Homework 2 - Advanced Digital Signal Processing

Marco Ceran

A.A. 2020-2021

1 Introduction

The purpose of this homework is to implement Pedestrian Dead Reckoning (PDR) and to get information on the motion of a smartphone held by a walking man from gyroscope and accelerometer data. In particular number of steps, step length, direction and shape of the trajectory is the information that is going to be computed from data. In the second part of the activity a Bayesian tracking algorithm (Kalman filter) will be used to estimate the evolution of the position of the smartphone, along with its speed and the distortion of its trajectory from a true circle.

2 Execution and results

We are given two pairs of three-dimensional arrays, the first one contains the accelerometer while the second one contains the gyroscope data (we have two separate datasets, a and b). The accelerometer data contains information about the instantaneous acceleration of the body in motion sampled with period $T = 10.07\ ms$ for the three axes x, y and z, expressed in m/s^2. The gyroscope data contains instant angular velocity with respect to the three axes x, y and z sampled at the same time instants as the accelerations, the angular velocity is measured in rad/s.

2.1 a.3)

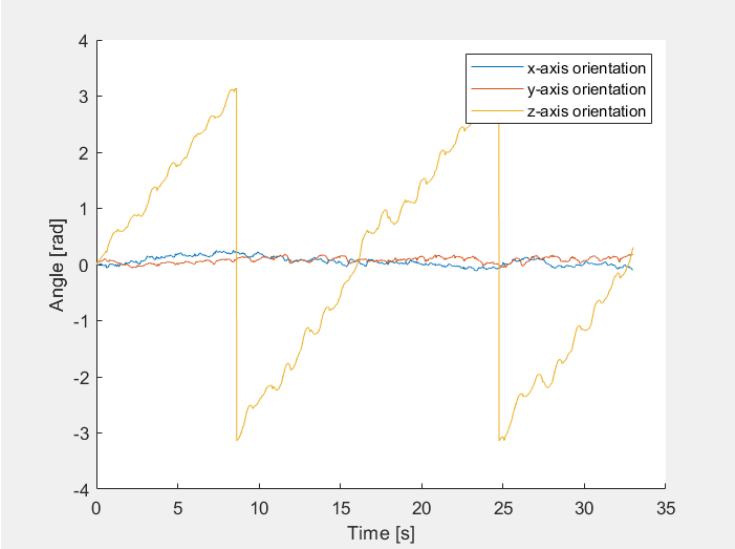
In order to compute the heading direction of the pedestrian in each time instant the angles are obtained by integrating the angular velocities in the G vector:

```
angles = zeros(size(G));
for n = 2:length(angles)
angles(:, n) = angles(:, n-1) + G(:, n)/fs;
end
```

Since the walking pedestrian moves in circular motion holding the phone still, we expect the angles to be (roughly) constant on the x and y axes, while the angles on the z should vary (in particular it should increase linearly) in order to represent the heading direction of the pedestrian. So a rotation matrix is going to be applied to the data in order to have the angles on x and y be zero mean. So this code generates and applies a rotation matrix along the y axis that minimizes the mean absolute value of the angle on the x axis (the mean absolute value of the angle on the y axis is already low enough that a rotation to reduce it has not been deemed necessary):

```
for n = 0:0.001:3.14
    aa = [cos(n) 0 sin(n); 0 1 0 ; -sin(n) 0 cos(n)]*angles;
    avgang(round(1+1000*n)) = mean(abs(aa(1, :)));
end
[~, rotang1] = min(avgang);
rotang1 = rotang1/1000;
rotmat1 = [cos(rotang1) 0 sin(rotang1); 0 1 0 ; -sin(rotang1) 0 cos(rotang1)];
angles = rotmat1*angles;
```

So after applying this rotation the angles that we have with respect to time are:



2.2 a.1)

One of the goal of this activity is to estimate the number of steps taken by the pedestrian: the first thing to do is to apply the rotation matrix to the acceleration values (the same one used for heading direction computation for consistency):

```
Acc = rotmat1*A;
```

