Cooperative AoA Wireless Positioning using LSTM Neural Network: Preliminary Results

Pedro M. Santos*, Luis Javier Puente Lam*†
*Cister Research Centre, Instituto Superior de Engenharia do Porto (ISEP)
†Faculdade de Engenharia da Universidade do Porto (FEUP)
Porto, Portugal
{pss,javie}@isep.ipp.pt

Abstract*—Wireless positioning (WP) enables ego or hetero-localization, e.g. for asset tracking applications. It can be provided by 3GPP technologies, complementing GPS with extra accuracy where conditions are challenging (e.g., urban canyons). We address scenarios where base-stations (BS) track a mobile user using a wireless Angle-of-Arrival (AoA) technique and, by exchanging angle estimates, can estimate the location of a target node. Cooperative position determination from angle estimates has been addressed by estimation techniques (e.g., Least Squares). We investigate if machine learning techniques, notably a Long Short-Term Memory Neural Network (LSTM-NN), can offer competitive performance. The LSTM is designed to be fed sequential error-affected angle measurements and output position estimates. We assume the path taken by the target mobile node is known beforehand. At training stage, the LSTM is trained to learn a subset of scenario locations (belonging to the node's path) with well-known positions; then, in runtime, it corrects position estimates for points in the whole path. The LSTM architecture is selected as it retains temporal relationship between input samples. Preliminary simulation results show competitive accuracy when the angle estimate error is relatively large (15°) with respect to a baseline value (5°).

***Index Terms*—Wireless positioning, Angle-of-Arrival, LSTM, Least Squares**

I. Introduction

Wireless positioning (WP) is gaining traction as an enabler of services such as autonomous driving, precision agriculture and asset tracking. Realising the potential of WP, 3GPP's Release 18 and future releases plan for considerable improvements in 5G NR positioning [1], including sidelink positioning [2]. In this paper, we look into the use-case of tracking the location of a target asset by network base-stations (BS). We leverage the well-established WP technique of Angle-of-Arrival (AoA), in which receivers equipped with antenna arrays determine the direction of the transmitter from phase differences in the received carrier. The resulting angle estimates are tainted with measurement error. Furthermore, in a strictly AoA-based approach, no distance can be obtained, and therefore no position estimate can be produced. To address these two issues, a cooperative approach can be explored, in which base-stations exchange data to improve accuracy of position estimates. Typical solutions for cooperative AoA positioning rely on estimation-based solutions, such as Approximated Maximum-Likelihood [3] or Least Squares [4]; Deep Learning (DL) solutions are less obvious.

In this paper we evaluate the accuracy involved in using a Long Short-Term Memory Neural Network (LSTM-NN) to cooperatively estimate the location of a target node from AoA estimates. At training stage, we feed the LSTM-NN with existing angle measurements from all BS to a set of **anchors**, i.e., locations with well-known positions, and train it to produce the exact position of the corresponding anchors. The anchors should be a subset of points in the trajectory the target mobile node will perform; thus, it is a requirement that the trajectory is known *a priori*. We rely on the ability of neural networks to interpolate and extrapolate beyond training data to produce accurate position estimates between anchors; and on the ability of LSTMs to retain sequential relationships, notably the sequence by which anchors are traversed. We compare our proposal against a benchmark solution, Least-Squares (LS). Preliminary results show competitive but not outperforming accuracy for a baseline scenario; however, the LSTM outperforms slightly LS when we increase angle error (std.dev.) from 5° (baseline) to 15°.

The remainder of this paper is as follows. Section II reviews relevant literature. The system model and used techniques are described in Section III. Section IV reports performance results, and Section V draws final remarks.

II. Related Work

Relative positioning has been a famous application of wireless networks [5]. Angle-of-arrival (AoA) is a well-established technique to determine the angle of the impinging signal. It presumes the existence of multiple antennas at the receiver (closer than the wavelength); based on the difference in phase of the received signal at the various antennas, the receiver can estimate the angle at which the transmitter is located (using correlation methods such as MUSIC [6]). Cooperative AoA methods explore the scenario in which multiple receivers

This work was supported by national funds through the FCT/MCTES, Portuguese Foundation for Science and Technology, within the CISTER, ISEP/IPP Research Unit (UIDP/UIDB/04234/2020), and also by project Route 25 (ref. TRB/2022/00061 - C645463824-00000063), funded by the EU/Next Generation, within call N.º 02/C05-i01/2022 of the Recovery and Resilience Plan (RRP).

exchange their AoA measurements and can estimate the transmitter's position by intersecting the AoA projections. In [3], a (cooperative) iterative approach to this problem is presented; the authors of [4] propose a closed-form solution.

