Danmarks Tekniske Universitet

DTU

RF-based positioning for Unmanned Aerial Vehicles

Author

Ernestas Simutis - s212571

Supervisors

Matteo Fumagalli
Jeppe Heini Mikkelsen
Peter Iwer Hoedt Karstensen

January 13, 2023

Contents

1 Goals

- Investigate different RF technologies for positioning and evaluate their pros and cons, e.g. WiFi, UWB, BLE, etc.
- Implement an RF based range measurement method
- Compare range measurement using RSSI (Received Signal Strength Indication) and RTT (Round Trip Time)
- Implement a Bayesian filter for positioning using aforementioned range measurement(s)

2 RF-based ranging

RF (radio frequency) based ranging problem is concerned with inferring distance between two agents/robots in space using radio antennas.

The chapter is divided in a following way: first describing general methods of measuring distance between two RF devices/antennas, secondly investigating existing protocols making measurements and lastly selecting a technology for later implementation of positioning system based on ranging technology.

Requirements for application:

- Long range - preferably up to 100m or more (so that agent can cover bigger area).
- High precision - accuracy of at least 10cm at short distances (in order to avoid collisions whenever two agents would be close to each other).
- Measurement frequency of at least 10Hz or more (this allows agent to execute faster trajectories).

2.1 Range measurement methods

Literature points out two main methods for making distance measurements over RF antennas: RSSI (Received Signal Strength Indication) and time based methods. There is also approaches based on angle of arrival (AoA) but it's not explored here.

2.1.1 RSSI

As the name suggests the received signal strength hints at the power of incoming radio signal. There is non-linear corelation between RSSI and distance between two antennas [1]:

$$RSSI = -10n \log_{10}(d) + C$$

thus it is possible, for instance, to fit a curve on a measured experimental values like shown in [1]. However, the main disadvantage that for commercial protocols the quantity over distance curve flattens out at certain distance and further it's impossible to tell the a difference between different measurements. The effect can be seen in Figure 1 for indoor applications (meaning having obstacles in a path). The figure shows plots of RSSI vs Distance plots for different protocols like WiFi, BLE, etc.

Also in [2] authors provide line-of-sign scenario results of BLE and WiFi RSSI from a commercial phone. Figure 2 shows their results on measuring RSSI multiple times i.e. data is collected at different times. The variation between different BLE data points is relatively huge and influences range accuracy is a significant way. Both BLE and WiFi curves flatten out as distance is increasing and it's evident that above certain limit signal strength is not usable as a viable distance measurement. Also there is quite a bit of variation, curves are not smooth so the precision of data points is quite low.

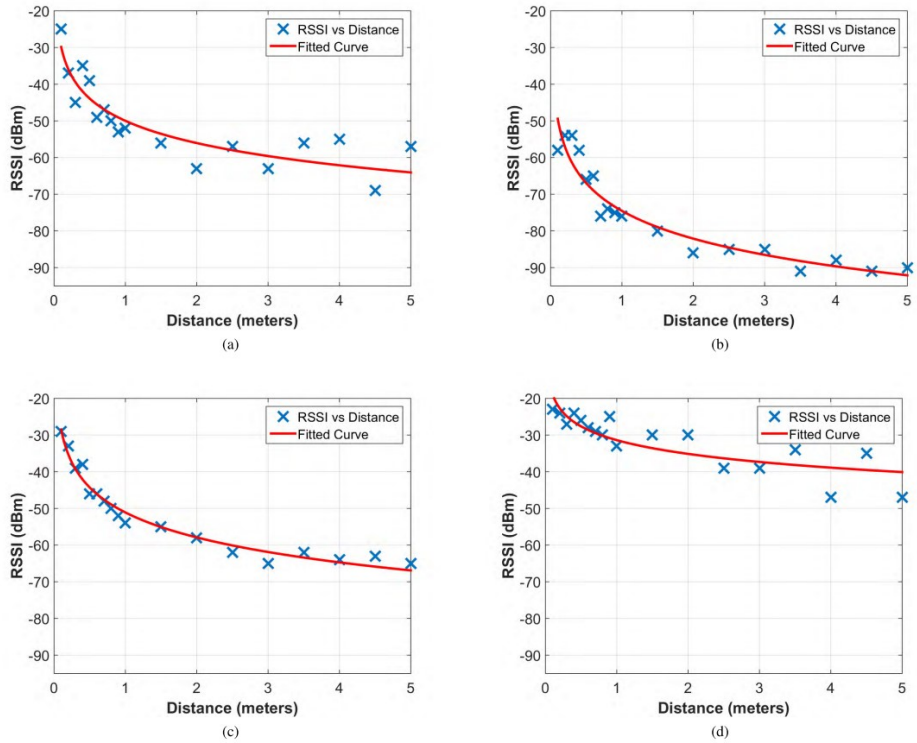

FIGURE 6. Curve fitting for the path loss in environment 1. (a) WiFi. (b) BLE. (c) Zigbee. (d) LoRaWAN.

Figure 1: Various protocol RSSI over distance curves [1].

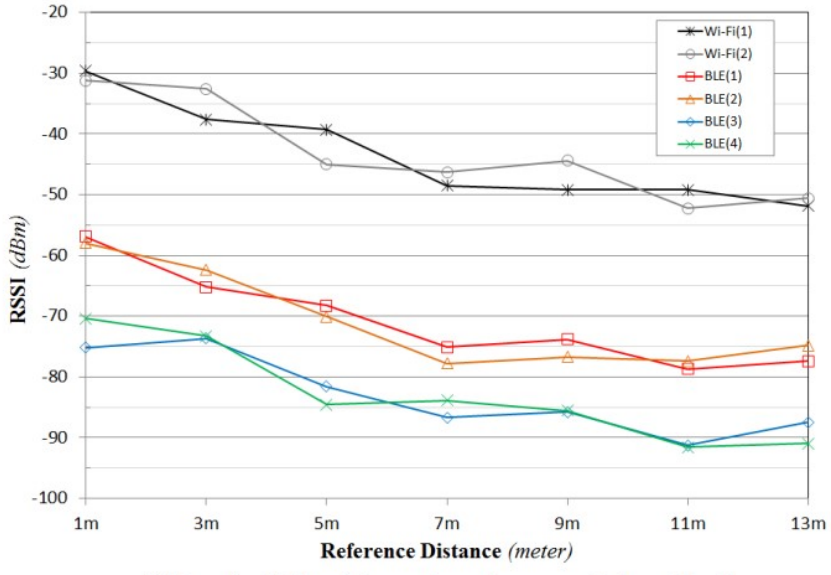

Figure 2: BLE RSSI curve on Samsung Galaxy Note3 [2].