After applying the rotation to the acceleration values the mean values of acceleration on the three axes go from *[-1.0695 -0.0252 9.7495]* to *[-0.3360 -0.0252 9.8022]*, this is consistent with the fact that an accelerometer at rest on the surface of the Earth will measure a gravity acceleration of $9.8 m/s^2$.
To estimate the number of steps from the data first the acceleration value on the horizontal *x-y* plane is computed, then the local acceleration variance is computed, in a window of 30 samples, that amount to a fraction of the time taken by the pedestrian to make a step:

```
A_abshorz = sum(Acc(1:2, :).^2).^0.5;
A_locmean = movmean(A_abshorz, [15 15]);
A_locvariance = movmean((A_abshorz - A_locmean).^2, [15 15]);
```

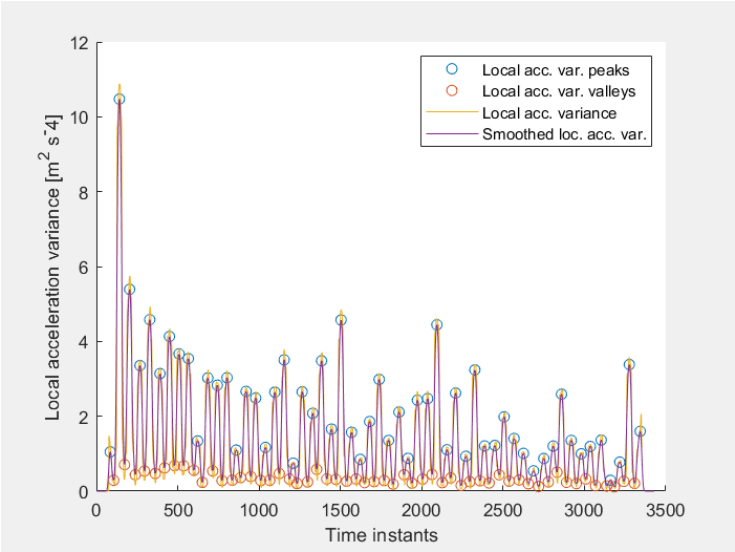
The next step is to estimate the number of steps by the number of peaks and valleys in the local acceleration variance (there is a peak in the instantaneous acceleration during the swing phase of a step, so this fact is used to estimate number of steps). The local acceleration variance previously computed has some oscillations that make the use of a smoothing filter necessary in order to estimate the correct number of peaks and valleys, for this purpose a binomial filter has been chosen (i.e. a discrete-time approximation of a gaussian filter obtained by repeatedly convolving the vector *[0.5 0.5]* with itself):

```
h = [1/2 1/2];
binomialCoeff = conv(h,h);
for n = 1:150
    binomialCoeff = conv(binomialCoeff,h);
end
```

After obtaining this filter the number of peaks and valleys can be estimated by taking the length of the vectors obtained with the MATLAB *findpeaks()* function (the indexes of peaks and valleys have also been computed as they are going to be used in subsequent tasks):

```
[peak, peakind] = findpeaks(conv(A_locvariance, binomialCoeff));
[vall, vallind] = findpeaks(-conv(A_locvariance, binomialCoeff));
vall = -vall;
N_steps = length(vall);     % this is the estimated number of steps
```

With this method the number of steps has been estimated to be 55; the graph of the original and filtered local acceleration variance, along with its peaks and valleys, is here reported:



2.3 a.2)

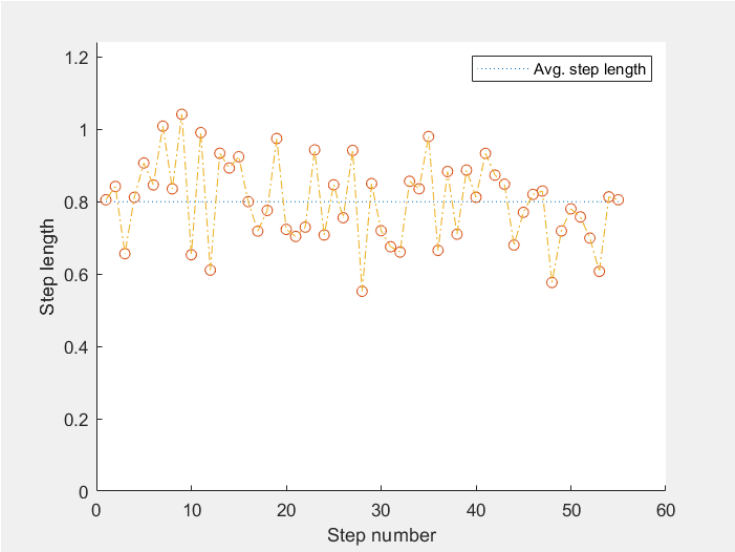
After having estimated the number of steps, the length of each of those steps is going to be estimated. To do so the Weiberg Step Length algorithm is used: first we take the vector of absolute values of the acceleration and we filter it with a low pass filter with cut-off frequency 3Hz (the fs variable is the sampling frequency):

```
Atilde = lowpass(A_abshorz, 3, fs);
```