Deep Learning (DL) techniques have been applied to trajectory estimation and AoA estimation. LSTMs are fairly popular for predicting vehicle trajectories as they retain temporal relationships; therefore, the typical application is the use of pre-existing trajectory datasets to forecast future vehicle positions [7], [8]. However, our work does not focus on forecasting trajectories, but rather on runtime accuracy improvement. A noteworthy reference that applies DL to AoA-based position estimation is [9]. It also addresses cooperative positioning relying on AoA estimates from cellular base-stations. The authors describe a CNN to produce a 2D Direction-of-Arrival (azimuth and elevation), that is then feed to a geometric framework to determine a position estimate. In our approach, we train our LSTM to take in angle estimates and produce position estimates, thus being more comprehensive. Other works are [10], with focus on improving position accuracy of the ego-vehicle from V2X data and AoA estimates from impinging signals (therefore user-centric, unlike our scenario); and [11], in which angle estimation is improved by training a Support Vector Machine (SVM) with real-world measurements of vehicular wireless channels (thus requiring extensive field campaigns to collect such experimental data).

III. LSTM-based AoA Positioning

A. Scenario & Premise

Our scenario assumes a set of **static wireless nodes** $\mathbf{n}$ (alike cellular base-stations) and one target mobile node that follows a known trajectory, refered to as **Node-of-Interest (NoI)**. The trajectory of NoI is assumed known a priori and described by **waypoints** w. The static nodes wish to obtain accurate estimates of the location of the NoI in runtime. To that end, the nodes are able to perform Angle-of-Arrival (AoA) estimation and to run locally a localization procedure based on exchanged angle estimates. Communication between static nodes is presumed instantaneous. Figure 1 represents the scenario (orange: NoI; green: static nodes).

When performing angle estimation, AoA estimates have a measurement error ϵ_α, modelled as a normal distribution $\mathcal{N}(\mu_\alpha, \sigma_\alpha)$. The angle error ends up affecting the quality of the position estimate. Several estimation-based solutions have been proposed (see Section II); by contrast, we investigate the performance of a learning-based approach. We select two procedures, a traditional estimation-based approach to serve as benchmark, and one based on neural networks; these are:

- (non-iterative) Least Squares technique (Sec. III-C).
- Long Short-Term Memory Neural Network (LSTM-NN) (Sec. III-D).

B. Rationale for Learning Approach

We use a supervised learning approach, meaning that the LSTM must first be trained with existing information. The input dataset are error-affected angle estimates to locations in

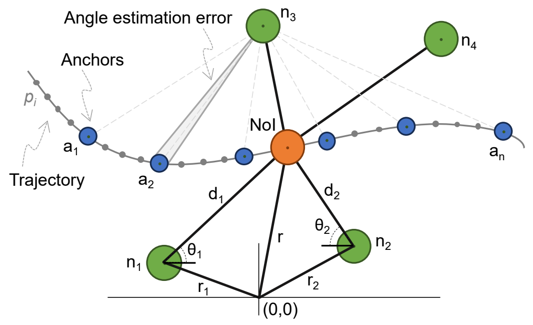


Fig. 1: Nodes $\mathbf{n}$ position a Node-of-Interest (NoI) from AoA estimates on a known path. Angle measurements to a subset of points, named anchors $\mathbf{a}$, are used to train the LSTM; at runtime, LSTM estimate NOI position for all points $\mathbf{p}$ in path.

the scenario referred to as **anchors** $\mathbf{a}$, that are a subset of the trajectory followed by the NoI (in Figure 1 – blue: anchors; grey: trajectory points); and the desired output of the neural network is the actual position of a_i. As explained earlier, we define trajectories as a set of waypoints w; currently, we make anchors to be co-located with waypoints (hence, for the remainder of this discussion, we will use the two terms interchangeably). Furthermore, as the LSTM retains temporal relationships between samples, input points are fed in the same order as the trajectory is traversed.

