Spatial Prediction of UAV Position Using Deep Learning

Márton Bertalan Limpek*†, Géza Szabó*, László Hévizi*, István Gódor*
*Ericsson Hungary Ltd., Budapest, Hungary,
email: {marton.limpek,geza.szabo, laszlo.hevizi, istvan.godor}@ericsson.com
†HSN Lab, Department of Telecommunications and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Hungary

***Abstract*—This paper presents a deep learning approach for drone navigation aimed at accurately estimating a drone's position based on Received Signal Strength (RSS) in a simulated environment. A predefined route is established, and the drone collects RSS and Inertial Measurement Unit (IMU) data while traversing it multiple times. This dataset serves as the foundation for training various neural network architectures. We create multiple training and test datasets to compare different models and identify the most effective approach for predicting the drone's location. After training, the neural networks are evaluated using distinct test data, showing that deep learning models can reliably predict a drone's position in simulation. The developed predictive models enhance the accuracy and reliability of autonomous drone operations in real-world scenarios.**

***Index Terms*—Drone Navigation, Received Signal Strength (RSS), Model Comparison, Position Estimation**

I. Introduction

In recent years, there has been a significant rise in the utilization of Unmanned Aerial Vehicles (UAVs), commonly referred to as drones. Drones are increasingly being used across various industries, including delivery services, search and rescue operations, real estate marketing, agriculture, and infrastructure inspection. Various real-world uses of UAVs encompass tasks such as search and rescue missions, delivering goods, agricultural endeavors, and conducting inspections on structures.In addition to being utilized by many for recreational purposes such as capturing photos and videos due to their affordability.

In various contexts, having precise information about the whereabouts of a drone is of utmost significance. This holds true whether the drone is being piloted remotely or operating autonomously. In this paper, we delve into exploring methods to accurately determine the location of a drone. This involves leveraging Received Signal Strength Indicatior (RSSI) data, a metric that measures the strength of signals received by the drone, data coming from the IMU, and employing sophisticated machine learning techniques to process and interpret this data effectively. The proposed method in action can be seen here: [1]

Despite the increasing popularity of drones, there are several challenges and limitations in accurately determining their location. These challenges come from different factors, like environmental conditions, sensor limitations, and the complexity of the operational context.

Most drones come equipped with a GPS module to facilitate precise location data access, in addition to various other sensors. PX4 offers compatibility with various Global Navigation Satellite System (GNSS) receivers and magnetometers (compasses). Additionally, it supports Real Time Kinematic (RTK) Global Positioning System (GPS) Receivers, which enhance GPS systems to provide accuracy down to the centimeter level. [2] The study [3] demonstrates the reliable performance of GNSS RTK for UAV localization using the Third Generation Partnership Project (3GPP) Long-Term Evolution (LTE) Positioning Protocol (LTE Positioning Protocol (LPP)) over a cellular network, with no loss of RTK GNSS fixed solution observed during both static and flight tests.

However, there may be situations where, for instance, GPS functionality is compromised due to factors such as signal interference or obstruction by tall buildings or dense foliage. In such cases, it becomes crucial to rely on alternative methods for locating the UAV. For example, in urban environments with tall buildings that block GPS signals this situation could easily happen. It is also worth noting that cellular positioning can also complement GNSS by providing an independent verification method. A method for redundancy in localization methods would ensure the continued operation of the UAV even in challenging conditions where GPS signals are unreliable or unavailable.

Using RSSI values to predict a drones location is an established research problem. Authors of [4] focus on outdoor 3-Dimensional (3D) drone geolocation using Radio Frequency (RF) signals and machine learning-based techniques. With their approach, which is - 'Hybrid RSSI-based 3D geolocation blending trilateration and ML-driven regression' - an average 3D error of approximately 11.7 meters could be achieved. Our paper extends their work by elevating the complexity of the applied ML models exploiting the sequential nature of the data.