Then the Weiberg step length is obtained as:

$$SL_{Weiberg_k} = K \cdot \{max_{j=[i_{(k)} \pm w]} \widetilde{a}_j - min_{j=[i_{(k)} \pm w]} \widetilde{a}_j\}^{1/4}$$

in which K is a constant that depends on the average step length, in our case the average step length has been set to 80 cm, that is the average step length for a male adult of average height walking at a medium to high pace. With this procedure the step lengths that have been found are here represented:

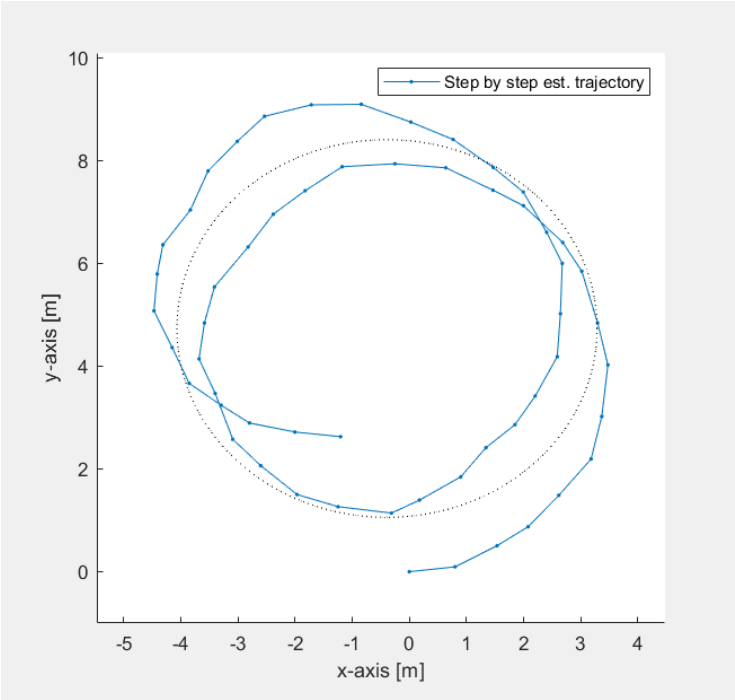


2.4 a.4)

Once the number of steps, as well as the length of those steps and the heading direction of the pedestrian in each time instant have been computed the time series of the subsequent positions can be obtained, starting from the *(0, 0)* point in the x-y plane, by adding the increment (or decrement) to the coordinates after each step. In MATLAB code this is equivalent to:

```
positions = zeros(2, N_steps+1);
for i = 1:N_steps
positions(1, 1+i) = positions(1, i) + SL(i)*cos(angles(3,
    vallind(i)-(length(binomialCoeff)-1)/2));
positions(2, 1+i) = positions(2, i) +
    SL(i)*sin(angles(3, vallind(i)-(length(binomialCoeff)-1)/2));
end
```

The trajectory that is obtained from these positions is:



In the above graph the estimated positions are compared to the circle that minimizes the sum of squared radial deviations from the trajectory points (the code for the LS circle was taken from MATLAB Central File Exchange).