Ultimately, we wish to have good positioning accuracy not only near anchors (where we have good characterization of the positioning error), but also **between** anchors. The trajectory followed by the NoI can be described as an ordered set of points $\mathbf{p}$ between anchors set at some arbitrary distance step (e.g., 1 meter in between). To facilitate the use of the ML technique in real conditions, we aim to have the set of anchors $\mathbf{a}$ to be a small subset of $\mathbf{p}$; in that way, from the statistical characterization of the angle error at a few locations, we get good positioning accuracy across the whole trajectory. During runtime, as the NoI tranverses the set of points $\mathbf{p}$, nodes $\mathbf{n}$ compute their angle to the NoI and feed them as inputs to the trained model; in turn, the trained model produces estimates of the NoI position.

With respect to the LS approach, there are two limitations to the LSTM-NN approach: it requires samples of angle estimates from all static nodes to known locations (the anchor points), and it requires previous knowledge of the trajectory of the NoI. We assume these to be acceptable requirements in situations where vehicles perform the same trajectory routinely, e.g. buses or garbage collection trucks. We leave it for future work to re-assess the requirement of prior knowledge of the trajectory. In contrast, the LS method operates solely at runtime, i.e., it only requires the angle estimates to the NoI.

C. Least Squares Closed-Form Solution

The authors of [4] presented a closed-form solution for positioning based on AoA measurements. The position (x, y) of the NoI $\mathbf{r}$ can be obtained from (refer to Figure 1, bottom):

$$\mathbf{r} = \mathbf{r_i} + d_i \cdot \mathbf{v_i} \tag{1}$$

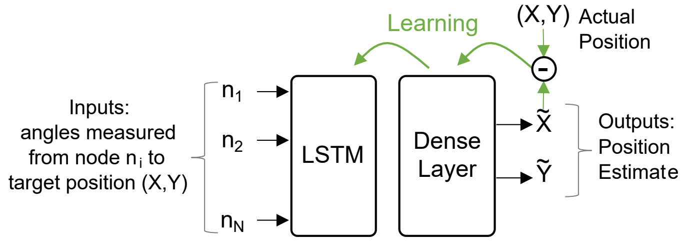

Learning
(X,Y) Actual Position
-
Inputs: angles measured from node n_i to target position (X,Y)
n_1
n_2
n_N
LSTM
Dense Layer
X̃
Ỹ
Outputs: Position Estimate

Fig. 2: Architecture of developed LSTM neural network.

where $\mathbf{r_i}$ is the location of node n_i, d_i is the range between n_i and the NoI, and $\mathbf{v_i}$ is the unitary vector between n_i and the NoI.

$$v_i^T = [\cos\theta_i \qquad \sin\theta_i] \tag{2}$$

Given that we do not have the distance, some mathematical re-arrangement allows us to arrive to the following expression:

$$\begin{bmatrix} -x_1\sin\theta_1 + y_1\cos\theta_1 \\ \dots \\ -x_n\sin\theta_n + y_n\cos\theta_n \end{bmatrix} \approx \begin{bmatrix} -\sin\theta_1 & \cos\theta_1 \\ \dots & \dots \\ -\sin\theta_n & \cos\theta_n \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

Eq. 3 can be written more conveniently as

$$\mathbf{b}(\theta) \approx \mathbf{H}(\theta)\mathbf{r} \tag{4}$$

The well-known least squares (LS) solution can be applied, defined as

$$\mathbf{r}^{\mathrm{LS}} = \left[\mathbf{H}^{\mathrm{T}}(\theta)\mathbf{H}(\theta)\right]^{-1}\mathbf{H}^{\mathrm{T}}(\theta)\mathbf{b}(\theta) = \mathbf{H}^{\#}(\theta)\mathbf{b}(\theta) \tag{5}$$

where $\mathbf{H}^{\#}(\theta)$ is the pseudo-inverse of $\mathbf{H}(\theta)$. The existence of a solution depends on $\mathbf{H}^{\#}(\theta)\mathbf{b}(\theta)$ being invertible. We refer to [4] for additional details.

