# Deep Particle-filtered UAV Localization

**Seongjoon Kang and Tommy Azzino**

Department of Electrical and Computer Engineering
NYU Tandon School of Engineering
6 MetroTech Center, Brooklyn, NY 11201
[sk8053, ta1731]@nyu.edu

## 1 Introduction

Since its introduction, Unmanned Aerial Vehicle (UAV) technology has seen applications in fields such as military, aerial photography, 3D mapping, aerial inspection, search operations, etc. These applications are expected to increase with the upcoming 5th generation (5G) communication technology. However, moving UAV technology to 5G systems comes with its challenges, one of which is accurate localization. Accurate localization involves finding the best estimate of the UAV's 3D coordinates (x, y, and z) and it is key for applications such as package delivering with drones, and disaster monitoring, to name a few. 5G systems operate at millimeter-wave (mmWave) frequencies where the communication between a Transmitter (TX) and a Receiver (RX) becomes highly directional due to beamforming to overcome the extremely high propagation attenuation at mmWave frequencies with respect to legacy 4G-LTE frequencies which operate in the sub-6 GHz band. When there is a clear Line of Sight (LOS) path between TX and RX, the localization procedure is simply based on the spatial and temporal information of the signals exchanged (e.g., Global Positioning System - GPS). However, in urban environments where the majority of UAV applications are envisioned, the communication mainly occurs on Non-Line of Sight (NLOS) paths (rays) given by reflections from buildings, trees, obstacles, etc. Legacy methods, such as GPS, do not guarantee accurate localization in urban scenarios (e.g., Google Maps in Manhattan), thus compromising the user experience and safety in certain applications. Also, the localization problem becomes even more complex for UAVs in urban environments, given the varying height component involved (typical localization techniques are designed for 2D ground user position estimation).

In our project, we propose to solve the UAV localization problem in urban scenarios by combining Deep Learning (DL) and Latent Dynamical System (LDS) methods. The goal is to train a DL model to predict the location (x, y, z coordinates) of the UAV based on some selected wireless communication characteristics between the UAV and the Base Station(s) (BS)[1]. For each BS in the scenario, we train a different Deep Neural Network (DNN) to estimate the 3D location of the UAV (more details about the simulated scenario will be given in Section 4.1). However, the set of estimated positions of each BS might be very noisy. As such, in order to infer accurate UAV localization from the DNNs' output, we develop an algorithm based on particle filtering to filter out such noise and provide a better final estimate of the UAV location. We will show that our algorithm is able to improve the performance of UAV localization when compared to other methods. However, our solution could be improved by considering some aspects that can affect particle filtering. This will be part of our future work related to this project.

The reminder of this paper is organized as follows, Section 2 gives an overview of the current literature related to this problem. Section 3 provides a detailed definition of the problem addressed in this work, as well as a description of the main algorithm developed to tackle it. Then, Section 4 defines the scenario from which we collect relevant data for the problem and evaluates the performance of the proposed algorithm. Section 5 ends the paper with a summary of the obtained results and possible future work in this area.

[1] In our project, we will use the terms BS and TX interchangeably. As well as, UAV and RX.

## 2 Related Work

Xiao and Zeng [1] provided an excellent overview of communication and localization techniques anticipated for 6h generation (6G) wireless networks. In addition to a comprehensive analysis of state-of-the-art wireless localization methods, they also characterize the co-design of localization and communication for future 3D networks with aerial-ground integration. In addition, Kanhere and Rappaport [2] discussed several methods to achieve high localization accuracy of ground users in 5G and beyond networks. Yu and Guo [3] introduced a localization algorithm for ground users in heavy NLOS environments using a *weighted least squares* solution. Bhattacherjee et al. [4] proposed two different DL-based methods for wireless localization of ground Users (UEs) in two static 5G scenarios generated through ray-tracing simulation (i.e., wireless simulation of a given environment). However, in general, the literature lacks work in the context of 5G wireless UAV localization in challenging urban settings. Therefore, we intend to investigate a relatively unexplored research area. Moreover, to the best of our knowledge, this is the first work that addresses the problem of UAV localization in urban scenarios and considers a multi-disciplinary approach by merging DL and LDS theories.