To meet the above mentioned requirements, this paper proposes the development of an alternative method to localize the drone and overcome on the mentioned challenges. In this

979-8-3315-0973-6/24/$31.00 ©2024 IEEE

approach, we begin by constructing a virtual environment to conduct all experimental procedures. This environment consists of multiple components, which will be elaborated upon in the forthcoming section (Section III). Notably, the simulated environment is unique in its ability to leverage ray tracing methodology for calculating path losses/RSSI values. Within this environment, various datasets are gathered, which served as inputs for training distinct Neural Network (NN) models. These models are trained and evaluated their performance, comparing their outcomes.

Overall, this paper aims to develop a method for enhancing drone localization accuracy by leveraging RSSI data and advanced machine learning techniques. Through the construction of a simulated environment and the utilization of neural network models, the objective is to create a robust localization system capable of accurately predicting the drone's position in various operational scenarios. By addressing the challenges associated with traditional localization methods and harnessing the power of machine learning, this paper endeavors to contribute to the advancement of drone navigation technology and facilitate the safe and efficient deployment of drones in real-world applications.

II. Related work

According to [5], positioning, Detect-and-Avoid, and other safety-related sensors are critical for Beyond Visual Line of Sight (BVLOS) operations, as well as for UAVs autonomy. While GPS is often used for location, it may not always be sufficient or necessary for safe operations in certain scenarios. The robustness of any sensor system should be evaluated during the operational approval process, considering the risks involved. In GPS-denied environments like urban canyons, alternative methods such as vision-based systems or dead reckoning should be considered viable options for positioning.

Authors of another study [6] propose a novel approach by integrating Huber loss with altitude constraints to enhance localization using RSSI data. The proposed method median error is 5.4 meters in static conditions. However, this error increases to 62.98 meters in dynamic situations. In our work the dynamic environment (moving drone) is more emphasised and since this method has a relatively large error, we are experimenting with other techniques (see below). Authors of [7] introduce a method, were traditional fingerprint-based method and deep-learning is combined to predict the location of a drone in an indoor environment. Their method is similar to the work in this paper, however in that case the used deep-learning model is a classifier but in this case it predicts continuous values (regression).

Another method using Recurrent Neural Networks (RNNs) for three-dimensional node localization in UAV-assisted wireless networks is proposed in [8]. In this paper the nodes are localised, but the models - RNN - used for localisation is relevant to my case. Their approach achieved significant results, including enhanced localization precision. As for the future direction, Long Short-Term Memory (LSTM) model is also mentioned, which is utilized in this method.

III. Implementation details

A. The Simulator

Gazebo is an open-source 3D physics simulator that is widely used for simulating robots and other complex systems. It provides an environment for testing and developing robotics algorithms and hardware in a safe, controlled virtual environment. Gazebo can simulate sensors, actuators, and the physical interactions between objects, making it a valuable tool for simulating complex robot behavior and testing control algorithms.

Gazebo is highly customizable and can be extended with plugins and other features to support a wide range of use cases. It is commonly used in research, education, and industry for developing and testing robots and other complex systems.

The development of Gazebo is sponsored by the Open Source Robotics Foundation, which also sponsors the development of the Robot Operating System (ROS). As a result, Gazebo integrates well with ROS and is often used in conjunction with ROS to provide a complete simulation environment for robotics research and development.

Some of the key features of Gazebo [9] include its support for multiple physics engines, its ability to simulate sensors and actuators, its support for multiple robot models, and its integration with ROS.

B. GPU Accelerated Radio Frequency Path Tracer

Specialized processors, such as Graphics Processing Units (GPUs), are vital for task acceleration. Originally designed for graphics, GPUs are now utilized for general-purpose computing, including Artificial Intelligence (AI) and video processing, thanks to their powerful shader pipelines that significantly outperform traditional Central Processing Units (CPUs). Nvidia's 2019 RTX cards introduced real-time hardware ray tracing to consumers.

Our tool [10] harnesses Nvidia's RTX for efficient ray tracing, enabling precise calculations of RF transmission. It models radio propagation in complex indoor environments, considering detailed surface materials like metal coatings and accounting for user mobility and object dynamics that may disrupt connectivity between emitters and detectors. More detailed working mechanism of the tool is discussed in [11].