D. Neural Networks for Cooperative AoA Positioning

The adopted neural network is composed of one LSTM Input layer followed by one Dense layer. We use an LSTM Recurrent Neural Network for their ability to retain temporal relationships between input points. The LSTM Input layer is composed by $|\mathbf{a}|$ discrete time steps (as many as the waypoints/anchors in the path) and admits, for time step i, an input vector containing the information of the angles of each static node n_j to the anchor point a_i, i.e., $\alpha_{\mathrm{i,j}}(i \in \{1, ..., |\mathbf{a}|\}$, $j \in \{1, ..., |\mathbf{n}|\})$. Angles defined in the $[0°, 360°]$ are normalized to the $[0, 1]$ range. In turn, the Dense layer has 128 neurons implementing the ReLu activation function; a ReLu activation function is also applied to the output of the LSTM layer. The Dense layer has two outputs, describing the X and Y-coordinates of the position estimate $\tilde{z}$. Figure 2 depicts the adopted architecture.

IV. Analysis

We developed a simulator of the system model described in the previous section to evaluate the performance of the proposed method. For statistical significance, we create a set of scenarios over an 2D grid of 10×10 10m-wide cells. One characteristic differentiating scenarios is the $|\mathbf{n}|$ locations (x, y) of the static nodes n. These are sampled from an uniform distribution, with care to balance node distribution throughout all quadrants (each quadrant gets one fourth of the nodes; if nodes remain, they are distributed uniformly through the quadrants). The second characteristic of each scenario is the NoI trajectory; these are produced by random walks, defined by intersection-bound waypoints w. To generate the random walk, at each waypoint, the direction of the next waypoint is picked from an uniform distribution. Visited points are not visited again.

In the current setup, we make the anchor points $\mathbf{a}$ to be the same as waypoints. The true angles between anchors and static nodes $\alpha_{\mathrm{i,j}}$ are computed, and for each anchor-node pair we generate 100 error-affected angle estimates $\tilde{\alpha}_{\mathrm{i,j}}$, by adding error components ϵ_α drawn from a Gaussian distribution with mean μ_α=0 and standard deviation σ_α to the true angle value. For training the LSTM, we use the Mean Square Error (MSE) loss criterion (as this is a regression problem) and Adam optimizer with learning rate l_{r}. We perform training for a given scenario in batches of $N_{\mathrm{ep\_batch}}$ epochs until $N_{\mathrm{ep\_max}}$ epochs. We stop training if, at the end of a batch, the loss value is below a threshold loss value $l_{\mathrm{thr\_exit}}$ or if the maximum number of epochs $N_{\mathrm{ep\_max}}$ is met. We allow the algorithm to re-initiate training from scratch for a given scenario if it is not converging by the end of the first or fifth batch.

For model testing, the actual trajectory of the target node is produced. We interpolate trajectory points $\mathbf{p}$ between every two sequential waypoints at a distance step of $1m$. We then compute the angles from all p_i to all nodes n_i, and add a noise sample drawn from the normal distribution $\mathcal{N}(\mu, \sigma_\alpha)$ (as done for the training set).

Development of the mobility and wireless simulator and implementation of the Least Squares and LSTM mechanisms was carried out in Python 3.9.7. Specifically for the LSTM, we used PyTorch and leveraged CUDA acceleration for faster processing (Nvidia driver 552.22, Cuda version 12.4 over Nvidia GeForce RTX 3050). To characterize the positioning error, we compute the Euclidean distance between estimate and actual location, i.e., $|z - \tilde{z}|$. Table I shows most scenario and methods parameter values. We ran simulations for N_{scen} =30 different scenarios (different node locations and trajectories). Figure 3 shows the empirical cumulative distribution function (ECDF) of the path length of the 30 produced scenarios (i.e., number of waypoints/anchors in each scenario).

TABLE I: Parameter values (and alternative when applicable)

Name	Baseline value [/ Alternative]
Scenario dimensions	100x100 m
Nr. static nodes $\|\mathbf{n}\|$	4 / 12
Nr. different scenarios N_{scen}	30
Angle error σ_α	5° / 15°
Nr. samples/anchor	100
Nr. hidden layers	2
Nr. inputs	Nr. static nodes $\|\mathbf{n}\|$
Nr. outputs	2 (X & Y)
Nr. Epochs Batch $N_{\mathrm{ep\_batch}}$	2000
Nr. Maximum Epochs $N_{\mathrm{ep\_max}}$	20000
Learning rate l_{rate}	0.01
Loss threshold for exit $l_{\mathrm{thr\_exit}}$	0.00150