2.5 a.5)

In this section the goal is that of implementing a Kalman filter to track the motion parameters of the moving pedestrian from the above computed parameters (positions, step lengths, heading directions).
The KF parameter vector is a 4x1 vector whose elements are estimates of the position and speed of the pedestrian on both the x and the y axes:

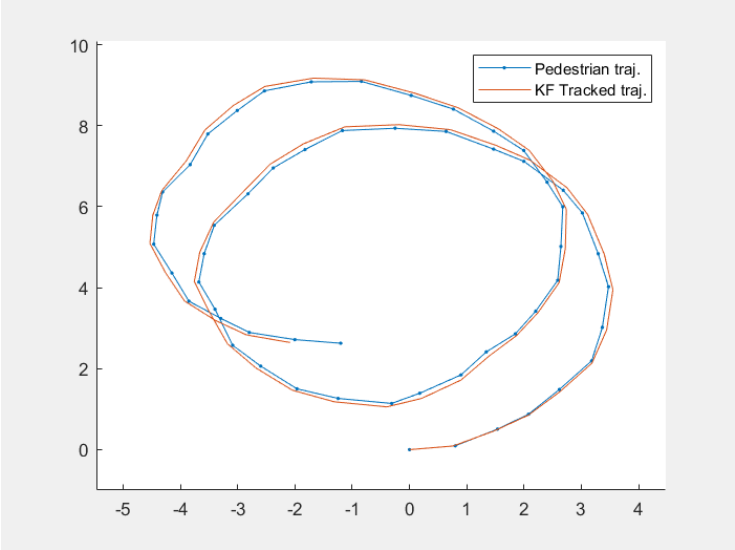
$$\theta = [x\ y\ v_x\ v_y]^T$$

The prediction matrix is:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

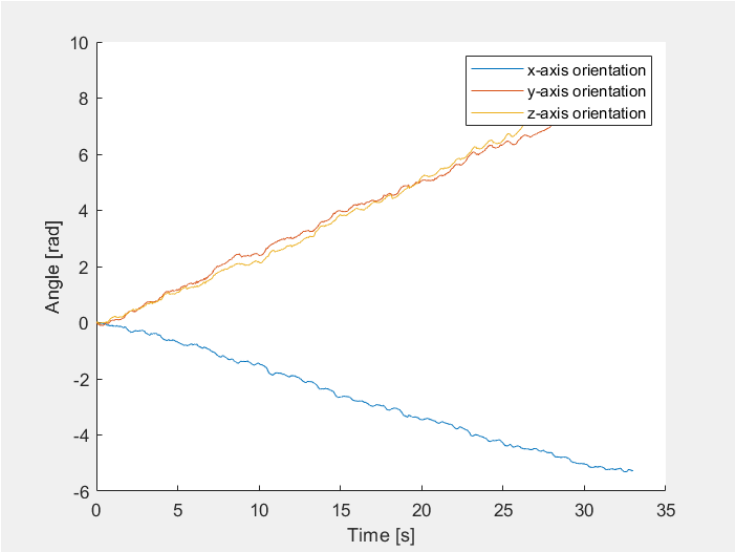
where T is the mean step duration, computed as total duration of the inertial parameter acquisition divided by the number of steps.
The Kalman filter implementation has been carried out with the usual iterative process (the number of iteration being the number of steps) consisting of the

following phases: prediction, measurement (in which the estimated positions are extracted) and update (with computation of the KF gain). By varying the covariance of the excitation the KF varies its ability to track the motion of the pedestrian with high accuracy.
With this KF the results that have been obtained are:

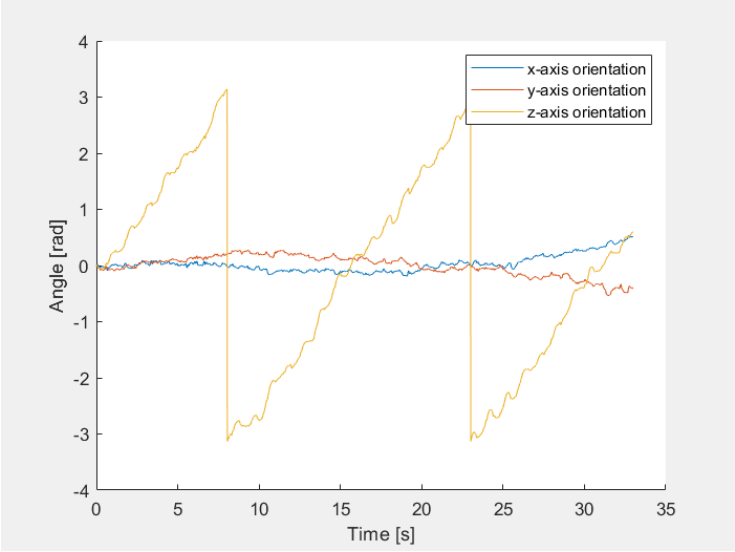


2.6 b.3)

