LNSM Project

Localization Navigation and Smart Mobility – Prof. Nicoli, Prof. Brambilla – A.Y. 2021/2022

Temesgen Semu Balcha
10695128

Gaetano Ivan Ferrara
10865269

Stefano Maxenti
10526141

Eliana Mazza
10521024

***Abstract* — Given the TDOA measurements and the AP scenario setting, it is possible to process the data and then plot the AGV trajectory, by implementing both the Iterative NLS approach and the Extended Kalman Filter (EKF).**

Index terms* — *TDOA, iterative NLS, EKF

I. Introduction

The implementation of iterative non-linear least-squared (NLS) technique and the Extended Kalman Filter (EKF), allows us to compare them and plot the trajectory of the moving AGV.
The 4 tags on the object communicate with 6 Access Points (AP), getting different TDOA (Time Difference Of Arrival) measurements, then converted into distance measurements. The Master AP is the second one.
We implement the Jacobian Matrix for the TDOA measurements and for the model itself.
The project focuses on localizing the device from the measurements, and to prove that the Kalman Filter outperforms the Iterative NLS output.

II. Data analysis and preprocessing

Given all the TDOA measurements referred to the 4 tags, we start from the assumption on the slide that the sampling rate associated to all of them is 10 Hz (one sample every 0.1 s). We notice that one of them contains only 262 samples, while two had 665 and the last one 663.
In order to avoid issues with missing values, we consider three of the four tags up to their 663rd measurement because the remaining tag is initially supposed to run out of battery after 262 timesteps, that is about 26 seconds.
Since the AGV is moving on the horizontal plane, we do not consider the z-coordinate of the APs.
The main problems to deal with are NaNs values, outliers and noise.
At first, we decide to deal with the NaN issue by calculating for each tag the median value over each column of data (corresponding to the same time instant) such that we could just replace the invalid value with the new one. The median function allows to avoid outliers in the calculations as well.
After the first iterative plots (Figure 2) we discover that the assumption about the first tag is wrong: it does not stop working after 26.2 seconds, but it works until the end. The reason of the fewer samples is found in the different sampling rate, which we estimate around 2.5 Hz (one sample every 0.4 seconds).
We then decide to change approach and simply replace each NaN with the previous valid sample. We are aware that in this way there will be time instants for which the AGV seems to be steady in a spot and then suddenly moves to a not-too-far one, but since the sampling time is very narrow, we decide to take the risk.
The initial median approach, though based on the wrong premise and therefore removed from the final version of the project, worked adequately because of its ability to filter out outliers, so that even if one tag is constantly wrong, it can be balanced by the other three contributions.

After plotting the TDOA graphs, we see a great amount of noise.
We want to compare the performance of the NLS and EKF algorithms on the raw data versus the performance on further filtered data (namely, an offline denoising).
To do that, a Savitzky-Golay filter is used to approximate the samples' behavior to a polynomial function (we choose order 1 – linear function – and 11 points as sliding window – a bit more than 1 second of sampling).

An example of the obtained result for one tag's TDOA is shown in Figure 1. All other graphs are in the graph folder.

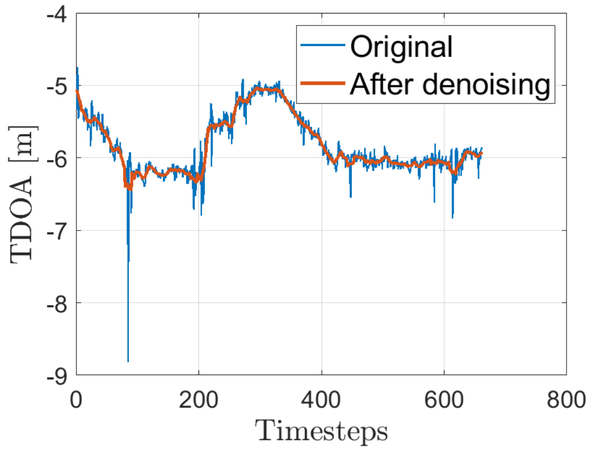

Figure 1

It is important to notice that it would be impossible to use this last procedure in real-time tracking, as the filtering requires information on all the points of the trajectory. However, it could theoretically improve performance in offline tracking.
To use the further processed dataset, it is needed to change the *denoise* boolean variable to *true* in the MATLAB files.