2.1.2 Time based

These methods are based on measuring radio wave propagation time between (or ToF - *Time of Flight*) antennas/devices and calculating the distance by knowing light speed i.e. $d = ct$, where $c = 3 * 10^8$ m/s. One obvious drawback here is that two clock must be synchronized if to infer the distance by ToA (*Time of Arrival*). However, this can be avoided by doing a round trip (in literature referred as TDoA - *Time Difference of Arrival*) and relying solely on one clock. The scheme is illustrated in 3 where T_{prop} is propagation time in which we are interested in for ranging applications. The value can be computed by $T_{prop} = \frac{1}{2}(T_{round} - T_{reply})$. Also, Figure 3 correctly indicates the scale of T_{round} and T_{prop}. Important thing to notice here, if the distances we're measuring are in order of meters, $T_{prop} = d/c$ evaluates to nanosecond scale while for response computer might have to execute hundreds of instructions taking up way more time than the time we're interested in. Thus it's of most importance to know how much time computation takes in the responding device. When selecting/designing a system to use RTT for positioning, responding device must have very fine granularity control of computing resources. If one would try to make these computations on OS (Operating system) level, the time jitter introduced by OS scheduling system would be so large that propagation time would vanish in the error of T_{reply} measurement.

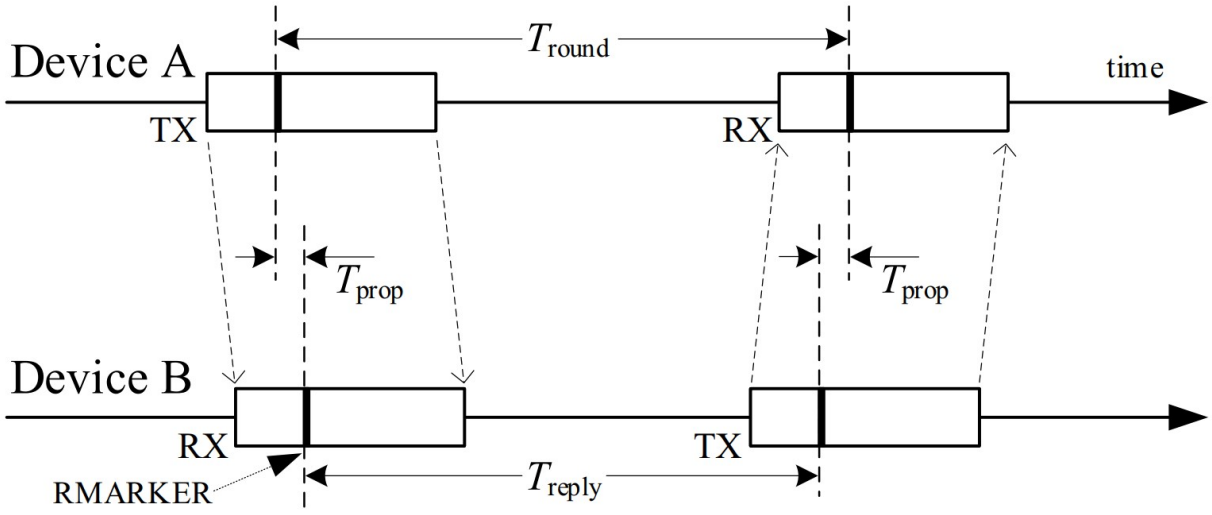

Figure 3: Round trip time [3].

2.2 RF positioning

There are many existing protocols/technologies to exchange data between devices over the air i.e. by transmitting it over electromagnetic waves. In this chapter, a few of most popular ones will be reviewed in terms of how suitable each of them is for positioning applications. WiFi, UWB (Ultra Wide Band) and BLE (Bluetooth Low Energy) protocols are investigated.

All of the below described protocols do support RSSI ranging (by the nature of them

being radio communication the signal strength assessment can always be made) but due to limitations the focus is on on time based ranging methods to fulfill the application requirement with selected technology.

Although conceptually calculating distance from time it took to traverse the space for a radio wave is very simple, implementing it in reality is very difficult because time stamping must be done at a hardware level to have reasonable precisions when distance equation involves light speed. For instance, $1\mu s$ corresponds to $300m$ difference in distance measurement. Thus significant effort is spent to develop standard protocols and techniques to make this work in practice.

2.2.1 WiFi

IEEE 802.11-2016 standard includes a Fine Time Measurement (FTM) protocol for WiFi ranging technique. This allows very granular measurement of time, enabling time-stamping WiFi transmissions (obtaining RTT) precise enough to calculate distances between devices. However hardware support is very limited even for high-end WiFi products.

Even with specialized (expensive) hardware this approach requires quite a bit of effort to install required driver, firmware, kernel version. The distance measurement is also not that precise - best case scenario gives $1 - 2m$ accuracy.

For instance, in paper [4] authors were able to make it work while getting mentioned accuracy and collecting data point from devices up to couple hundred meters apart. Nevertheless, the used hardware is not readily available for purchase (they mention a chip used, but quick internet search doesn't give any practical answers where to buy antenna with the exact chip on it) or requires one to buy full router (costing hundreds of dollars for each device).

2.2.2 BLE

BLE protocol is widely supported and available even on low-end cheap devices like Arduino microcontroller. The main problem is that no standard way of measuring RTT is present in the protocol thus the only only option to get a range measurement is to use RSSI which was already discussed as not suitable for the application/requirement of the project.

2.2.3 UWB

UWB is one of the most widely used radio technology for positioning applications because of the wide support for for range measurement and existing protocols to do that. Many chips sets are present on market with off the shelf distance measurement capabilities (like DW1000 [5] or U1 in Apple products). This shows the extent to which technology is matured enough to be used in real-world applications.

Standard IEEE Std 802.15.4z-2020 [3] specifically defines ranging techniques that can be readily implemented in hardware and software of UWB supporting microcontrollers. Method described in the standard is based on RTT with additional measures to increase robustness and precision in practical implementations. However, in essence round trip time approach

is selected because of no need to calibrate the clocks on devices and if time measurements are done in a precise way the distance calculation is very straightforward.

Due to convenience, availability and other properties that meets the requirements UWB based ranging is select to be used in special course positioning experiments as most suitable protocol.

2.3 Other

There are many RF communication protocols that might be promising for positioning applications. For example, LoRa has potential in long-range low-accuracy positioning application or 5G could be also be considered as alternative approach with similar performance to UWB although not that widely adopted yet. However, these were not investigated in detail here.

