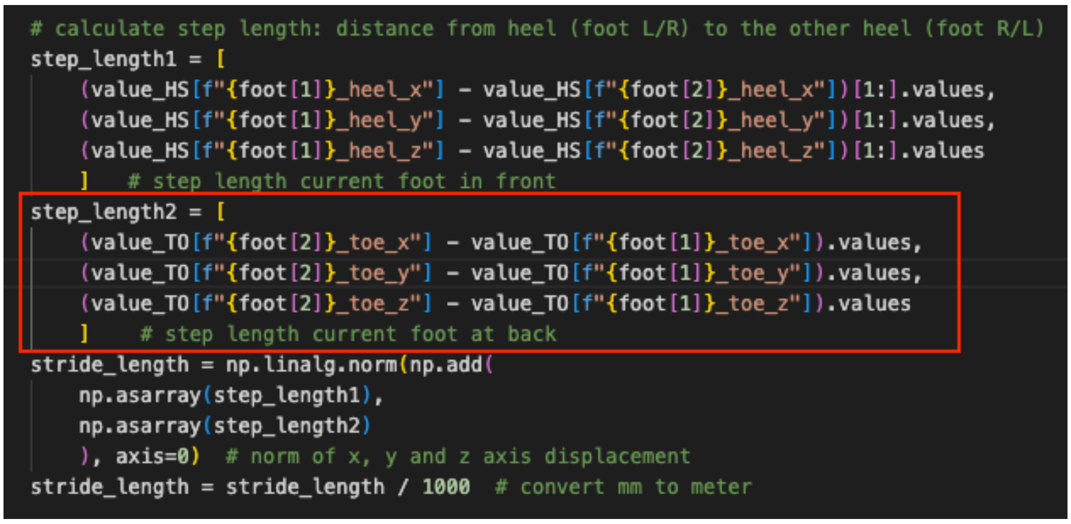
The heel-to-heel measurement of step length is not always correct, because when the current foot is at the back (i.e., at its toe-off event), its heel is already in the air and horizontally moving forward - thus part of the foot length has been subtracted from this step, resulting in a shorter stride length. Instead, for this step, toe-to-toe distance should be used, because toes from both feet are on the treadmill/ground at this point in time.

Old code:



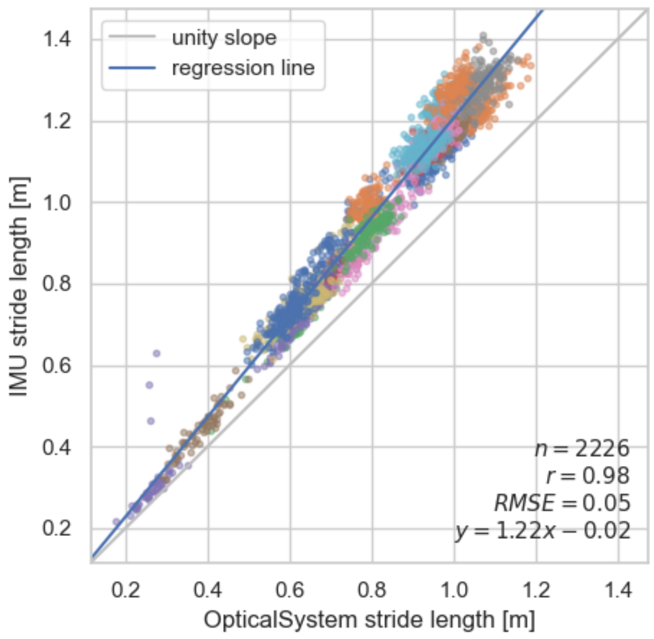
```
value_HS = LR_merged.query('HS == True', inplace = False).copy()
value_TO = LR_merged.query('TO == True', inplace = False).copy()

# calculate step length: distance from heel (foot L/R) to the other heel (foot R/L)
step_length1 = [
    (value_HS[f"{foot[1]}_heel_x"] - value_HS[f"{foot[2]}_heel_x"])[1:].values,
    (value_HS[f"{foot[1]}_heel_y"] - value_HS[f"{foot[2]}_heel_y"])[1:].values
    ]   # step length current foot in front
step_length2 = [
    (value_TO[f"{foot[2]}_heel_x"] - value_TO[f"{foot[1]}_heel_x"]).values,
    (value_TO[f"{foot[2]}_heel_y"] - value_TO[f"{foot[1]}_heel_y"]).values
    ]    # step length current foot at back
stride_length = np.linalg.norm(np.add(
    np.asarray(step_length1),
    np.asarray(step_length2)
    ), axis=0)  # norm of x and y axis displacement
stride_length = stride_length / 1000  # convert mm to meter
```

New code:

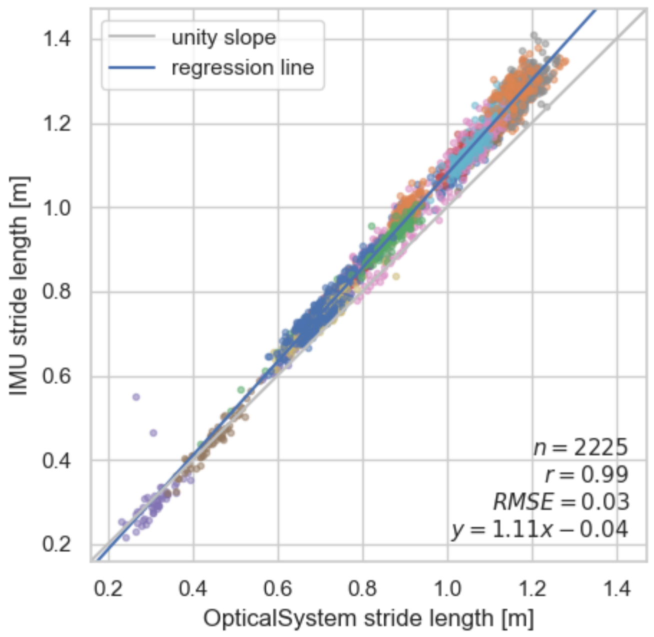


```
# calculate step length: distance from heel (foot L/R) to the other heel (foot R/L)
step_length1 = [
    (value_HS[f"{foot[1]}_heel_x"] - value_HS[f"{foot[2]}_heel_x"])[1:].values,
    (value_HS[f"{foot[1]}_heel_y"] - value_HS[f"{foot[2]}_heel_y"])[1:].values,
    (value_HS[f"{foot[1]}_heel_z"] - value_HS[f"{foot[2]}_heel_z"])[1:].values
    ]   # step length current foot in front
step_length2 = [
    (value_TO[f"{foot[2]}_toe_x"] - value_TO[f"{foot[1]}_toe_x"]).values,
    (value_TO[f"{foot[2]}_toe_y"] - value_TO[f"{foot[1]}_toe_y"]).values,
    (value_TO[f"{foot[2]}_toe_z"] - value_TO[f"{foot[1]}_toe_z"]).values
    ]    # step length current foot at back
stride_length = np.linalg.norm(np.add(
    np.asarray(step_length1),
    np.asarray(step_length2)
    ), axis=0)  # norm of x, y and z axis displacement
stride_length = stride_length / 1000  # convert mm to meter
```

Compare results:
Control treadmill with old code:



Control treadmill with new code:


unity slope
regression line
IMU stride length [m]
OpticalSystem stride length [m]
$n = 2225$
$r = 0.99$
$RMSE = 0.03$
$y = 1.11x - 0.04$