C. Control of the drone

To control the drone in the simulated environment we integrated PX4-Autopilot, which is an open-source flight control software with modular architecture. [12]

The integration of ROS with PX4 and the Gazebo Classic simulator works through the MAVROS MAVLink node, which enables communication between ROS and PX4. In this setup, PX4 receives sensor data from Gazebo's simulated

environment and sends motor/actuator commands. PX4 also communicates with the Ground Control Station (GCS) and an Offboard Application Programming Interface (API) (like ROS) to send telemetry data and receive commands for controlling the simulated drone. [13]

We created the mission plans - to gather training and test data - in QGroundControl. [14]

IV. Data Collection and Model Teaching

For the data collection we had to implement missions for the drone, where we define the trajectory where should the drone fly and we set also the speed of the drone. The data that we are collecting - both the training data and test - is simulated. The missions are defined in the QGroundControl Graphical User Interface (GUI). Three missions are defined for gathering training data, which missions look the same from the top view in the QGroundControl GUI Figure 1. Also, three missions are defined as for gathering test data for three cases. The three cases for the training and test dataset are the following:

1) Constant altitude, constant speed
2) Varying altitude, constant speed
3) Varying altitude, varying speed

The speed of the drone is $v_{const} = 5\ \frac{m}{s}$ in the case where the altitude is not changing and in the varying case it is between $v_{varying} = 1\ \frac{m}{s} - 10\ \frac{m}{s}$ on the straight sections.

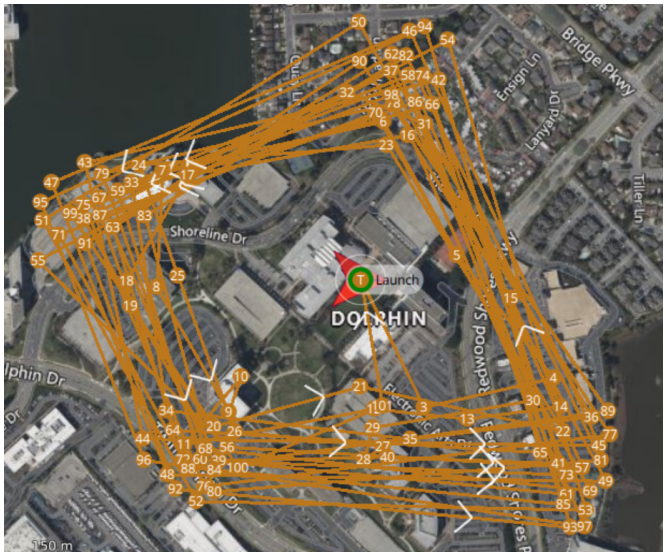


Fig. 1: Data collection mission plan for teaching, generated and displayed in QGroundControl

The perimeter of the area where we are collecting data is around 2,463 m, the size of the area is around 367,156 m^2.

Six Base Stations (BSs) are placed in the area, the drone in the simulation is (virtually) connected to all of these BSs. The placement of the BSs can be seen on 3