3 Positioning

With enough distance measurement to a number of known static or dynamic beacons/landmarks one can estimate agents position in space. Chapter covers application and implementation of EKF (Extended Kalman Filter) for estimation of 3D position using range measurements in space and writing a simulation to validate the filter before moving to real-world experiments.

3.1 EKF

Due to sensor measurement noise state estimator is necessary. Thus Kalman filter will be implemented and deployed to track agents position assuming noisy observations of distance. Before going into details it's important to think about state evolution and observation models because Kalman filter assumes both to be linear. During project constant velocity model will be used, described by:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{z}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ z_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{z}_k \end{bmatrix}$$

or

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}\boldsymbol{x}_k$$

which is linear by itself. $\boldsymbol{F}$ here comes from discretization of continuous time constant velocity model matrix $\mathbf{F}_k = e^{\mathbf{A}\Delta t}$

If it were a system like drone model would be non-linear, more complex and include actuation signal in state transition. Yet the project is aiming at investigating plausibility of localization with UWB antennas rather and experiment will be done by carrying agent device by foot. This cannot be modeled and can be treated as a black box constant velocity model.

However, Euclidean distance measurement model (from beacon to agent) has non-linearity due to the *norm*:

$$\|\boldsymbol{x} - \boldsymbol{b}\|_2 = \sqrt{(\mathrm{b_x} - x)^2 + (\mathrm{b_y} - y)^2 + (\mathrm{b_z} - z)^2}$$

where $\boldsymbol{x} = (x, y, z)$ is vector representing agents position and $\boldsymbol{b} = (b_x, b_y, b_z)$ is position of one visible (in radio communication sense) beacon in space. This is where EKF comes into play, the method proposes to use linearized version of non-linear function by taking it's Jacobian and assuming that linearized model representation around a small deviation neighborhood holds true and usa that in place of the measurement matrix $\boldsymbol{H}$. This will be described later but, in short, one can take partial derivatives of the function with respect to each agent's

position dimension and evaluate them at current state estimate. For instance:

$$\frac{\partial h(x)}{\partial x} = \frac{x - \mathrm{b_x}}{\sqrt{(\mathrm{b_x} - x)^2 + (\mathrm{b_y} - y)^2 + (\mathrm{b_z} - z)^2}}$$

General discrete time EKF is defined as described in the following paragraphs, based on [6] and [7]. Starting with notation: $\hat{\mathbf{x}}_{k|k-1}$ represents the estimate of $\mathbf{x}$ at time instance k given observations up to and including at time $k-1$.

Prediction step

$$\hat{\boldsymbol{x}}_{k|k-1} = f\left(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k\right)$$
$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k$$

Update step

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h\left(\hat{\boldsymbol{x}}_{k|k-1}\right)$$
$$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k$$
$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1}$$
$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$$
$$\boldsymbol{P}_{k|k} = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k\right) \boldsymbol{P}_{k|k-1}$$

$\boldsymbol{h}$ here is non-linear measurement function, $\boldsymbol{H}$ - jacobian of measurement function with respect to state variables, $\boldsymbol{R}$ - measurement covariance, $\boldsymbol{z_k}$ - actual measurement and $\boldsymbol{K}$ being Kalman gain. If our measurement measurement covariance is small (way smaller than process noise covariance) i.e. we bought an expensive sensor with precise capabilities update step it will trust the measurement way more and $\boldsymbol{K}$ gain will be large. If it's the other way around filter will rely on model rather than trusting noisy sensor. These are two tunning nobs for the filter. State transition $\boldsymbol{F}$ and measurement $\boldsymbol{H}$ matrices are partial derivatives (*Jacobians*) of state evolution and observation model functions:

$$\boldsymbol{F}_k = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k}$$
$$\boldsymbol{H}_k = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k|k-1}}$$

In constant velocity case $\boldsymbol{f}(\boldsymbol{x})$ is already linear so that part can be discarded i.e. only $\boldsymbol{H}$ is required. Distance measurement Jacobian can be computed in code by:

```
1     def getH(x_op, beacons):
2         H = np.zeros((len(beacons), len(x_op)))
3         for i, b in enumerate(beacons):
4         H[i][:3] = ((x_op[:3] - b.get_pos()) \
5             / np.linalg.norm(x_op[:3] - b.get_pos())).T
6         return H
```

function takes in agent position at time $k-1$, beacon list and return partial derivatives of state variables with respect to each of the beacons. It's assumed that beacon number is fixed to 4.

3.2 Simulation

Figure 4 shows a trail run in simulation. Please note that system state is includes the third dimension however for plotting purpose only first two components are shown. The setup is running EKF described in previous section on noisy measurement. Noise is provided by adding a random sample from Gaussian distribution to each distance measurement at every timestamp. Big X is marking the start of a trajectory, GT line is ground truth trajectory generated by applying state transition matrix at each instant of time (moving at arbitrary speeds to cover a rectangle), blue path is the predicted one and lastly the numbered dots represent beacons arbitrarily placed in space. Additionally, Figure 5 shows similar simulation scenario plotted in three dimensions.

The code used for implementation of filter and generating plot etc. is uploaded together with the report. Simulation specific part can be found in `simulation.py`.