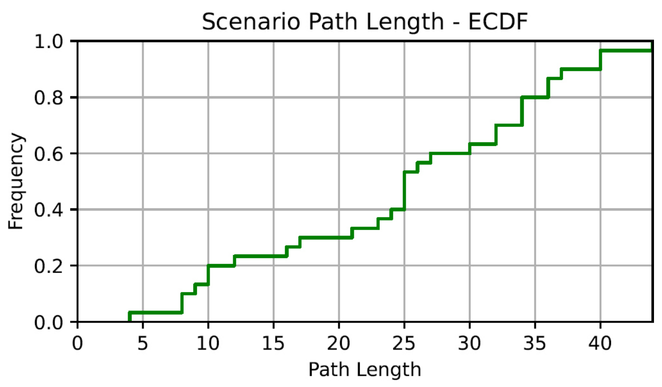


Fig. 3: Histogram of produced path lengths.

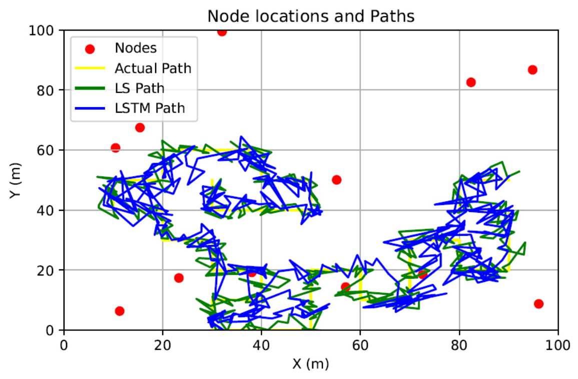
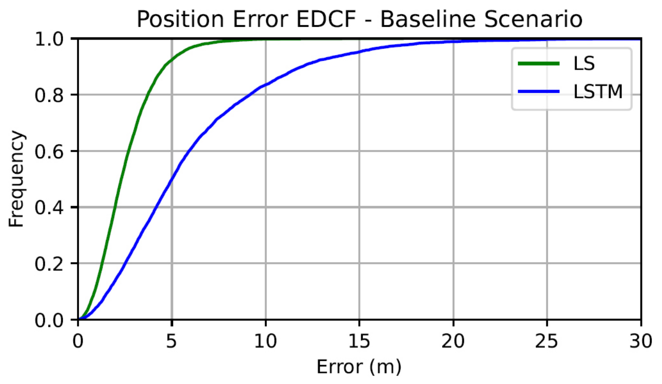
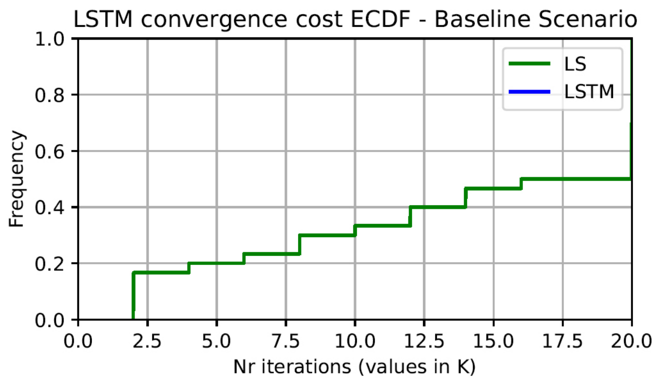


Fig. 4: Example scenario (nodes in red, trajectory in yellow), and position estimates from LS (green) and LSTM (blue).



(a) EDCF of error to actual position.



(b) EDCF of LSTM convergence cost.

Fig. 5: Baseline scenario.

A. Baseline Results

We evaluate the performance of the LSTM and LS using the default values of Table I. Figure 4 shows the example of a scenario with 10 static nodes, the actual trajectory (yellow), and the LS (blue) and LSTM (green) estimated trajectories. Figure 5 shows on the first subfigure (left) the ECDF of the position error at all p trajectory points (from all N_{scen} scenarios). The second subfigure (right) shows the ECDF of the convergence cost of the LSTM across scenarios, i.e., the number of epochs required to produce satisfactory results (loss below $l_{\text{thr\_exit}}$) for each scenario, and in how many scenarios the maximum number of epochs was used up.