## 3 Problem Definition and Algorithms

### 3.1 Task

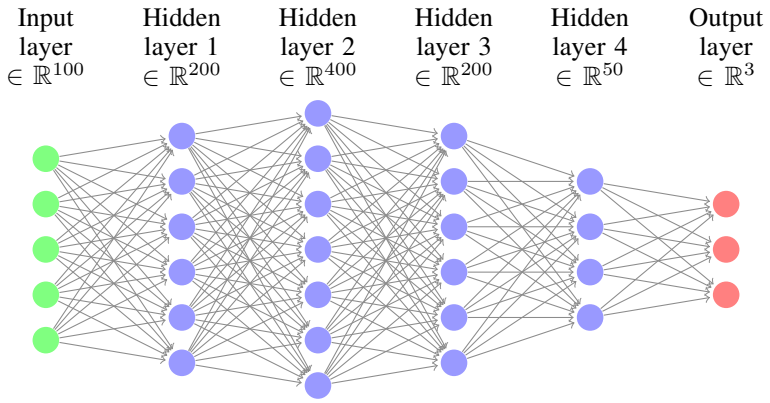



Figure 1: DNN architecture adopted to estimate the 3D location of the UAV. Note that each BS has been trained using this architecture but with different training data each. [The number of neurons drawn in the picture does not correspond to the actual number of neurons for each layer].

In this work, we address the problem of providing accurate localization of UAVs in challenging urban environments where the localization task is worsened by the dominant NLOS propagation of mmWave electromagnetic waves, causing legacy techniques such as GPS to fail or provide highly erroneous position estimates. In such scenarios, we foresee a UAV flying at low altitude (greater than or equal to 15 meters) across the streets of a city. The UAV is simultaneously connected to a certain number, $N_{BS}$, of BSs that provide 5G mmWave connectivity. We propose a two-step solution to accurately estimate the location of the UAV, explained in the following.

First, we devise a strategy based on DL to estimate the three-dimensional location of the UAV at each BS. In particular, we train a DNN for each BS with inputs consisting of a selected set of *wireless features* that the BS can estimate from the connected UAV. In this work, these *wireless features* are obtained through ray-tracing simulation, as explained in Section 4.1. We choose the inputs to our DNN models similar to what has already been done in the literature by Bhattacherjee et al. [4]. To define these inputs, we first define a path (ray) as the route a signal covers from the BS (TX) to the UAV (RX). This route can be a straight line (LOS) or can consist of reflections, refractions, diffractions, and transmissions (NLOS). Note that NLOS propagation causes the presence of multiple paths between a TX and RX. In addition, note that the UAV can be served simultaneously by multiple TXs (i.e., the Base Stations). We now define 6 input characteristics for our DNN architecture. Time of Arrival (ToA) is the time taken by a signal to reach a BS from the UAV, using some

path. Azimuth angle of Arrival (AoA) is the azimuth angle at which the signal of a given path arrives at the TX, while Zenith angle of Arrival (ZoA) is the zenith angle of arrival. Path power (P) is the power of the received signal at the BS for a given path from the UAV. The units of all angles are generally degrees or radians. ToA is measured in microseconds in urban environments, while P is measured in decibel milliwatts (dBm) or milliwatts. All the above characteristics refer to a single path, but generally, a 5G mmWave urban environment consists of multiple paths, $N_{paths}$. So, the overall input to each BS's DNN will consist of a tensor with dimensions $25 \times 4 = 100$ (where 4 is the number of characteristics, as introduced above, and 25 is the number of paths $N_{paths}$). The output of each DNN will be a three-valued tensor representing the location of the UAV specified as x, y, and z co-ordinates. Figure 1 depicts the DNN architecture adopted by each BS. The details behind this architecture and the training procedure employed are beyond the scope of this paper, however, in Section 4.1, we will provide information regarding the training data used to train each DNN. Moreover, we assume the presence of an entity in the network that collects the predicted 3D location from each BS (this is typically the case in current cellular networks to handle the mobility of users, as an example). Hence, at this network entity, we have a set consisting of $N_{BS}$ predictions for each true location at time step $t$ that we mathematically describe as follows:

$$\mathbf{X_t^{pred}} = \left[\mathbf{x}_{pred,t}^{b_1}, ..., \mathbf{x}_{pred,t}^{b_i}, ..., \mathbf{x}_{pred,t}^{b_{N_{BS}}}\right] \tag{1}$$

$\mathbf{x}_{pred,t}^{b_i} \in \mathbb{R}^3$ is the UAV location predicted by BS, $b_i$, with $i = 1, 2, ..., N_{BS}$, at time $t$. Some predictions in this set might be missing due to the lack of connectivity between the UAV and the corresponding BS.