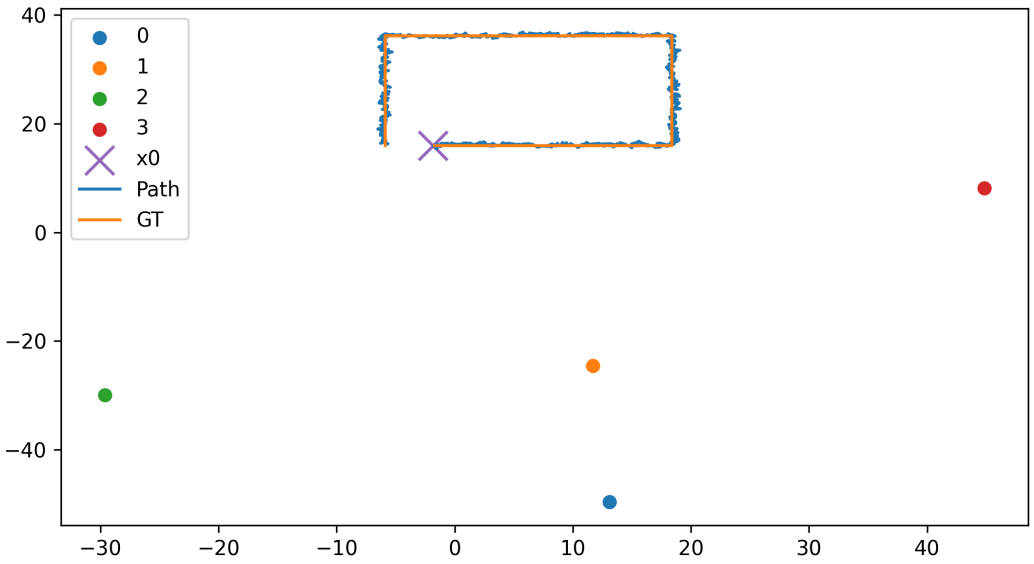

Figure 4: Simulation

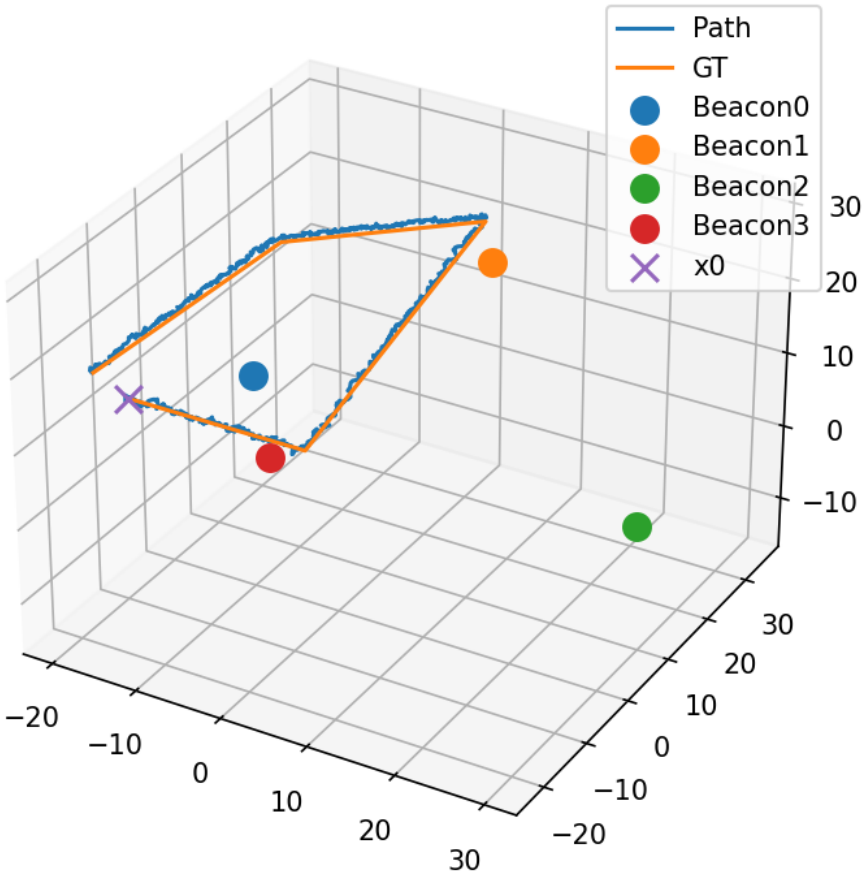

Figure 5: Simulation 3D

4 Experiments

Practical experiments are conducted during the course are discussed here. That includes: tuning KF (Kalman Filter) parameters (mainly evaluating covariances) by real-world experiments, testing out whole system and evaluating positioning accuracy developed in previous chapter. Code for conducting experiments and analyzing data is in the attached *zip* file and later will be referenced in text.

4.1 Setup

Experiments are done with microcontroller module from Makerfabs [8]. The board has ESP32 microcontroller and DW1000 UWB antenna chip and provides simple yet effective way to quickly implement/test application utilizing UWB distance measurement. They also provide ready to use code example that can be readily uploaded to microcontroller. It include a library abstraction around the ranging functionality of DW1000 chip. The distance measurements are forwarded to serial port and information is extracted on a laptop for storing and processing later on. Each device can be configured as an anchor or a tag, former being beacon sending distance data to a tag (receiver). Later, a rosbag is recorded in a PC to collect measurements from all the beacons in real time. This is done by parsing serial output of a tag device and publishing these measurement as a ROS topic. There is only one tag and N beacons, the tag also has a marker attached to it that Opticon system can track and publish the ground truth of an agent in real-time as a ROS topic. The mentioned topics are recorded in a single bag so that raw data can be replayed later to test positioning system on a recorded data multiple times offline.

DW1000 chip has many operating modes which have different modes optimized for different properties such as measurement update rate, accuracy and application specific operating distance. The one selected in the experiments is for accuracy, sparse update rate and moderate distance. Every of them have different tradeoffs and must be configured depending on case by case basis.

4.2 Results

First, it's important to address the assumption made in the simulation environment. It was assumed that sensor measurement variance is known. For filter to have good convergence properties we would like to know it up to a reasonable accuracy level when in operation. Therefore, the first experiment was conducted to measure the precision of UWB range measurements compared to a ground truth. DTU ASTA's Opticon system was used to generate ground truth labels (limited to 10m range) and two UWB devices, one configured as a tag and another one as an anchor. During the experiment anchor was moved to different position around the track, ground truth distance calculated between devices by taking norm of two position from Opticon and corresponding measurement by the means of UWB antennas was recorded. Figure 6 shows the probability distributions of measured values by real-world experiments in blue and the ground truth in orange.