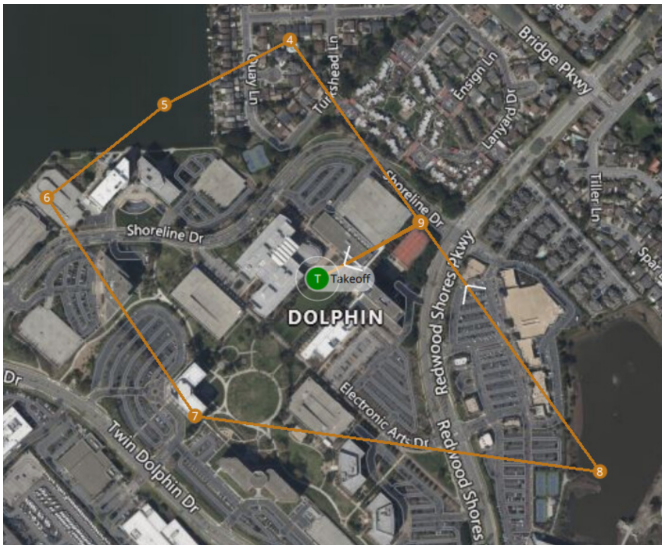


Fig. 2: Test mission plan in QGroundControl

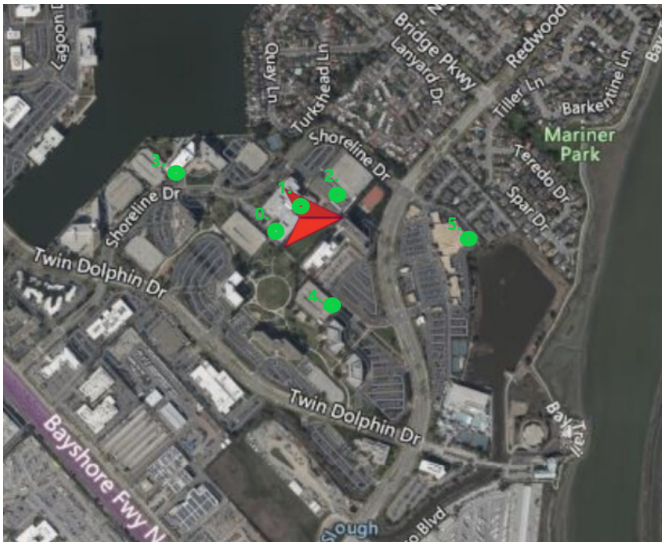


Fig. 3: Location of the six BSs

The data collection script is implemented in python as a ROS node, which collects the data periodically while the drone is doing the mission. The sampling period in every case is constant $T_s = 0.1$ second.

One data record contain the following information:

1) Path loss (from each BS)
2) Angular velocity (x, y, z direction)
3) Linear acceleration (x, y, z direction)
4) Latitude, Longitude, Altitude
5) Local coordinates (x, y, z)

Theoretically, the correlation between path loss and RSS can be considered to be -1 under ideal conditions. The ray tracing software returns the path loss values of the rays, and in this work we used those values for teaching the models.

A. Data Analysis

After collecting the data, an Exploratory Data Analysis (EDA) is conducted. During the EDA, several steps are carried out, starting with analyzing the correlation between the path losses to identify relationships between the features. This is followed by the visualization of the collected data to provide a better understanding of its structure and distribution. Additionally, the takeoff and landing segments of the test data were removed to avoid any potential biases they might introduce. Finally, the data underwent transformation, including scaling and Principal Component Analysis (PCA) in some cases to ensure consistency and improve model performance.

Since we also examined how the prediction changes based on the number of the BSs, we needed to identify which BS not to consider in different scenarios. We made this choice based on the correlation of the path losses and the Variance Inflation Factors (VIFs) values of the different path losses.

$$VIF_i = \frac{1}{1 - R_i^2} \tag{1}$$

, where R_i^2 is the coefficient of determination obtained by regressing the i-th variable against all other variables. We dropped the feature - correspondig to a BS - with the higher VIF value first. After the PCA the first 8 principal components were chosen. Since take off and landing is not in our main interest, we removed those parts from the test data sets.

B. Models

The primary objective is to accurately predict the moving drone's location within the designated area given different conditions. Essential to this task is determining the number and placement of BSs required for precise predictions. Six base stations, strategically positioned atop buildings, replicated real-world conditions within the area.

We used Deep Neural Network (DNN), RNN and LSTM models, implemented in TensorFlow/Keras. DNN model consists of three hidden layers with ReLU activations and dropout regularization, designed for predicting positions using three output units. It is compiled with the Adam optimizer and trained with a mean squared error loss function for regression tasks.

RNN model has three SimpleRNN layers, with the first two returning sequences and the final one providing the output sequence for prediction. It is designed for regression tasks with three output units, compiled using the Adam optimizer, and trained with a mean squared error loss function on reshaped input data.