One important thing to notice at this point is that it would be unfeasible to train a unique DNN which incorporates training data collected from multiple BSs. First, the collection of such amount of data by a unique network controller will cause a huge waste of communication resources, second, this alternative would require to train a model for a fixed number of BSs, $N_{BS}$, but network deployments in cities are frequently changing, especially with the advent of 5G technologies. Therefore, it is reasonable to assume that each BS locally estimates the UAV location and sends only the 3D estimate to the network controller, which will be responsible of providing a final accurate estimate.

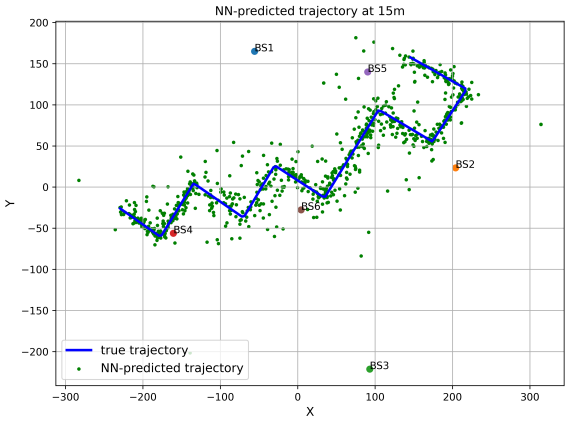



Figure 2: DNN predicted UAV locations for each BS at a given UAV trajectory. The UAV trajectory is taken from the scenario we will introduce in Section 4.1. During this flying path the UAV keeps a fixed height of 15 meters.

Figure 2 depicts the set of DNN-predicted UAV locations, $\{\mathbf{X_t^{pred}}\}_{\mathbf{t}=0:T}$, for a certain trajectory in the 3D surroundings of the city. Looking at the green dots representing predictions from each trained DNN, we realize that it is hard for the network controller to infer the exact UAV location from this set of noisy measurements. This is due to the UAV being in NLOS with respect to each BS for most of its flying route, making the task of predicting the UAV location hard, even for well-trained DNNs. Hence, we need to refine our approach by introducing more sophisticated techniques to accurately predict the final UAV locations.

The second step in our work consists in the adoption of particle filtering as an LDS approach to provide a more accurate estimated location of the UAV from the noisy estimates of the set $\mathbf{X^{pred}}$. The details of this algorithm are given in the next section, Section 3.2.

### 3.2 Algorithm

In this subsection, we introduce the details of the particle filtering algorithm that we developed. After training each BS's DNN, we can compute the distribution of the predicted outputs given the data. For each BS, $b_i$, we define the distribution of the predicted outputs from its DNN as follows:

$$\begin{aligned} P_{b_i}(\mathbf{x^{pred}}|\mathbf{y^{b_i}}, \mathbf{x^{true}}, \mathbf{G_{b_i}}) &= P_{b_i}(\mathbf{G_{b_i}}(\mathbf{y^{b_i}})|\mathbf{x^{true}}) \\ &= P_{b_i}(\mathbf{G_{b_i}}(\mathbf{y^{b_i}}) - \mathbf{x^{true}}|\mathbf{x^{true}}) \\ &= P_{b_i}(\epsilon|\mathbf{x^{true}}) \end{aligned} \tag{2}$$