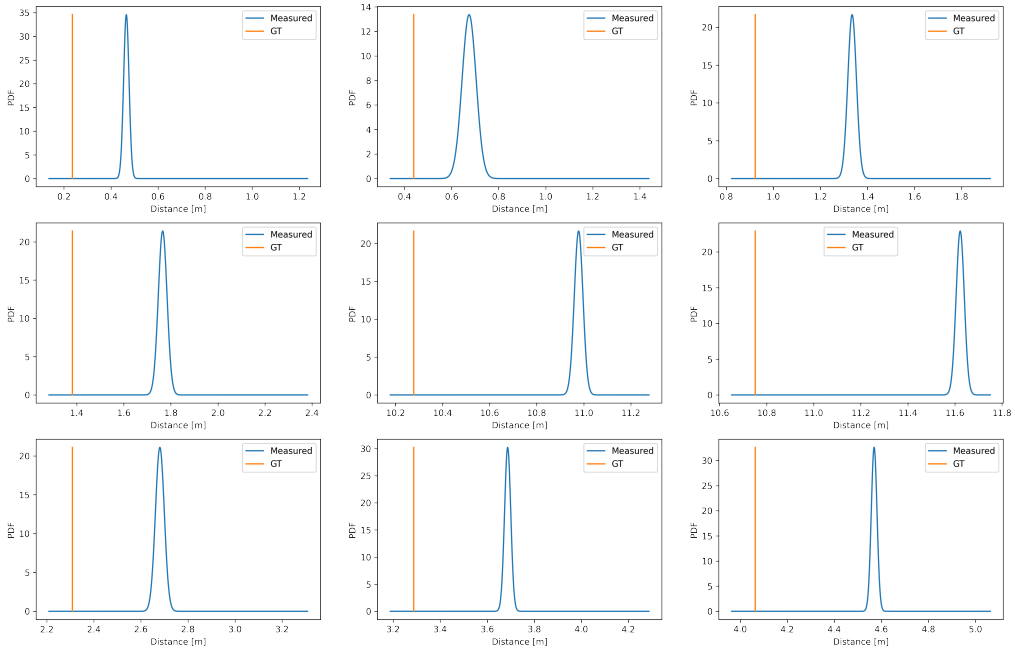

Figure 6: Distance measurements PDFs.

Looking at the plots one noticeable thing is that mean value of measured values are shifted to the right - meaning sensor gives out bigger distance than it actually is, this could be called bias. The relationship of bias and distance is illustrated in Figure 7. Additionally, it can be modeled, at least in this rang, by a line fit, which is shown in the graph too. It approximates the bias reasonably well and will improve localization accuracy in this distance range. In a way, it's calibrating the sensor so that measurements have the same mean value as ground truths.

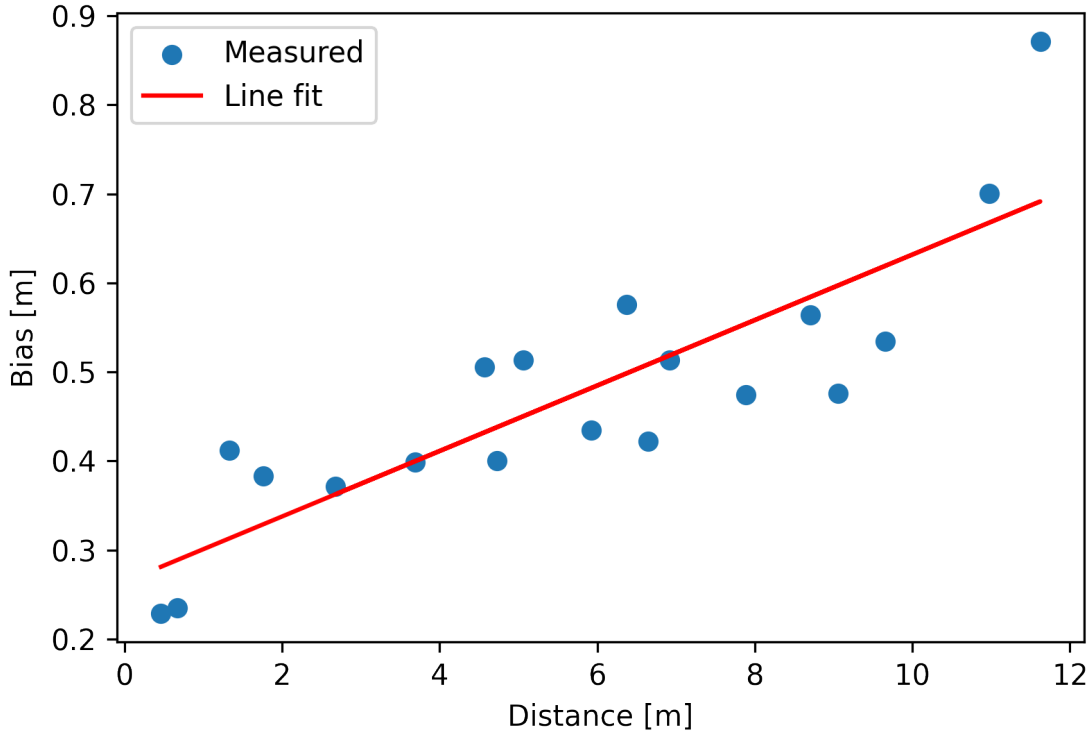

Figure 7: Measurement bias over distance.

Next, let's look at the variance and distance relation. The question to ask here if they are dependant on each other or we can use single constant values for all range measurements. Figure 8 shows them on single plot, plus a line fit on the data (which is, of course, bad representation of data because of one outlier). It's clearly seen that variance is very low and of almost same magnitude through out the data points. Thus, we can conclude that under tested conditions constant variance value can be used in EKF. For instance, an average value of all these variance points.

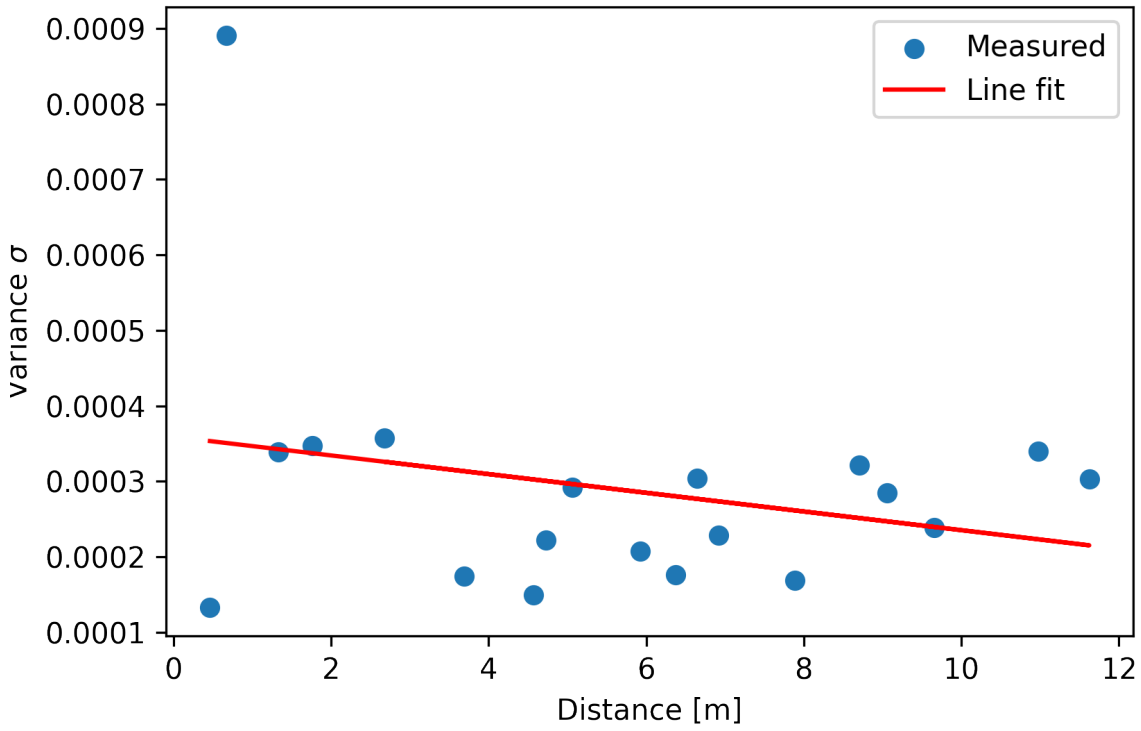

Figure 8: Distance over measurement variance.