We observe that in this baseline scenario, the LSTM performs worse than the LS approach. Under LS, 92.4% of the samples have errors inferior to 5m, whereas this value drops to 49.8% LSTM (Figure 5a). Figure 5b informs that 50% of scenarios used 20 000 epochs (i.e., the maximum allowed).

B. Number of Nodes

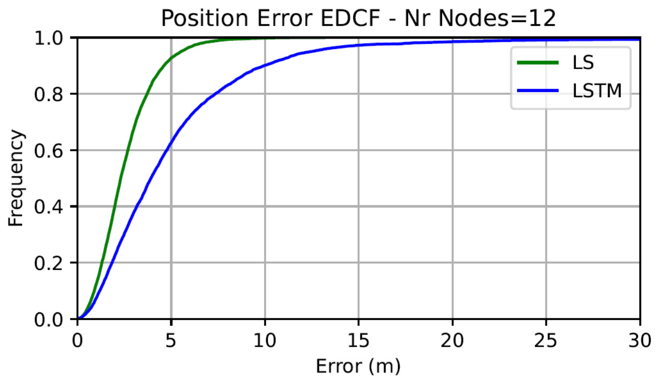
We alter the number of static nodes from 4 to 12. We keep a multiple of 4 to keep node distribution homogeneous across quadrants. Figure 6a shows that overall accuracy improves for LSTM: errors up to 5m affect 62.6% for LSTM, whereas for the baseline scenario this value would be close to 50%. LS reports little improvement (92.7% of samples under 5m error) but still outperforms LSTM. As for convergence cost of the LSTM training, 40% of the scenarios required the full set of allowed training epochs (Figure 6b).

C. Angle Error

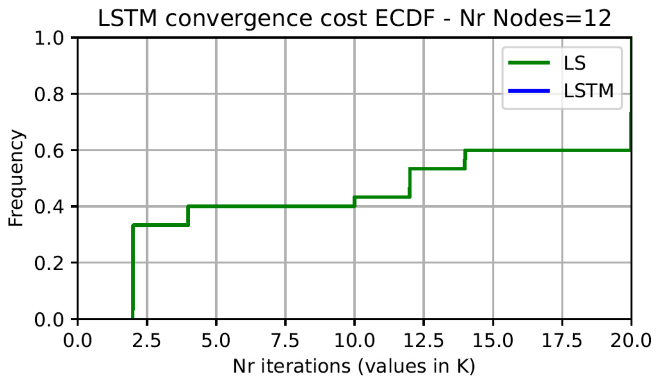
We alter the standard deviation of the angle estimation σ error from 5° to 15°. Figure 7 shows that the accuracy degradation of the angle estimates impacts both methods considerably. Errors up to 5m apply only to 25.4% and 26.76% of samples for LS and LSTM respectively. However, unlike previous scenarios, we observe that the LSTM approach outperformed the LS technique (albeit marginally). Only 5 samples of LSTM in 7854 total produced larger error than LS's; and the LS results are on average 6.10% higher than the LSTM's. These results required training for all scenarios to use up all available epochs (graph obviated).

V. Conclusion

We evaluate the accuracy and overhead of performing cooperative positioning via AoA estimates using an LSTM. LSTM showed inferior performance to a benchmark method (Least Squares) in a baseline scenario, but it became competitive when the standard deviation of the angle error increased



(a) EDCF of error to actual position.



(b) EDCF of LSTM convergence cost.

Fig. 6: Number of nodes = 12.