This model is a LSTM network with three LSTM layers, where the first two return sequences and the last LSTM layer outputs the final hidden state. It is designed for regression tasks with three output units, compiled using the Adam optimizer, and trained on reshaped input data with mean squared error loss.

The mentioned datasets are used to train the models IV, with supervised learning. All of the models are tuned using Keras Tuner, a hyperparameter optimization framework that simplifies the search for optimal values through a define-by-run syntax. Keras Tuner supports various search algorithms, allowing for flexible and effective exploration of hyperparameter spaces. [15]

V. Evaluation

In the evaluation phase we used our models IV-B to make predictions for the data in the test sets. The output is a 3D position. The Euclidean distance between two points in 3D space was calculated to measure the error between each predicted and actual points. After that, for each test case the average of the error distances are calculated. Since the calculation of the average is sensitive to outliers, which are extreme values that differ significantly from the rest of the data, the average was calculated for the all of the error distances and for the subset of the data - removed outliers at the 0.95 percentile - as well.

During the evaluation it turned out, that PCA during the data pre-processing improves the performance of the models in case of 6 BSs.

We summarized the results in Table I to make the model comparison easier. We had 36 simulation cases, not counting the cases where we experimented with different data preprocessing techniques like PCA. In the table the average of the error distances (ε) are inserted in meter [m] unit. We also examined the standard deviation (σ) of errors, which indicates the variability of prediction errors, allowing to assess the reliability and consistency of the model's performance.

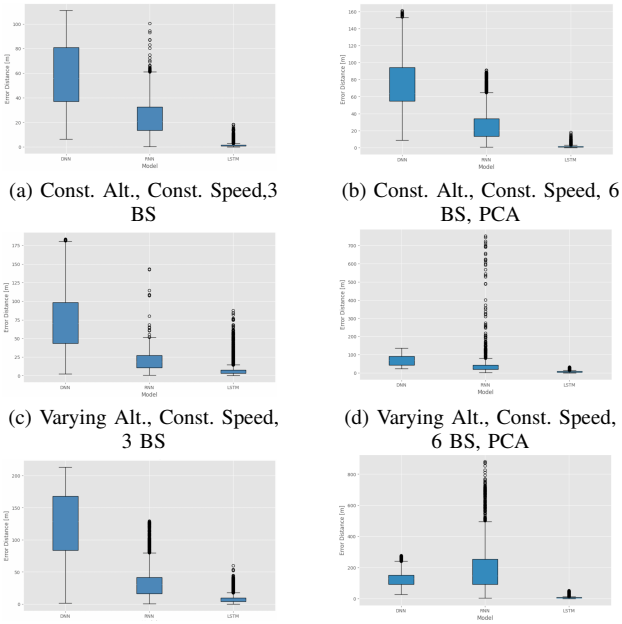
TABLE I: Model Performance

Test Case	**Model**					
6 BS	***DNN*** (ε, σ)		***RNN*** (ε, σ)		***LSTM*** (ε, σ)	
C. A.[1], C. S.[2]	102.5	46.4	19.1	12.3	1.8	3.7
V. A., C. S.[2]	104.3	41.5	19.6	9.3	5.9	3.6
V. A., V. S.	126.9	51.1	25.4	12.3	7.8	5.5
5 BS	***DNN*** (ε, σ)		***RNN*** (ε, σ)		***LSTM*** (ε, σ)	
C. A.[1], C. S.[2]	113.4	34.27	18.7	11.4	1.4	2.6
V. A., C. S[2]	106.6	50.4	22.9	14.6	5.9	5.7
V. A., V. S.	139.8	29.8	24.6	9.6	6.91	6.2
4 BS	***DNN*** (ε, σ)		***RNN*** (ε, σ)		***LSTM*** (ε, σ)	
C. A.[1], C. S.[2]	111.38	55.6	20.1	12.8	2.2	4.3
V. A., C. S.[2]	110.1	45.1	26	16.8	6.5	6.7
V. A., V. S.	154.1	44.5	27.1	19.9	8.5	5.7
3 BS	***DNN*** (ε, σ)		***RNN*** (ε, σ)		***LSTM*** (ε, σ)	
C. A.[1], C. S.[2]	57.7	24.7	23.7	14	1.6	1.8
V. A., C. S.[2]	76.7	42.1	19.7	11.4	8.1	10.3
V. A., V. S.	123.2	55.6	33.1	23.4	7.9	7
6 BS, PCA	***DNN*** (ε, σ)		***RNN*** (ε, σ)		***LSTM*** (ε, σ)	
C. A.[1], C. S.[2]	84.3	29.4	26.1	18.8	1	1.2
V. A., C. S.[2]	83.6	27.3	37.1	49.7	5.3	4.5
V. A., V. S.	110.5	51.1	177.4	119.8	6.5	6.2