For the second datasets of intertial parameters the first computation that has been carried out is the heading direction one. From an inspection of the angles obtained by integration of the angular velocity we can see that the phone is not held in a fixed, parallel to ground, position, because the z component of rotation is not the only non-negligible one:



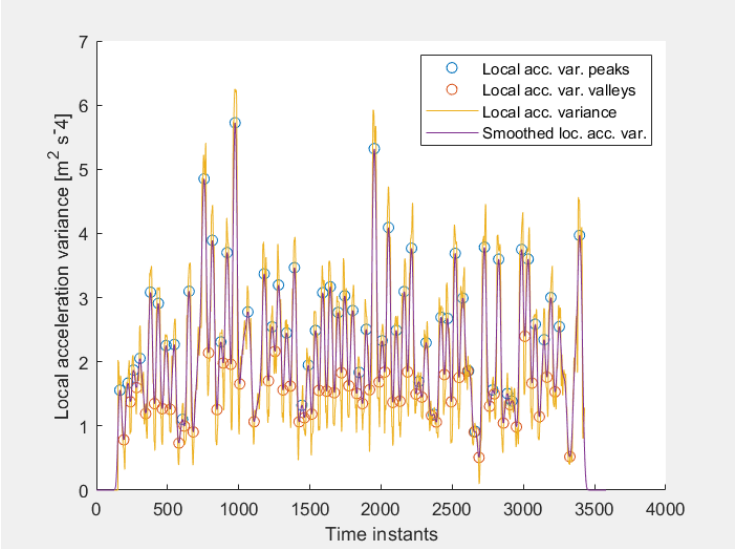
(in this graph the angles are not wrapped to the $[-\pi,\ \pi]$ interval).
So in order to correctly estimate the heading direction of the pedestrian we need to apply a rotation matrix to the inertial parameters data, that applies a rotation both on the x and on the y axis, after doing so the angles representing pedestrian orientation are:



This result is obtained by rotating the inertial parameters by 0.681 *rad* on the x axis and 0.569 *rad* on the y axis.

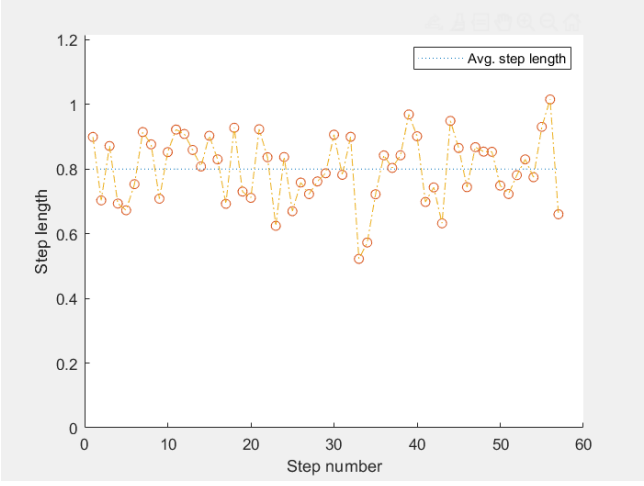
2.7 b.1)

The same rotations are applied to the acceleration data for consistency. The procedure to estimate the number of steps is the same as the one used for the first dataset, and the number of steps is estimated to be the number of valleys in the following graph:



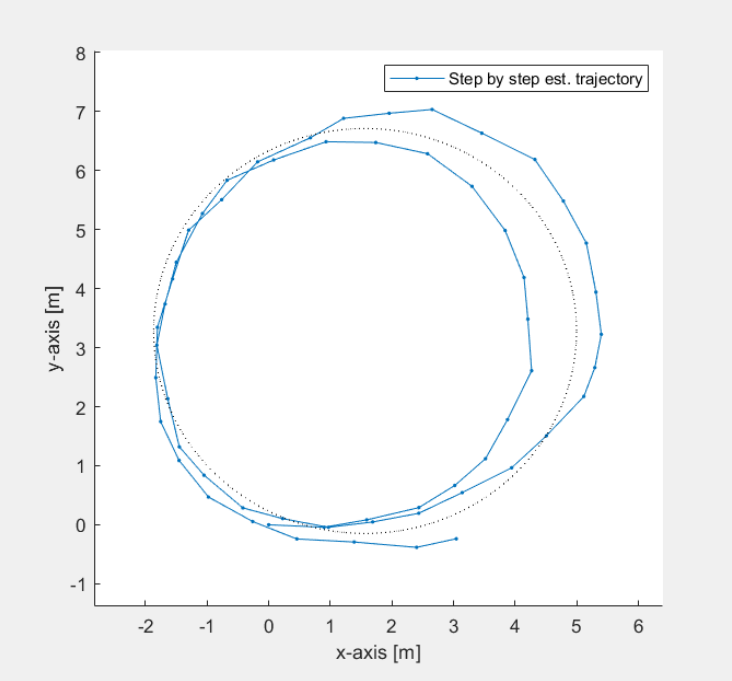
2.8 b.2)

Also for the step length estimation the procedure is the same as in the first dataset, once again the average step length has been set to 80 cm, the results are shown below:



2.9 b.4)

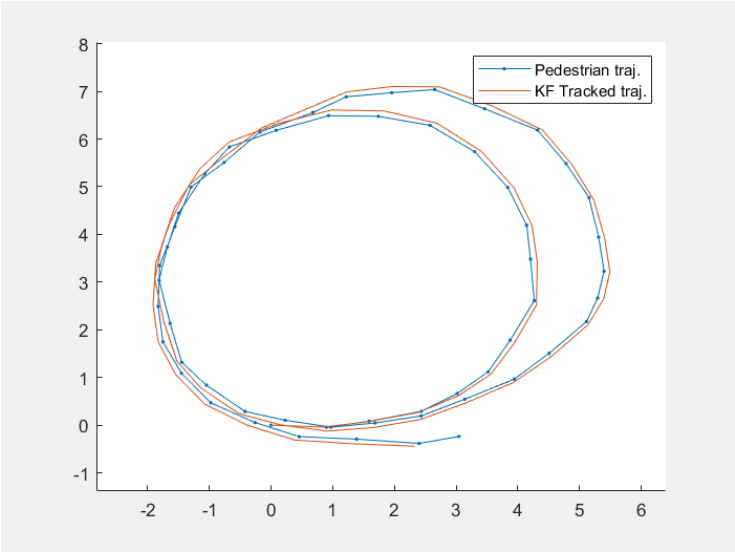
The positions were computed using the step length and the heading angle values previously computed, the trajectory that has been obtained is:



The trajectory is compared to the LS fitted circumference, and we get a closer fit to a true circumference in this dataset compared to the first one.

2.10 b.5)

The Kalman filter used for this second dataset is the same as the one illustrated for the first dataset.
Also in this case the KF was able to correctly track the motion of the pedestrian as estimated in the previous steps:



2.11 2)

The second part of this activity deals with the estimation of the evolution of the pedestrian's position carried out using Bayesian tracking based on the inertial parameters (accelerometer and gyroscope data only). For this purpose an extended Kalman filter has been used: the vector of parameters consists on position on x and y, absolute value of pedestrian speed, heading direction, angular velocity, and absolute value acceleration on the horizontal plane (z axis components are neglected):

