Proceedings of AIAA SCITECH 2022 Forum, 3-7 January, 2022, San Diego, CA, USA and Virtual Event
Paper number AIAA 2022-1759, DOI:10.2514/6.2022-1759

An INS/GNSS fusion architecture in GNSS denied environments using gated recurrent units

Patrick Geragersian[1],Ivan Petrunin[2] and Weisi Guo[3]
Cranfield University, Bedfordshire, MK43 0AL, United Kingdom

Raphael Grech[5]
Spirent Communications PLC, Aspen Way, Paignton, Devon TQ4 7QR, United Kingdom

One of the most used Position, Navigation and Timing (PNT) technology of the 21st century is Global Navigation Satellite Systems (GNSS). GNSS signals are affected by urban canyons that limit line-of-sight and reduce satellite availability to receivers. Smart cities are expected to adopt autonomous Unmanned Aerial Vehicles (UAV) operations for critical missions such as transportation of organs which are time-sensitive. Therefore, higher accuracy for position and velocity information is required. This paper investigates the use of Gated Recurrent Units (GRU) as a suitable technique that can memorize previous information in conjunction with the inputs (consisting of attitude, change in attitude, and change in velocity) to reduce position and velocity error when GNSS is not available. The fusion approach is developed and tested using Spirent's SimGEN GSS7000 hardware simulator which simulates GNSS signals and Spirent's SimSENSOR software that simulates accelerometer and gyroscope stochastic and deterministic errors. GNSS outage is varied between 1 and 20 seconds randomly to affect predicted position and velocity. The data is collected and used to train the GRU to predict the position and velocity error measured by the Inertial Measurement Unit (IMU). From the performance evaluation, a 60% reduction in Root Mean Squared Error (RMSE) is observed compared to Recurrent Neural Networks (RNN). Comparing 95th percentile with Inertial Navigation System (INS), RNN, and GRU, an 80% reduction is observed between INS and RNN. Furthermore, a 35% drop in the 95th percentile is observed between RNN and GRU.

I. Nomenclature

ARW	=	Accelerometer Random Walk
DCM	=	Direction Cosine Matrix
EKF	=	Extended Kalman Filter
GNSS	=	Global Navigation Satellite System
GPS	=	Global Positioning System
GRU	=	Gated Recurrent Unit
GRW	=	Gyroscope Random Walk
IMU	=	Inertial Measurement Unit
INS	=	Inertial Navigation Unit
LSTM	=	Long Short-Term Memory
RMSE	=	Root Mean Squared Error

[1]PhD candidate, School of Aerospace, Transport and Manufacturing (SATM), Cranfield University
[2]Lecturer, Centre for Autonomous and Cyberphysical Systems, Cranfield University
[3]Professor of Human Machine Intelligence, Centre for Autonomous and Cyberphysical Systems, Cranfield University
[5]Technical Strategist in Emerging Technologies, Spirent Communications PLC

RNN = Recurrent Neural Network
UAV = Unmanned Aeriel Vehicle

II. Introduction

Rapid urbanization has transformed human settlements into increasingly complex urban landscapes. Not only are there "urban canyons" that limit Line-Of-Sight and create environments that allow multiple signals from the same satellite to arrive at a GNSS receiver, but also present challenges in increased interference from other radio systems [1]. This is especially true for autonomous vehicles where full self-control is desired for smarter cities [2]. These cities could deploy autonomous air and ground vehicles for various tasks including critical missions such as disasters and pandemics.

Sensors, such as GNSS receivers and IMUs, can provide positioning and orientation data (as defined by the roll, pitch, and yaw) that can be used by navigation systems. However, these sensors/receivers can provide erroneous data that will impact their performance. For GNSS receivers, these problems (besides those mentioned above) are clock errors, ionospheric and tropospheric delays, and receiver noise to mention a few [3]. IMUs are typically composed of an accelerometer and gyroscope. Both these sensor readings are affected by various errors such as biases (including random walk) and scale factor errors [4]. Typically, these errors are split into deterministic and stochastic errors. Therefore, it is of benefit to use different sensors together to mitigate/reduce some of these errors.