[1]There is no intentional change – constant (C) – in altitude (A) in the mission.
[2]The drone's speed (S) is varying (V) in the bends.

In the data collection phase we defined three cases as mentioned in IV. In the table we include the initials for the different test cases.

In Table I in the last section the average of the error distances are inserted in case of 6 BSs, doing PCA on the data.



(a) Const. Alt., Const. Speed,3 BS

(b) Const. Alt., Const. Speed, 6 BS, PCA

(c) Varying Alt., Const. Speed, 3 BS

(d) Varying Alt., Const. Speed, 6 BS, PCA

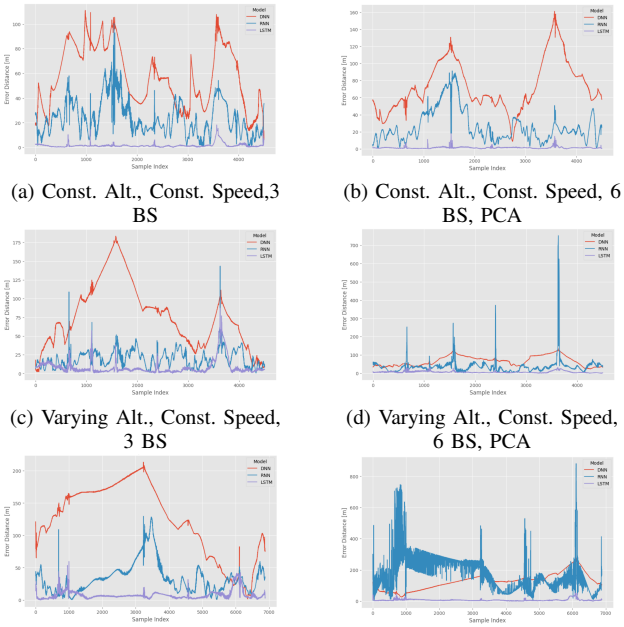
(e) Varying Alt., Varying Speed,3 BS

(f) Varying Alt., Varying Speed,6 BS, PCA

Fig. 4: Spread of error distances

In our study, all results were generated and compared. However, since they were similar, we focus on comparing the results in two specific cases: first, with 3 BSs, the lowest number of BSs, to demonstrate that even with minimal information, our model was able to make accurate predictions; and second, with 6 BSs and PCA, as this configuration resulted in the best performance.

Based on the results in Table I and Fig. 4, in our case the performance of the models are not dependent on the number of BSs. Fig. 4 clearly illustrates that the RNN model consistently produces many extremes and outliers across all cases. These extreme values indicate a larger variability in the model's error distances. When we apply PCA to the data, the performance of the RNN model worsens further, leading to a broader spread of errors and even more pronounced outliers. On the other hand, applying PCA results in a significant improvement for the LSTM model. This enhancement is reflected in a tighter distribution of error distances and fewer occurrences of extreme outliers, making the LSTM model more reliable. The box plots on Fig. 4 highlights that the DNN model exhibits the worst performance among the models, as indicated by the generally larger spread of error distances, with the exception of cases (d) and (f). In contrast, the LSTM model consistently demonstrates the best performance, showing the smallest spread of error distances in every case.