$$\theta = [x\ y\ v\ angle_z\ \omega_z\ a_{xy}]^T$$

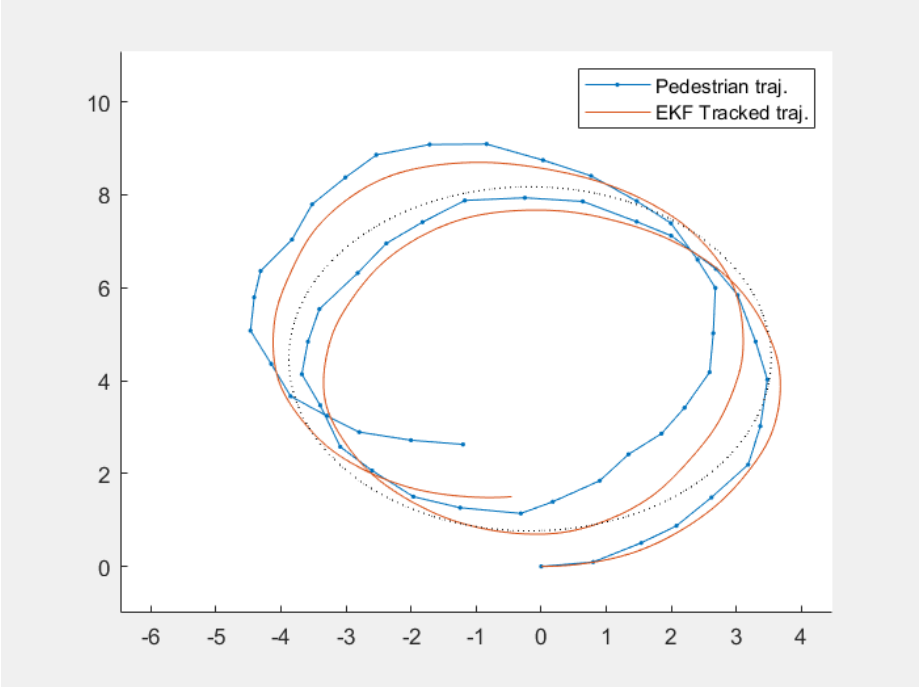
The F_k matrix used in the prediction step is:

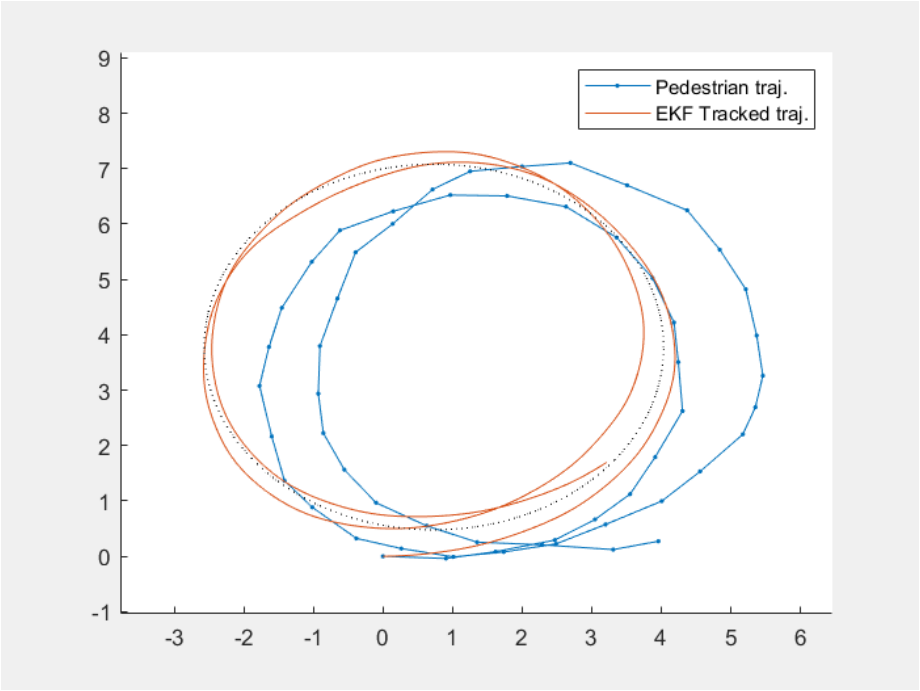
$$F_k = \begin{bmatrix} 1 & 0 & t_s \cdot cos(\theta(4)) & t_s \cdot sin(\theta(4)) & 0 & 0 \\ 0 & 1 & -(t_s \cdot \theta(3) + t_s^2 \cdot \theta(6)/2) \cdot sin(\theta(4)) & (t_s \cdot \theta(3) + t_s^2 \cdot \theta(6)/2) \cdot cos(\theta(4)) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_s \\ 0 & 0 & 0 & 0 & t_s & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