III. Iterative NLS

The estimate of the position, for each timestep, is found by minimizing the square modulus of the difference between the measurement and the model.

$$\hat{u} = \underset{u}{\operatorname{argmin}} |\rho - h(u)|^2 = \underset{u}{\operatorname{argmin}} \sum_{i=1}^{N} \left(\rho_i - h_i(u)\right)^2$$

In our case, measurements are TDOA and therefore the model is given by the intersection of hyperbolas.
This is a valid solution when we suppose the errors affecting the measurements as gaussian and uncorrelated. Since in general no closed-form solution can be associated to the non-linear localization problem, iterations are necessary. We use 1000 iterations for each step. In addition to that, in each step we start from the latest found position in the previous one. Actually, NLS does not need this help to find the position, but it could improve performance.

It is possible to see the four tag trajectories in Figure 2.

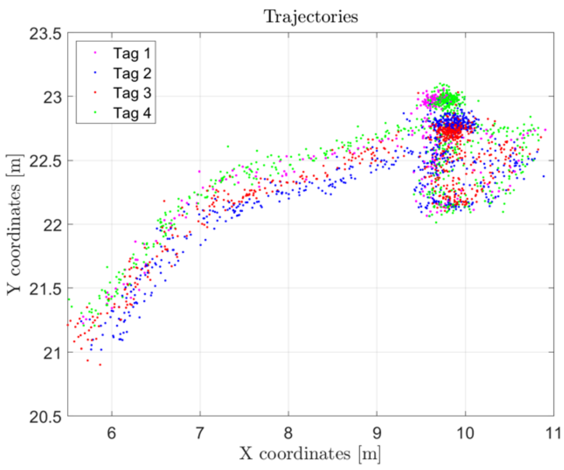

Figure 2

The usage of the offline filtering approach leads to much better results, as shown in Figure 3.

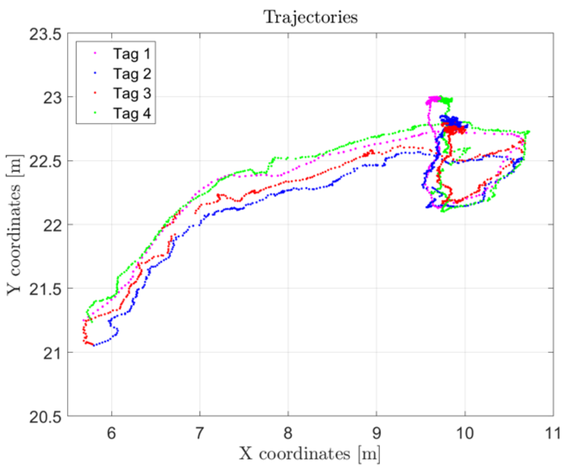

Figure 3

As anticipated in the previous section, we notice that all the four tags are able to draw the entire path. This is the reason why we conclude that there is a tag affected by a bigger sampling time. Indeed, by running an animation of the trajectory in MATLAB, we notice the Tag 1 to be "anticipated" with respect to the others.

It is easy to see from the plot that the trajectory is still quite noisy. We believe a Kalman Filter to be a better solution to provide localization of a moving object.

IV. Extended Kalman Filter

A Kalman Filter works by considering the previous time-step predicted states, to find a realistic trajectory, avoiding many errors due to multipath effect.
This approach relies on two models: the measurement and the motion model. According to the second one, we decide to use the Random Walk model.
The equations that describe it are the following:

$$x_t = u_t = \begin{pmatrix} u_{x,t} \\ u_{y,t} \end{pmatrix}$$

$$\begin{cases} u_{x,t} = u_{x,t-1} + Tw_{vx,t-1} \\ u_{y,t} = u_{y,t-1} + Tw_{vy,t-1} \end{cases}$$

$$w_{v,t} \sim \mathcal{N}(0, \sigma_v^2 I_2)$$

In our specific scenario, the matrixes of the Kalman Filter have the following sizes:

H: [5x5]; R: [5x5]; F: [2x2]; Q: [2x2]

Since the system equations are not linear but the noise is still considered as gaussian, we decide to implement an Extended Kalman Filter.

We are able again to plot the final four tag trajectories in Figure 4.