Traditional methods to combat these issues include using multiple sensors/receivers to provide a better estimate of position. A classic Bayesian fusion technique is the Kalman Filter. Kalman filters continuously calculate an estimate of the system state based on sensor inputs. Two steps are used in a Kalman filter, prediction, and update. The advantage of using Kalman filters is that of low computational cost due to no older timesteps being memorized. However, Kalman filters can only represent linear systems with gaussian error distribution. Extended Kalman filters solve this problem by using a Taylor series expansion to linearize a model. However, highly non-linear systems will not be represented well and this adaptation of the Kalman filter still assumes Gaussian error distribution [4-5,15]. Another adaptation of the Kalman filter is the Unscented Kalman filter (UKF). UKFs use approximate known statistical distributions by determining the minimum set of points around the mean that will then be able to describe the true means and covariance of the statistical distribution [15]. Therefore, there is no need to linearize the model. Though, the approximations are not global. They are based on a small set of trial points. Moreover, these systems can only be applied well to models that are driven by gaussian noises [16]. Additionally, for prolonged GNSS outages or inaccuracies when INS/GNSS signals are used, true and estimated positioning diverge over time as heavy reliance is placed on the INS [7]. Depending on the application/mission, this may not be a method that could be relied on. Therefore, a fusion method that can mitigate these issues is highly desired. To be able to deliver better positioning and velocity information, improvements in accuracy are required over the traditional Kalman filtering methods.

Recent methods to overcome these issues include artificial neural networks (ANN). Some of these proposed methods have included multi-layer perception (MLP) and radial basis function neural network (RBFNN) to predict INS errors during Global Positioning System (GPS) outages [8-9]. Currently, MLP has real-time implementation problems and RBF networks are not able to consider past error dependencies. To improve on the mentioned issues, an alternative neural network proposed is input-delayed neural networks (IDNN). Nevertheless, the major issue with IDNN is the computational cost and training time that is required [10]. This, therefore, requires a new neural network-based fusion architecture that would solve the issues discussed. Recurrent Neural Networks (RNN) is a type of ANN that has the advantage of collecting records so that each is dependent on the previouhases [11]. However, RNN have difficulties solving problems that require learning long-term dependencies due to the gradient of the loss function decaying exponentially over time [12-13]. Candidate techniques include using high dimensional memory points or Gated Recurrent Units (GRUs) which solve the issue of the exponentially decaying loss function by using update and reset gates. The reset gate is used to determine how much of the past information it has received needs to be forgotten. This is done by multiplying the weights with the hidden state h_{t-1} and input x_t. They are then added together, and a sigmoid function is used to squash the results between 0 and 1. The update gate is used to determine how much of the previous information is required to be carried forward. This is like the reset gate, but different weights are used when multiplying the input and the hidden state [13].

Therefore, the paper proposes a method of predicting errors in cases where there are missing/ inaccurate GNSS updates This architecture is then tested on the Spirent SimGEN GSS7000 and SimSENSOR to provide a realistic testing scenario. The rest of the paper is organized as follows: first, the state of the art in sensor fusion is discussed and some of the existing RNN papers are discussed for reference points in Section III. In Section IV, the proposed fusion architecture is presented and explained. Hereafter, the testing methodology used to test the proposed architecture is discussed in Section V. Lastly, the results are discussed and explained in Section VI.