(a) Const. Alt., Const. Speed,3 BS

(b) Const. Alt., Const. Speed, 6 BS, PCA

(c) Varying Alt., Const. Speed, 3 BS

(d) Varying Alt., Const. Speed, 6 BS, PCA

(e) Varying Alt., Varying Speed,3 BS

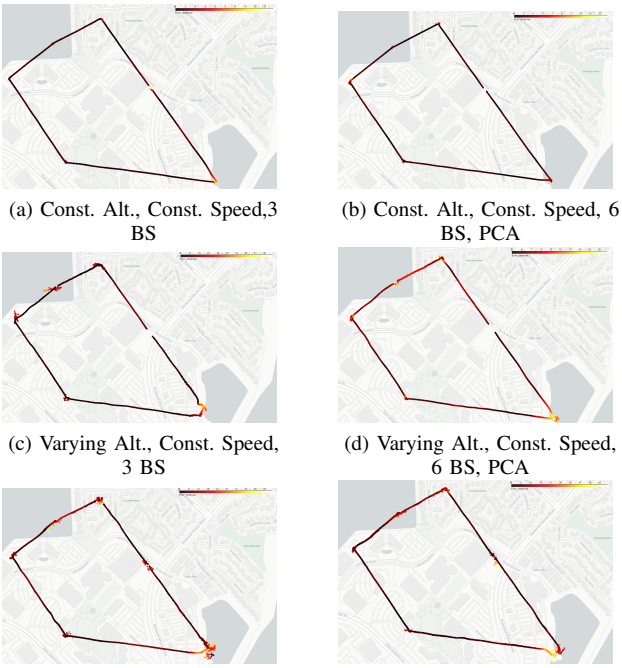
(f) Varying Alt., Varying Speed,6 BS, PCA

Fig. 5: Error along trajectory

We are comparing on Fig. 5 the error distances along the test trajectories. Among these models the LSTM looks the most stable, meaning there are a lot less spikes on the graphs than on the other two examined models. It is recommended to examine Fig. 5 and Fig. 6 together. On Fig. 6 we are focusing only for the LSTM model, since according the performance of the examined models that is the most accurate and stable. On the heat maps we can see where are the bends in the trajectory, which is an important information, since in some cases the error distances become the highest on those parts, meaning a the outliers/extremes could be observed there with the highest probability. The error distances calculated from the predictions by RNN model shows a highly oscillating pattern in the case of Fig. 5(f).

It is clear from these results, that the DNN model provided the worst performance among the three models and the average of the error distances is more than 80 m in each case, which is highly inaccurate compared to the other two models. Furthermore, the on average DNN model exhibits the highest standard deviation values, which indicates a greater variability in prediction errors compared to other models.

The RNN and LSTM models provided a lot more precise estimation for the position, which we expected because the next position was influenced by the previous one in the series of drone movements, allowing these models to effectively capture the patterns in the data. However, in the case or the RNN model more spikes/outliers/extremes can be identified.



(a) Const. Alt., Const. Speed,3 BS

(b) Const. Alt., Const. Speed, 6 BS, PCA

(c) Varying Alt., Const. Speed, 3 BS

(d) Varying Alt., Const. Speed, 6 BS, PCA

(e) Varying Alt., Varying Speed,3 BS

(f) Varying Alt., Varying Speed,6 BS, PCA

Fig. 6: Heatmap along trajectory, LSTM

The standard deviation of the error distances was always the lowest in the case of LSTM, which means among these models it provided the most consistent predictions.

The best result is achieved on the dataset where both the altitude and the speed was constant. The average of the error distances is 1 meter in that case. The predictions in this case can be seen on Fig. 6(b) and on Fig. 5(b). The best result achieved in the most complex case is $\varepsilon = 6.5$ [m] in Table I.

By doing PCA on the data we can acquire more precise predictions with the LSTM model, however the more pre-processing step on the data also increases the time of the prediction, which we should keep low especially in the case of a moving drone.