Finally, a real experiment was conducted at DTU ASTA with Opticon system being ground truth and the setup closely following the one done in simulation i.e. having four beacons (or anchors) around an area. The tag/agent device was carried by hand to collect distance measurement from each of beacons simultaneously. Code used for data processing can be found in the root directory of repository in *experiment.py* script.

It was difficult to place the beacons in different heights because of the terrain/environment thus it's expected that z component of position estimate will have high variance due to lack of information. Still for validation of prototype even looking at two planar components is enough to evaluate the plausibility positioning approach using UWB antennas. At later stages this could be accounted online depending on beacon positions and possibly having more beacons to collect more information about agents position.

Also worth noting that measurement frequency is 10Hz for each beacon and connection to beacons is not stable through out the experiments, therefore measurements can be delayed or not arrive at a short enough time interval to make an update combining multiple measurements. This hints at a bigger issue - how to use data incoming at different time instances. In this case it's simply handled by a condition statements, saying that if measurement are less than some arbitrary Δt apart in time they are considered to have occurred at the same instance in time.

Later, EKF is applied to the data colleted to reconstruct the path taken by the agent,

I was trying to walk in an infinity symbol pattern. The results are depicted in Figure 9 as a planar 2D visualization with *Path* being the estimated position, *GT* - ground truth (Opticon), *BeaconX* - beacon positions and $x0$ - the starting point of the trajectory. Figure 10 show the same trajectory in 3 dimensions.

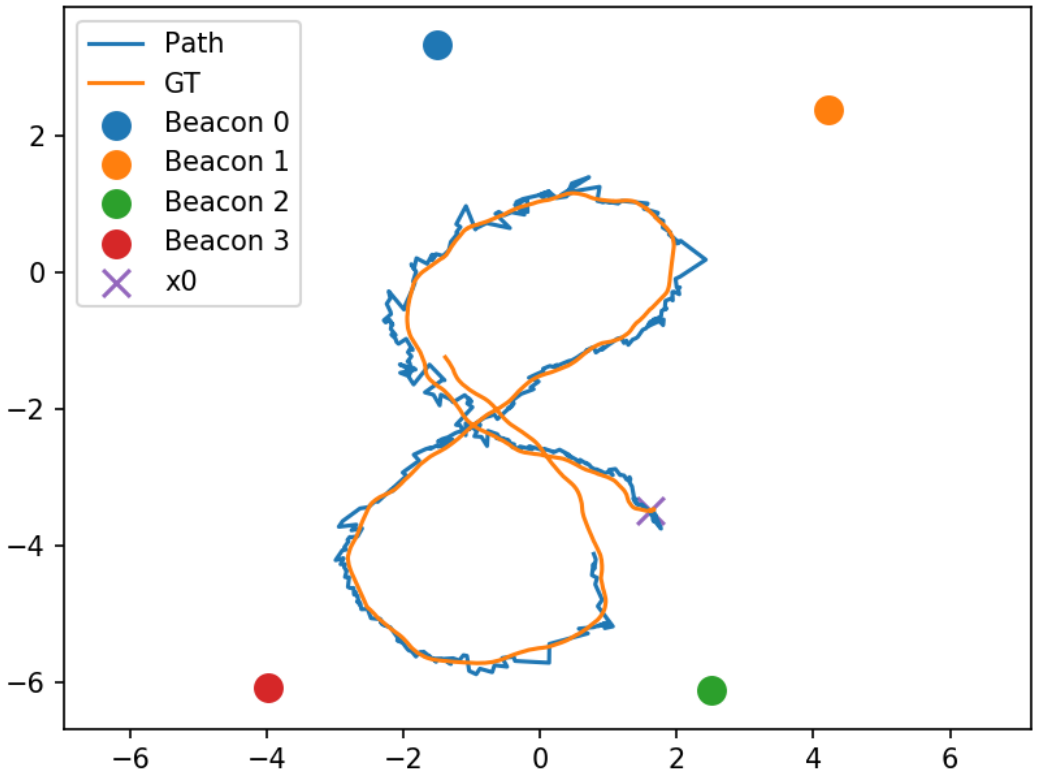

Figure 9: 2D plot of experiment trajectory.

You can notice that trajectories are of different lengths, it's just a residual of processing because two are incoming at different rates on ROS topics and should not be mistaken as filter estimates lagging behind.

The estimated trajectory has very sharp jumps. This comes from the fact that constant velocity model used to update estimate in between measurement is very bad representation of movement in the real trajectory, especially on turns (where path is not a straight line) and whenever valid measurement arrives the estimate abruptly jumps to correct the position based on range data from beacons. When used on a real system (for instance UAV) with better system model and known input the path would be way more smooth.