while the covariance matrix of the excitation is a 6x6 matrix that is obtained from the variance of the accelerometer and gyroscope data. The EKF iterative procedure is carried out with the standard steps: prediction, computation of covariance matrix of prediction, correction, computation of EKF gain, computation of covariance matrix of the MMSE estimate and parameters update. In particular the prediction step is not carried out with a linear model but with a nonlinear one (hence the choice of an extended KF), the relations being:

$$\begin{cases} \theta_n(1) = \theta_{n-1}(1) + (\theta_{n-1}(3)t_s + \theta_{n-1}(6)t_s^2/2)cos(\theta_{n-1}(4)) \\ \theta_n(2) = \theta_{n-1}(2) + (\theta_{n-1}(3)t_s + \theta_{n-1}(6)t_s^2/2)sin(\theta_{n-1}(4)) \\ \theta_n(3) = E[SL]/T + \theta_{n-1}(6)t_s \\ \theta_n(4) = \theta_{n-1}(4) + \theta_{n-1}(5)t_s) \\ \theta_n(5) = \theta_{n-1}(5) \\ \theta_n(6) = \theta_{n-1}(6) \end{cases} \tag{1}$$

Those relations specify the update of position on the x axis, position on the y axis, speed (abs. value), heading direction, angular velocity, acceleration. The latter two parameters are updated based on the error computed in the previous cycle(that compares the previous estimate from the true values obtained by the inertial sensors).
With the use of this EKF the trajectories computed for the two datasets are here reported:





We can see that in the first case the difference to the previously computed trajectory is narrower than in the second case. This can be attributed on the fact that the EKF based on the inertial parameters tends to be closer to a true circumference than the trajectory estimated with the first method. In the figure is reported (in black dotted line) the circle obtained with a least square fit to the EKF computed trajectory.
The EKF estimated time evolution of the pedestrian position has been computed to be $1.3795 m/s$ and $1.3619 m/s$ for the first and second dataset respectively, that is consistent with the *5 km/h* average walking speed (source: runtastic.com). If we compute the RMSE of the EKF estimated trajectories w.r.t. the LS fitted circle we get that the RMSE / LS fitted circle radius ratio is 10.2% for the first dataset and 5.08% for the second one, those percentages can be taken as a measure of the distortion of the estimated trajectories from a true circle. For instance the RMSE / LS fitted circle radius ratio is 12.8% and 9.7% (for the first and second dataset respectively) for the trajectory estimated with the combination of step length and heading direction.

3 Conclusion

We have seen in this activity how a preliminary data cleaning in the form of rotation may be necessary in order to get good results, and that given the accelerometer and gyroscope data several approaches are possible to compute the trajectory of the pedestrian. Since in this activity the (circular) shape of

the trajectory was explicitly stated in the assignment we can say that the EKF approach that estimates the trajectory directly from the inertial parameters yields more accurate results than the other methods, that are still quite good also considering their simplicity.

4 Bibliography

- Spagnolini U., *Statistical Signal Processing in Engineering*, Wiley, 2018
- Jimenez A.R., Seco F., Prieto C., Guevara J., *A Comparison of Pedestrian Dead-Reckoning Algorithms using a Low-Cost MEMS IMU*, 2009 IEEE International Symposium on Intelligent Signal Processing
- Wonho K., Youngnam H., *SmartPDR: Smartphone-Based Pedestrian Dead Reckoning for Indoor Localization*, IEEE Sensors Journal, 2015
- Svensson D, *Derivation of the discrete-time constant turn rate and acceleration motion model*, 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)
- Luise M., Vitetta G. M., *Teoria dei Segnali*, Milano, McGraw-Hill, 2009
- Izhak Bucher (2021). *Circle fit* (https://www.mathworks.com/matlabcentral/fileexchange/5557-circle-fit), MATLAB Central File Exchange.