VI. Conclusion

In this paper we proposed a spatial prediction method for UAV positioning. The proposed method involves the application of LSTM model with PCA based preprocessing. Our proposed method overcomes the state of the art in terms of accuracy by roughly halving the error distance from 11.7 m in [4] to 6.5 m in our case. This accuracy is maintained even in the most complex case, when the drone had a varying speed and each of its coordinates are changed at the same time.

We conclude that increasing the number of BSs, it does not necessarily improves the location prediction. The movement of the drone and the spatial location of the BSs has significant effect how much is the data correlated. High correlation between features typically results in higher VIF values for those features. If two features are highly correlated, the R-squared value in the VIF calculation will be high, leading to a higher VIF value.

In this paper the antennas' radiation pattern were isotropic. As a further work, we plan to conduct experiments with other radiation patterns of the antenna. A further improvement area is the increase of the accuracy of the results by involving real-world data through additional experiments. An additional further improvement will be adding uncertainty estimation into the deep learning model, as a structure that provides both position and uncertainty may offer more robust performance compared to models that only estimate position.

References

[1] M. L, "Drone," https://youtube.com/playlist?list=PL07omI2d5EozkRyK6okhfcW5gsrz4178D, 2024, [Accessed: 2024-11-11].

[2] PX4 Autopilot, "Sensor selection," https://docs.px4.io/main/en/getting_started/sensor_selection.html, accessed April 7, 2024.

[3] J. Jepsen, F. Gunnarsson, R. Wen, J. Wase, and K. Jensen, "Experimental results of utilizing the 3gpp lte positioning protocol for providing rtk reference data to high-accuracy uav operations," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, ser. Proceedings of International Conference on Unmanned Aircraft Systems. United States: IEEE, 2024, pp. 53–60.

[4] M. Belhor, A. Savard, A. Fleury, P. Sondi, and V. Loscri, "Enhanced rf-based 3d uav outdoor geolocation: from trilateration to machine learning approaches," in *2024 IEEE 27th International Symposium on Real-Time Distributed Computing (ISORC)*, 2024, pp. 1–8.

[5] Federal Aviation Administration, "Uas beyond visual line of sight (bvlos) aviation rulemaking committee (arc) final report," 2022, accessed: 2024-10-08. [Online]. Available: https://www.faa.gov/regulations_policies/rulemaking/committees/documents/media/UAS_BVLOS_ARC_FINAL_REPORT_03102022.pdf

[6] K. Hirotsu, F. Granelli, and A. Nakao, "Lora-based localization for drones: Methodological enhancements explored through simulations and real-world experiments," *IEEE Access*, pp. 1–1, 2024.

[7] S. Liu, H. Lu, and S.-H. Hwang, "Three-dimensional indoor positioning scheme for drone with fingerprint-based deep-learning classifier," *Drones*, vol. 8, no. 1, 2024. [Online]. Available: https://www.mdpi.com/2504-446X/8/1/15

[8] W. G. Negassa, D. J. Gelmecha, R. S. Singh, and D. S. Rathee, "An efficient three-dimensional node localization using recurrent neural networks in unmanned aerial vehicle-assisted wireless networks: an optimization perspective," *International Journal of Intelligent Unmanned Systems*, Jan. 2024.

[9] "Gazebo Simulator Website," http://gazebosim.org, accessed: May 18, 2023.

[10] "Next-generation Simulation Technology To Accelerate The 5G Journey." [Online]. Available: https://www.ericsson.com/en/blog/2021/4/5g-simulation-omniverse-platform

[11] G. Szabó and J. Pető, "Intelligent wireless resource management in industrial camera systems: Reinforcement learning-based ai-extension for efficient network utilization," *Computer Communications*, vol. 216, pp. 68–85, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366423004607

[12] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 6235–6240.

[13] PX4 Autopilot Documentation, "Ros interface for gazebo simulation," 2024, accessed: 2024-09-28. [Online]. Available: https://docs.px4.io/main/en/simulation/ros_interface.html

[14] QGroundControl, "Qgroundcontrol - drone control," 2024, accessed: 2024-09-28. [Online]. Available: https://qgroundcontrol.com/

[15] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.