Here, $\mathbf{x^{true}}$ is the true location of the UAV, which consists of the three Cartesian coordinates $x$, $y$, and $z$, whereas $\mathbf{x^{pred}}$ is the prediction given by the DNN. $G_{b_i}$ is the trained DNN for BS $b_i$, and $\mathbf{y^{b_i}}$ is the measurement input at BS $b_i$ consisting of the *wireless features* introduced in Section 3.1. $\mathbf{x^{true}}$ and $\mathbf{y^{b_i}}$ are given by the dataset. Also, $\epsilon$ is the difference, called *measurement noise*, between true and estimated location. Under assumption that the individual coordinates, $x$, $y$, and $z$ of a certain predicted output, $\mathbf{x^{pred}}$, are conditionally independent given $\mathbf{x^{true}}$, then the probability of the predicted output is given as follows:

$$P_{b_i}(\epsilon|\mathbf{x^{true}}) = P_{b_i}(\epsilon^x, \epsilon^y, \epsilon^z|\mathbf{x^{true}}) = P_{b_i}(\epsilon^x|\mathbf{x^{true}})P_{b_i}(\epsilon^y|\mathbf{x^{true}})P_{b_i}(\epsilon^z|\mathbf{x^{true}}) \tag{3}$$

In addition, the posterior distribution is approximately the same as the likelihood when the prior distributions are the same[2].

$$P(\mathbf{x^{pred}}|\mathbf{x^{true}}) = \frac{P(\mathbf{x^{true}}|\mathbf{x^{pred}})P(\mathbf{x^{pred}})}{P(\mathbf{x^{true}})} \approx P(\mathbf{x^{true}}|\mathbf{x^{pred}}) = P(\epsilon|\mathbf{x^{pred}}) \tag{4}$$

Thus, after sufficient training of each DNN, we calculate the probability of the predicted outputs by leveraging the error associated with each component, $x$, $y$, and $z$, between the predicted location, $\mathbf{x^{pred}}$, and the true one, $\mathbf{x^{true}}$. We approximate the likelihood distribution, $P(\mathbf{x^{true}}|\mathbf{x^{pred}})$, using the predicted output distribution, $P(\mathbf{x^{pred}}|\mathbf{x^{true}})$.
Since we collect data with 2-meter resolution (details in Section 4.1), we estimate the predicted output probabilities with 1-meter resolution to guarantee more accurate estimation. To do this, we rely on a Gaussian Process (GP) employing a simple Radial Basis Kernel (RBF) given as:

$$K_{2D}(\mathbf{x_1}, \mathbf{x_2}) = \sigma^2 exp\left(\frac{-||\mathbf{x_1} - \mathbf{x_2}||_2^2}{2l^2}\right) \tag{5}$$

Figure 3 shows a fitting example with a GP that applies the aforementioned RBF kernel function to get higher resolution probability maps by reducing the interval size of each grid from 2m to 1m. The fitting with GPs is applied to all predicted output maps for each BS and different UAV heights. For example, the probability of the predicted output map for BS 1 is plotted in Figure 4 for different heights of the UAV. The white parts of Figure 4 correspond to buildings in the simulated scenario. In our work, UAVs cannot fly inside buildings, therefore we cannot perform localization in such regions of the map. As such, for lower altitudes, we notice higher concentrations of obstacles, as expected. In the same way, we obtain the probability maps for all BSs at different positions in the simulated scenario.

With the trained DNNs and the precomputed probability distributions of the predicted outputs, we propose the LDS model illustrated in Figure 5. Hence, each BS $b_i$ will perform the particle filtering according to the following LDS model:

$$\begin{aligned} \mathbf{z_t^{b_i}} &= \mathbf{z_{t-1}^{b_i}} + \mathbf{w_t} \\ \mathbf{x_t^{b_i}} &= \mathbf{G_{b_i}}(\mathbf{z_t^{b_i}} + \mathbf{y_t^{b_i}}) + \epsilon_\mathbf{t}^{\mathbf{b_i}} \end{aligned}$$

[2]We assume that the prior distributions of $\mathbf{x^{true}}$ and $\mathbf{x^{pred}}$ are statistically the same after sufficient training of the DNNs.

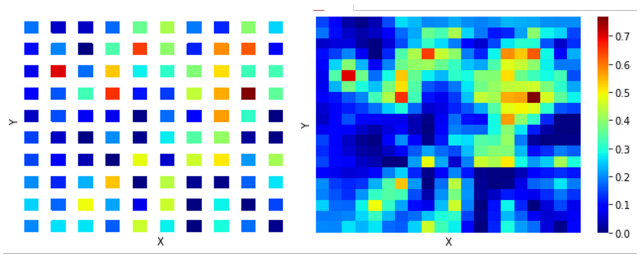



Figure 3: Distribution of the predicted outputs for different resolutions of a small part of the whole simulated area. The left picture depicts probabilities at 2-meter resolution, computed from the DNN's output, while the right one depicts the predicting probabilities at 1-meter resolution obtained using the GP.