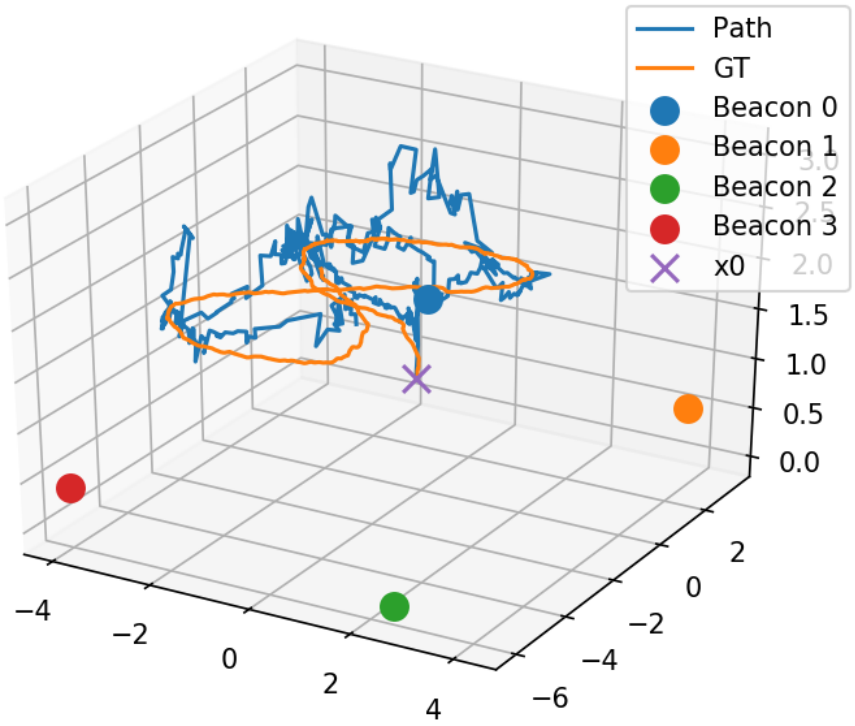

Figure 10: 3D plot of experiment trajectory.

Futhermore it was observed that waiting for 4 measurements from all beacons to arrive at the same time is not optimal meaning if the state is updated from at least 3 measurements it already contains enough information to inform the filter about true position of agent. While doing updates from less than 3 range data points at a time was observed to negatively influence filter performance.

As expected the z dimension is hardest to keep track because all of the beacons are almost in the same plane and give less information about height compared to other two directions.

Figure 11 depict the same experiment trajectory but with addition of EKF covariance plotted on top of the estimations. Important piece no notice here is that ground truth path is always in the ellipse of covariance, meaning that uncertainty accurately evaluates that estimate could be off by a certain degree and should not be trusted as is, rather should be thought as any point in the ellipse.

Following chapter will include quantitative evaluation of filter performance. The code used for obtaining metrics and plots can be found in `kfeval.ipynb` notebook.

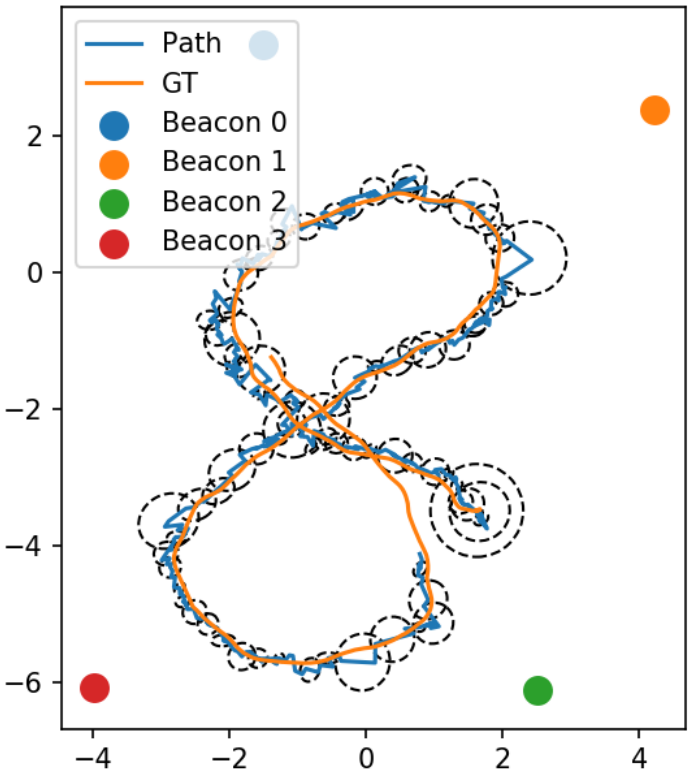

Figure 11: 2D plot of experiment trajectory with covariances.

To evaluate filter performance quantitatively several metrics can be used the most simple one could be MSE (Mean squared error), RMSE (Root MSE) or AME (Absolute mean error). But in order to compute these data points i.e. ground truth and estimated have to be aligned in time. This is not the case because Opticon system has one frequency of publishing ground truth and estimated are done in way lower frequency. Therefore each trajectory component (x, y, z) are linearly interpolated in time (using `scipy.interpolate` module) to be able to select same point in time for fair comparison. For instance, Figure 12 shows both of them sampled from interpolated trajectories in two dimensions. Table 1 summarizes performance of estimated trajectory quantitatively by metrics mentioned above.

Metric	x [m]	y [m]	z [m]
RMS	0.187	0.132	0.276
MSE	0.035	0.017	0.076
AME	0.158	0.105	0.212

Table 1: Error metrics.

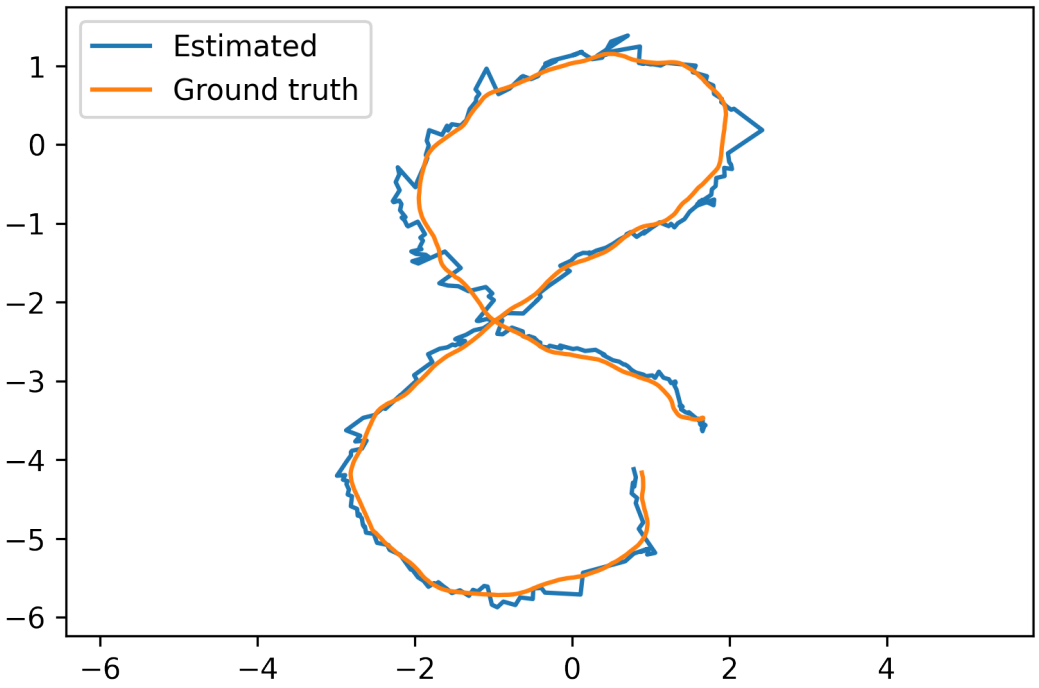

Figure 12: 2D plot of interpolated trajectories.