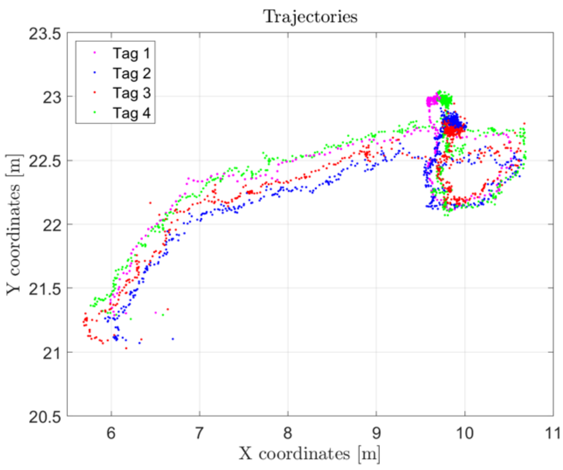

Figure 4

By comparison, we can easily see that our assumption about the improvements in the usage of a Kalman Filter is correct, being it much less noisy than before.
The usage of the offline filtering approach leads to slightly better results, as shown in Figure 5.

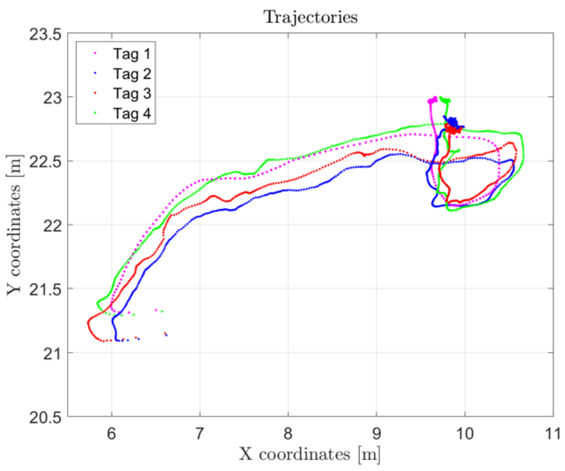

Figure 5

V. Conclusions

We decide to plot the mean of both Iterative NLS and EKF outputs of all the four tags, as to represent the unique trajectory of the AGV.
Each pair (x_t, y_t) is obtained as the mean of the positions of each tag at time instant t.
To avoid mixing samples taken at different time instants, we approximate the sampling rate of the mistaken tag to about 1/4 of the others, as described before.
The final result is shown in Figure 6.

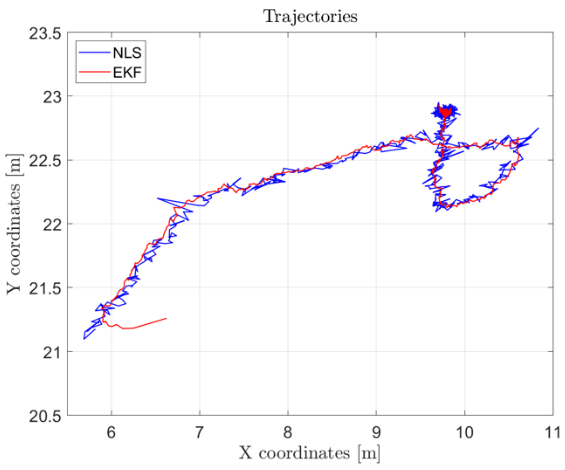

Figure 6

In our experiments, we see that Iterative NLS is always noisier that EKF and in a real tracking scenario EKF is more reliable due to considering the previous time-states.

According to MATLAB code implementation, the EKF is also faster and requires much less iterations than the other technique.

The biggest drawback of EKF is its inability to immediately find the starting point: it takes some timesteps to converge to the real position. NLS could be used instead to find the correct starting point before relying on EKF.

As anticipated before, while the offline denoising provides better results, it is not possible to use it in real time but after a necessary delay.

As a comparison, the final mean trajectory using the offline approach is shown in Figure 7.

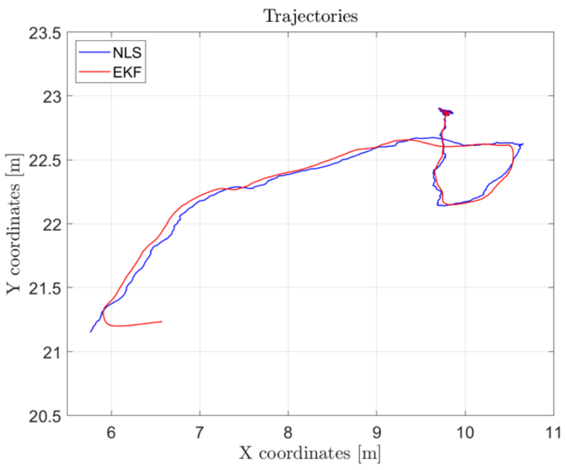

Figure 7

It is interesting to notice, though, that the mean of the results of Kalman Filter is very similar to the offline denoising version.

Overall, for real-time tracking, the Kalman Filter is the best option.