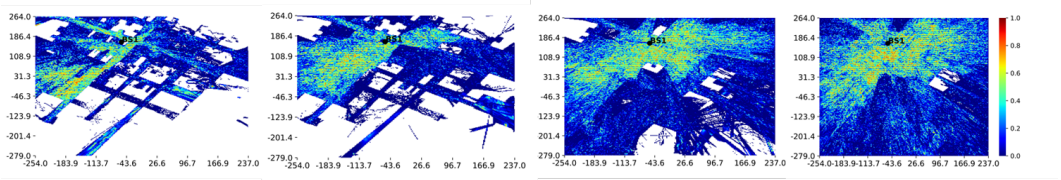



Figure 4: Distribution of predicted output over the entire simulated area for different heights of the UAV with respect to BS 1. When the height of the UAV is higher, the BS can predict the location of the UAV more accurately thanks to an increase in LOS propagation at higher altitudes.

Here, $\mathbf{z_t^{b_i}}$ is the latent variable, $\mathbf{w_t}$ is the noise of the latent state following the multivariate Gaussian distribution, $\mathbf{w_t} \sim N(0, \mathbf{Q})$, $\epsilon_\mathbf{t}^{\mathbf{b_i}}$ is measurement noise, following the distribution of the predicted outputs, and $y_t^{b_i}$ represents the wireless features that are the input of the DNN at BS $b_i$. Intuitively, the proposed model creates many perturbed outputs (we call them particles), by adding latent variables to inputs, to which different weights are assigned based on the importance computed by likelihood distribution of Equation 4.

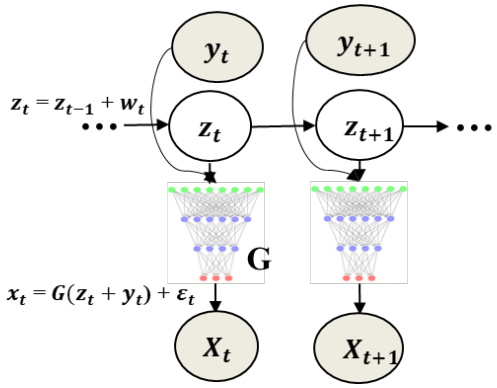



Figure 5: LDS model for particle filtering of the outputs from each DNN.

Algorithm 1 shows the main steps involved to perform particle filtering at each BS. Particle filtering recursively computes the importance weights approximated by normalizing the likelihood distributions, $P(\epsilon_\mathbf{t}^{\mathbf{b_i}} | G_{b_i}(\mathbf{y_t^{b_i}} + \mathbf{z_t^{b_i}}))$ that are precomputed and known. Accordingly, the particles are re-sampled and propagated to next state (lines 5-7). Then, the UAV location for each BS is estimated by averaging the outputs of a DNN from the re-sampled particles (line 8). Ultimately, among the estimations from all the BSs, the final predicted UAV location (by the network controller) is the one that has the maximum predicting probability (line 10).

**Algorithm 1:** Particle filtering algorithm

**input :** $\mathbf{BS}$: BS set, $\{G_{b_i}\}_{b_i \in \mathbf{BS}}$: Trained DNNs
**output:** $\{\mathbf{x_t^\star}\}_{t=0:T-1}$ : Optimal trajectory estimation