UWB chip DW1000 datasheet [5] states that it can measure distances up to 10cm accuracy which is inline with the estimation error. Absolute mean error at least in the (x, y) direction is around 10cm, the z component has a bit more and reasons for it were described before.

Also Kalman Filter assumes that noise in the measurements is unbiased and white [9]. We've seen before that measurements indeed had some bias involved (it was removed before fusing data into the filter). To make sure that measures taken to remove it were effective we can plot autocorrelation function on innovation $y - \hat{y}$ and see if 95% of the values fall into 2σ interval i.e. $\pm 2/\sqrt{N}$, N - number of innovations. Figure 13 shows the results and indeed whiteness is confirmed. Three curves are innovation auto-correlations for each of the beacons and the bound is shown in magenta color.

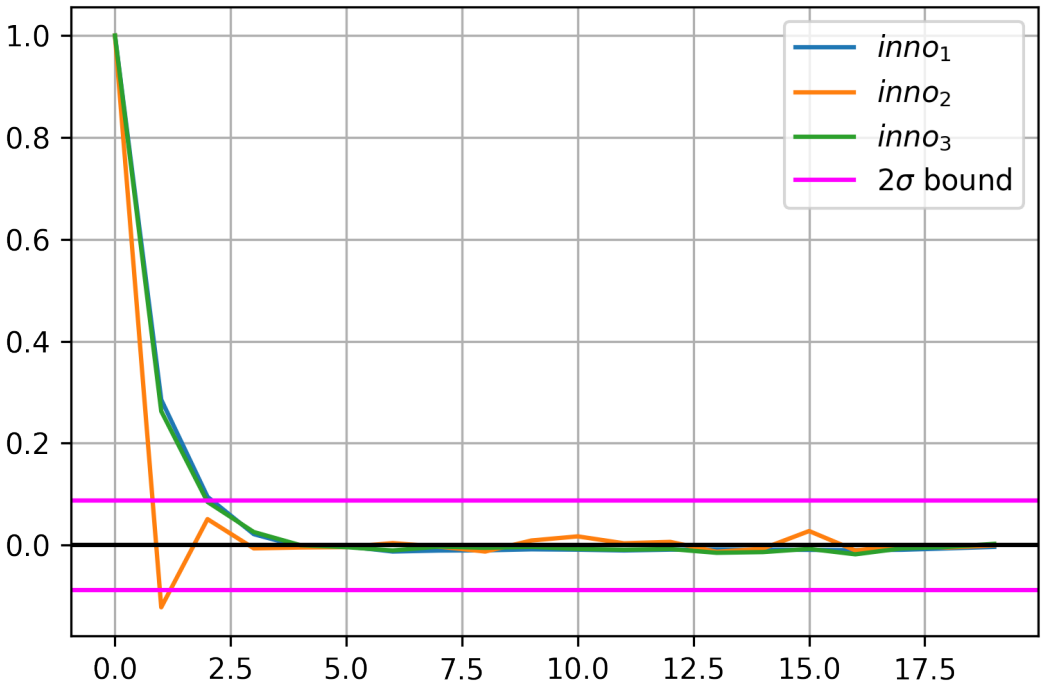

Figure 13: 2D plot of interpolated trajectories.

5 Conclusion & Future work

The project is a step towards a general localization system for UAV. The experimental setup now have only included doing estimation on an arbitrary walking trajectory of a human for which dynamics cannot be identified. In case of a real system i.e. flying drone this wouldn't be the case, the dynamics are mostly known and mathematical model can be derived from first principles and used along side estimation problem to make more reasonable prediction steps. Besides that, drone usually is equipped with more sensors like IMU, barometer or camera which can provide additional information to make localization more robust. Probably even adding single IMU and using constant acceleration model instead of constant velocity model would improve the positioning accuracy tremendously.

Considering positioning performance only for this data modality and constant velocity model the results obtained are satisfactory and any improvements on filter side would be marginal. For instance, tunning of the EKF (process noise and measurements) could be done on collected data applying optimization techniques but for project scope is not relevant.

Available UWB chips have a number of operating modes with varying properties. The scope of possibilities was not explored in detailed here and could further be investigated for possible adjustment. There could be modes with more desirable properties to the application in consideration. This would include studying manufacturer provided data-sheets.

Work can could be extended in the future to be used on multiple agents acting in an environment together. For example, drone could act as an anchor and tag at the same time with single or two UWB devices mounted on top and provide distance measurements to neighboring agents around. This would require specialized algorithm to incorporate data coming in from moving "anchors" and more care to covariances of estimates because this would introduce cross dependencies between measurements (now anchor positions are considered to be static/deterministic and fully known).

References

[1] S. Sadowski and P. Spachos, "Rssi-based indoor localization with the internet of things," *IEEE Access*, vol. 6, pp. 30149–30161, 2018.

[2] M. Ji, J. Kim, J. Jeon, and Y. Cho, "Analysis of positioning accuracy corresponding to the number of ble beacons in indoor positioning system," in *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pp. 92–95, 2015.

[3] "IEEE Standard for Low-Rate Wireless Networks–Amendment 1: Enhanced Ultra Wideband (UWB) Physical Layers (PHYs) and Associated Ranging Techniques," *IEEE Std 802.15.4z-2020 (Amendment to IEEE Std 802.15.4-2020)*, pp. 1–174, 2020.

[4] M. Ibrahim, H. Liu, M. Jawahar, V. Nguyen, M. Gruteser, R. Howard, B. Yu, and F. Bai, "Verification: Accuracy evaluation of wifi fine time measurements on an open platform," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 417–427, 2018.

[5] "Makerfabs ESP32 UWB module." `https://www.qorvo.com/products/p/DW1000`.

[6] D. F. Sebastian Thrun, Wolfram Burgard, *Probabalistic Robotics*. The MIT Press; 1st edition, 2005.

[7] G. Welch, G. Bishop, *et al.*, "An introduction to the Kalman filter," 1995.

[8] "Makerfabs ESP32 UWB module." `https://www.makerfabs.com/esp32-uwb-ultra-wideband.html`.

[9] I. R. Reid, "Estimation II," 2010.