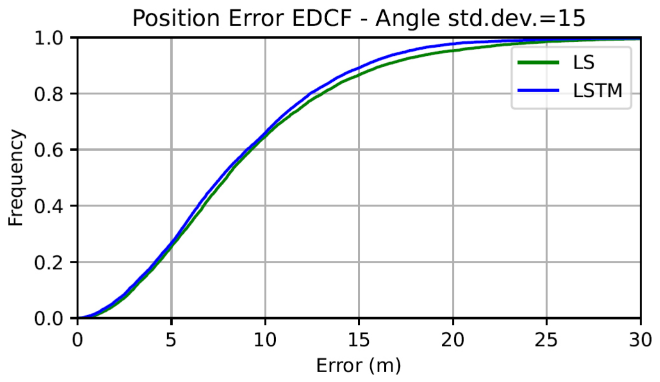


Fig. 7: EDCF of position error - Angle error std.dev.=15°

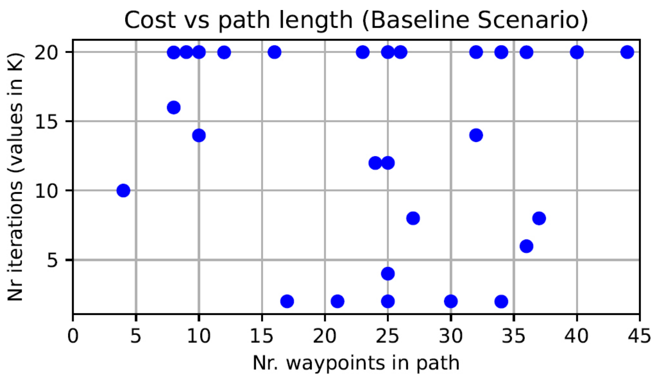


Fig. 8: Cost as a function of number of waypoints in scenario.

considerably, from 5° to 15°, with the LS estimates being on average 6.1% larger than the LSTM's.

Convergence cost remains a core point to study. We investigated if cost could be proportional to the path length, but found no discernible relationship (see Figure 8). The learning rate l_{rate} was an important factor; we found empirically values l_{rate}=0.1 and l_{rate}=0.001 to not converge in reasonable time. From the analysis of the various scenarios, we draw the hypothesis that offering a larger maximum number of epochs for training may improve accuracy (at the cost of additional computation), but this requires further study.

As future work we will test different structures for the LSTM. Different arrangements of the LSTM and Dense layer may take better advantage of the LSTM memory capabilities. We will also investigate approaches that relax the requirement of knowing the trajectory *a priori*, notably using a type of neural network that does not retain temporal relationships and performing training over all the intersections of the grid.

REFERENCES

[1] H.-S. Cha, G. Lee, A. Ghosh, M. Baker, S. Kelley, and J. Hofmann, "5G NR Positioning Enhancements in 3GPP Release-18," Jan. 2024.

[2] T.-K. Le, S. Wagner, and F. Kaltenberger, "5G Sidelink Positioning in 3GPP Release 18 and Release 19," in *2023 IEEE Conference on Standards for Communications and Networking (CSCN)*. Munich, Germany: IEEE, Nov. 2023, pp. 171–176.

[3] D. J. Torrieri, "Statistical theory of passive location systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, no. 2, pp. 183–198, 1984.

[4] A. Pages-Zamora, J. Vidal, and D. Brooks, "Closed-form solution for positioning based on angle of arrival measurements," in *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 4. Lisboa, Portugal: IEEE, 2002, pp. 1522–1526.

[5] N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Magaz.*, vol. 22, no. 4, pp. 54–69, 2005.

[6] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. on Antennas and Prop.*, vol. 34, no. 3, pp. 276–280, 1986.

[7] A. Ip, L. Irio, and R. Oliveira, "Vehicle Trajectory Prediction based on LSTM Recurrent Neural Networks," in *IEEE 93rd Vehicular Technology Conf. (VTC2021-Spring)*. Helsinki, Finland: IEEE, 2021, pp. 1–5.

[8] L. Rossi, A. Ajmar, M. Paolanti, and R. Pierdicca, "Vehicle trajectory prediction and generation using LSTM models and GANs," *PLOS ONE*, vol. 16, no. 7, p. e0253868, Jul. 2021.

[9] Y. Tian, S. Liu, W. Liu, H. Chen, and Z. Dong, "Vehicle Positioning With Deep-Learning-Based Direction-of-Arrival Estimation of Incoherently Distributed Sources," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20 083–20 095, 2022.

[10] A. Fascista, G. Ciccarese, A. Coluccia, and G. Ricci, "Angle of Arrival-Based Cooperative Positioning for Smart Vehicles," *IEEE Transactions on Intelligent Transp. Systems*, vol. 19, no. 9, pp. 2880–2892, Sep. 2018.

[11] M. Yang, B. Ai, R. He, H. Chen, Z. Ma, and Z. Zhong, "Angle-of-Arrival Estimation for Vehicle-to-vehicle Communications based on Machine Learning," in *2020 Int'l Conf. on Wireless Communications and Signal Processing (WCSP)*. Nanjing, China: IEEE, Oct. 2020, pp. 154–158.