Initialize $\mathbf{z_0^{b_i}}$, $\mathbf{y_0^{b_i}}$, and $\mathbf{x_0^{b_i}}$ for every BS, $b_i$
Initialize $K$ particles, $\{\mathbf{z_0^{j,b_i}}\}_{j=1:K}$, for each BS $b_i$
**for** $t = 0$ **to** $T-1$ **do**
  **for** $b_i \in \mathbf{BS}$ **do**
    From previous $K$ particles, compute importance weights for each particle, $j$; $\mathbf{w_t^{j,b_i}} = \frac{P(\epsilon_{\mathbf{t}}^{\mathbf{b_i}} | G_{b_i}(\mathbf{y_t^{b_i}} + \mathbf{z_t^{j,b_i}}))}{\sum_m P(\epsilon_{\mathbf{t}}^{\mathbf{b_i}} | G_{b_i}(\mathbf{y_t^{b_i}} + \mathbf{z_t^{m,b_i}}))}$
    Re-sample all particles,$\{\mathbf{z_t^{j,b_i}}\}_{j=1:K}$ based on the weights, $\{\mathbf{w_t^{j,b_i}}\}_{j=1:K}$
    Propagate the particles to next time step, $t+1$, via $P(\mathbf{z_{t+1}^{b_i}} | \mathbf{z_t^{b_i}})$ and observe $\mathbf{y_{t+1}}$
    $\mathbf{x_{est,t+1}^{b_i}} = \frac{1}{K} \sum_j G_{b_i}(\mathbf{z_{t+1}^{j,b_i}} + \mathbf{y_{t+1}^{b_i}})$
  **end**
  $\mathbf{x_{t+1}^\star} = \underset{b_n \in \mathbf{BS},\, \mathbf{x_{est,t+1}^{b_n}}}{\arg\max} \; P_{b_n}(\epsilon_{\mathbf{t+1}}^{\mathbf{b_n}} | \mathbf{x_{est,t+1}^{b_n}})$
**end**

# 4 Experimental Evaluation

## 4.1 Data

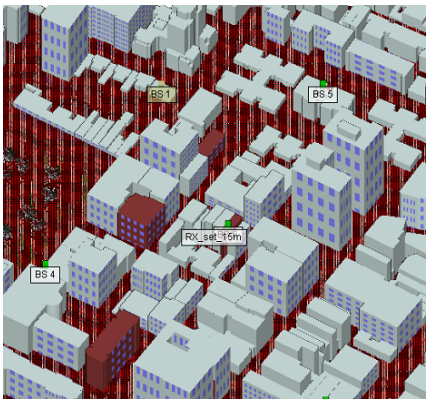
Since the research area of 5G mmWave UAV localization is relatively new, there are no publicly available datasets for our problem. Hence, we need to generate our own dataset. For an accurate generation of the aforementioned *wireless features*, we used a commercial ray-tracer called Wireless InSite by Remcom [5]. This software tool is widely adopted in the research community for generating realistic wireless data given the topology and layout of a certain environment and the position of TXs and RXs [6]. Wireless InSite generates the required data based on optics and electromagnetic theory. The ray-tracer requires to specify the total number of paths, $N_{paths}$, to consider (i.e. the number of paths of the signal from each BS to the UAV, as discussed in Section 3.1). In our case, we set this parameter to 25. The RX points corresponding to the UAV locations are spaced 2 meters apart and form a grid that covers the entire simulated area (simulating in Remcom a grid with higher resolution would require exponentially increasing simulation time and more hardware resources). Moreover, we generated these grids of RX points for fixed UAV heights at 15, 30, 60, and 120 meters. This data will constitute the training data for the DNNs. Additionally, the maps of the distribution of the predicted output for particle filtering are computed based on this data. The scenario adopted for the creation of our dataset represents an area in New York City close to Washington Square Park (the 3D layout of the city in this area was imported into Remcom). Figure 6(a) depicts the 3D layout of this area as well as the positions of some of the BSs in the scenario. On the other hand, Figure 6(b) depicts a set of trajectories that we used as test data to benchmark the performance of our algorithm against other solutions.

## 4.2 Methodology

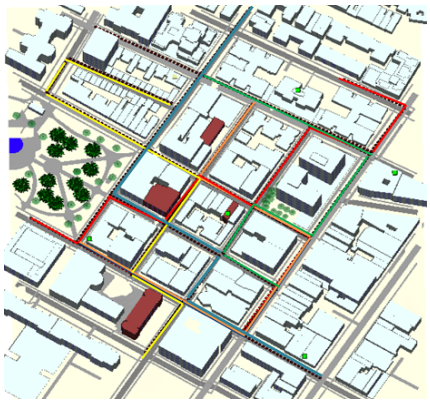
We create five random trajectories along the streets of the simulated environment (as depicted in Figure 6(b)) to obtain the test data ($x_t^{true}$, $y_t^{b_i}$). Based on the test data, we run the proposed algorithm and apply a Moving Average (MA) process to make the estimated UAV locations (UAV trajectories) smoother. Moreover, we compared the performance of our deep particle-filtered scheme with a Maximum A Posterior (MAP) estimator, defined as follows:

$$\mathbf{x_t^\star} = \underset{b_n \in \mathbf{BS},\, \mathbf{x_{est,t}^{b_n}}}{\arg\max} \; P_{b_n}(\epsilon_{\mathbf{t}}^{\mathbf{b_n}} | \mathbf{x_{est,t}^{b_n}}), \;\; \mathbf{x_{est,t}^{b_n}} = G_{b_n}(y_t^{b_n}) \tag{6}$$

Equation 6 uses the likelihood distribution according to Equation 4. We also adopt the same MA process to smooth the outputs of the MAP method. For comparison, we also take into account the average estimator, which just predicts the final UAV location as the average of the estimations from




(a) 3D layout of the simulated scenario in New York City. The red points represent the RX grid of UAV locations spaced by 2 meters.



(b) UAV trajectories for testing the proposed algorithm. Different colors represent different trajectories.

Figure 6: Simulated 5G mmWave urban scenario using Remcom's Wireless InSite [5].

all BSs (the outputs of the DNNs). Finally, we adopt the Mean Squared Error (MSE) between the estimated and true UAV locations (UAV trajectories) as performance indicator for bench-marking the algorithms introduced above. Note that in the following, we will present the results in terms of 2D localization since we are assuming a UAV flying at a constant altitude for each test trajectory.

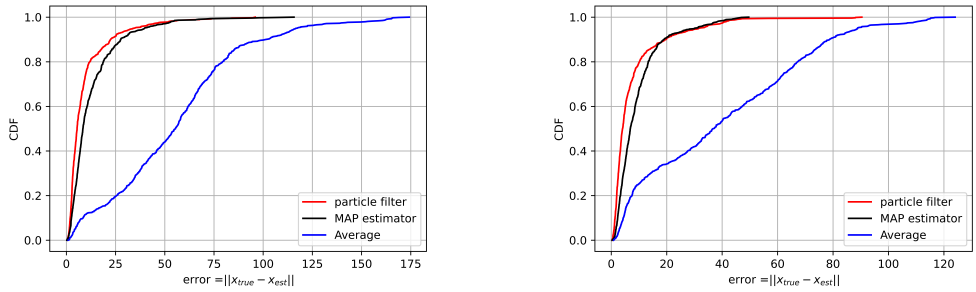
### 4.3 Results

In this subsection, we present the results obtained by running the algorithms discussed above using the UAV trajectories generated as test data. Figure 7 shows the CDF of the MSE of the final predicted UAV location computed over all test trajectories. The proposed particle filtering algorithm shows much better performance than the other two methods when estimating the UAV location at both 15m and 30m heights. For example, approximately $90\%$ of the time, the error in UAV localization using particle filtering is below 20 meters in both cases. Also, we can notice that in general the performance at 15 meters is worse given by the mostly NLOS signal propagation between the UAV and each BS. Whereas, at higher altitude the percentage of NLOS links will be lower. We observe some performance degradation also with the particle filtering approach. The main reason is due to the instability of DNNs, which relates to a larger variation of their output caused by a small variation of the input. Our DNNs may not be stable for some inputs, thus adding a latent variable to the input may generate outlier points for a few cases. This intuition is corroborated by observing that the simple average estimator provides much worse performance when compared to the other two schemes. Moreover, only a few particles in particle filtering can have significant weights, resulting in *sample impoverishment*, while most particles are disregarded during the resampling process. This is another reason of the performance degradation of our model.

Figure 8 shows the same trajectory at different heights estimated by the particle filtering algorithm and the MAP estimator, which are both combined with a moving average filter to make the estimation smoother. We observe that the particle filter successfully tracks the true trajectory for both 15m and 30m cases and, in general, provides better performance with respect to the MAP estimator.

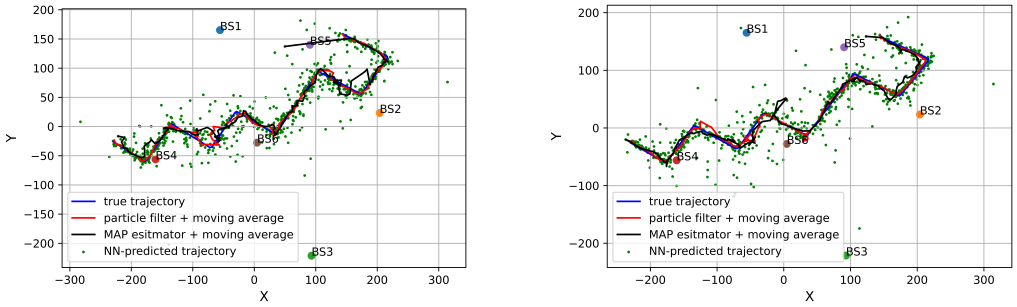
### 4.4 Discussion

The proposed algorithm shows better performance than the MAP estimator, which is one of the well-known optimal estimators. However, we observe some outlier predictions from our model because we do not consider some issues of particle filtering such as *sample impoverishment* after re-sampling of particles. To address this issue, we need to adopt new additional methods such as [7] and [8]. In addition, the instability of forward propagation in DNNs is another issue of the proposed scheme. We do not know how large the latent variable can be. Too small values may not force particles to a high likelihood area, but too large values cause large deviations from true points. Thus, we should




(a) UAV height = 15m (b) UAV height = 30m

Figure 7: CDFs of the MSE between the predicted and true trajectory.




(a) UAV height = 15m (b) UAV height = 30m

Figure 8: One example of UAV trajectory belonging to the test set.

choose the proper range of values for the latent variables to guarantee stability of the proposed algorithm.

## 5 Conclusions and Future work

In this work, we proposed a particle filtering algorithm to provide accurate UAV localization in 5G mmWave urban environments. In the proposed model, we utilize the likelihood distribution approximated with posterior probabilities obtained from the training data and the trained DNNs. Based on the likelihood distribution, we introduced an LDS model based on particle filtering to improve the localization performance. We showed that the proposed model achieves better performance than the MAP estimator and the average estimator. This implies that particle filtering helps estimating the true measure when some data point is corrupted by noise.

As future work, the proposed particle filtering strategy can be modified to prevent *sample impoverishment* and to reduce time complexity by changing sampling size flexibly. Also, we will try different types of DNN and choose the most stable one. For example, we can reduce the input dimension of the DNN by choosing a smaller number of *wireless features* or introduce some methods to prevent the *over-fitting* problem. Finally, we will test much higher UAV altitudes to evaluate our model in various wireless conditions.

## References

[1] Zhiqiang Xiao and Yong Zeng. An Overview on Integrated Localization and Communication Towards 6G. 2020.

[2] Ojas Kanhere and Theodore S. Rappaport. Position location for futuristic cellular communications: 5g and beyond. *IEEE Communications Magazine*, 2021.

[3] Kegen Yu and Y. Jay Guo. Improved positioning algorithms for nonline-of-sight environments. *IEEE Transactions on Vehicular Technology*, 2008.

[4] Udita Bhattacherjee, Chethan Kumar Anjinappa, Loy Curtis Smith, Ender Ozturk, and Ismail Guvenc. Localization with Deep Neural Networks using mmWave Ray Tracing Simulations. *IEEE Southeastcon*, 2020.

[5] Remcom. Wireless InSite. http://www.remcom.com/wireless-insite.

[6] Xuexia Yang and Yimin Lu. Propagation characteristics of millimeter wave in circular tunnels. 2006.

[7] Tiancheng Li, Shudong Sun, Tariq Pervez Sattar, and Juan Manuel Corchado. Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches. *Expert Systems with applications*, 2014.

[8] Seongkeun Park, Jae Pil Hwang, Euntai Kim, and Hyung-Jin Kang. A new evolutionary particle filter for the prevention of sample impoverishment. *IEEE Transactions on Evolutionary Computation*, 2009.

## Student Contribution

Tommy Azzino generated the dataset using Remcom. Seongjoon Kang implemented the particle filtering algorithm. Both of them trained the DNNs. Also, both of them collaborated in the writing of the report.