III.Fusion Approaches

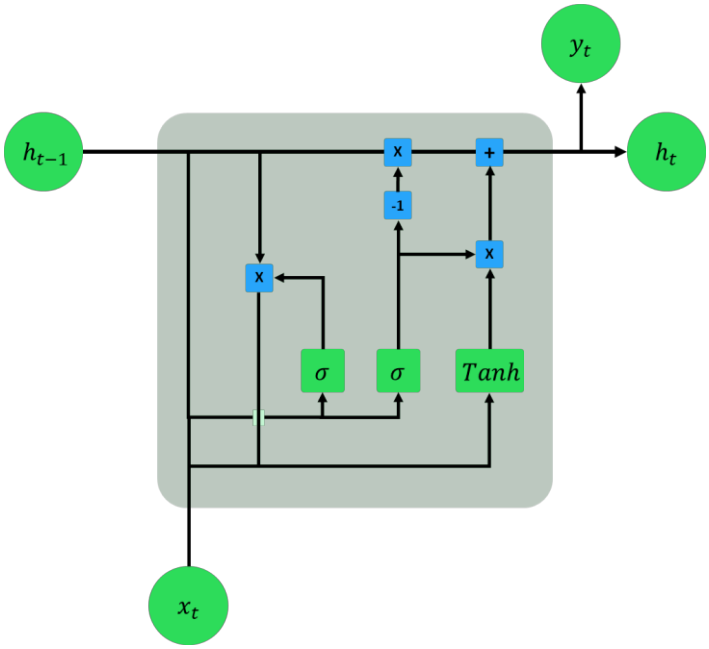

Figure 1. Inside a Gated Recurrent Unit

Recurrent Neural Networks are a class of artificial neural networks that are specifically used for sequences prediction problems. Derived from feedforward neural networks by David Rumelhart, they use part of the previous state outputs as part of the current input and therefore establishing a relationship between the output and the input [17]. These specific neural networks have been used primarily in handwriting and speech recognition. Recurrent Neural Networks have been used for sensor fusion in the past. The RNN algorithm is shown in equation 1 and 2[18]:

$$h_t = \sigma_h(W_{hh}h_{t-1} + W_{xh}X_t + b_h) \quad (1)$$

$$y_t = \sigma_y\text{-}W_y h_t + b_y \quad (2)$$

where:

h_t is the hidden layer vector, y_t is the output vector, X_t is the input vector, W_{hh}, W_{xh}, W_y are the parameter matrices, b_h, b_y are the bias terms and σ_h, σ_y are the activation functions. RNNs take the information from the previous state h_{t-1} and multiply it with a weight matrix. The same is done for the new input X_t and is then combined with the previous state to create the new hidden state h_t. Because of the way RNN is structured, it allows for information in the past to be linked with the information at the current timestep.

The activation functions (Tanh) are used to assess the sum weights of the input and decide which information is needed. However, RNNs suffer from vanishing and exploding gradient problems that may only provide a short-term memory for the past information being inputted. It also means that some information can be propagated even though the information is not useful [13]. Two types that are derived from RNNs (GRU and LSTMs) solve these issues.

LSTMs and GRUs use gates to determine whether the input information should be kept or should be removed. In a GRU, there are 2 gates that deal with this. An update gate is used to help the neural network determine how much information from, the previous time step needs to be passed to the next timestep. GRU architecture is shown in Figure 1. The formula for this is presented below [13]:

$$z_t = \sigma(W^z X_t + U^z h_{t-1}) \tag{3}$$

The reset gate is used to determine how much of the past information should be forgotten to improve the performance of the GRU. The formula is presented below:

$$r_t = \sigma(W^r X_t + U^r h_{t-1}) \tag{4}$$

These gates aid in removing the issues related to exploding and vanishing gradients. However, due to the additional gates, this adds computational complexity to the neural network which may slow down the training process. LSTMs on the other hand use 3 gates to solve these issues. These are the input gate, output gate and a forget gate. Because LSTMs use 3 gates, they are more computationally expensive than GRUs but can provide more accurate information for training the model [19].

One example of RNN being used is the work done by Dai et. Al. on an INS/GNSS integration using RNN. The paper explains how information from its GNSS and INS sensor are used to train RNN in detecting positioning and velocity errors. The author compares their proposed architecture against EKFs and extreme learning machine (EML). The results show up to 60% performance improvement when compared to the RMSE output of the EKF [20]. However, RNN suffers from gradient loss function which provides only a short-term memory solution as mentioned previously.

Another paper focuses on using an LSTM (Long Short-Term Memory) based architecture to predict horizontal position using GNSS and IMU. The author tests this algorithm on experimental data collected by driving a car in an open field. The solution demonstrates a 40% improvement compared to GNSS-only navigation without any external bias information [21]. However, there are some important issues to highlight with this paper. First, the test is only carried out on the horizontal plane. Whilst this may be sufficient for ground-based vehicles, UAVs also rely on the vertical axis information. Furthermore, the test was carried out in an open field. Therefore, the architecture proposed is not tested in urban environments where other errors such as multi-path or blockages occur. Lastly, LSTMs are more computationally expensive compared to RNNs or GRU as mentioned previously. Therefore, this may not be suitable for a small UAV

IV. Proposed Solution

IMU sensors measure the UAVs specific forces and angular velocities along the body frame. These values are integrated in the system to provide the position and velocity data to the navigation system. However, integrating these values will lead to a divergence between ground truth and INS position and velocity output due to the biases included in the measurements. To understand how the errors may propagate through the calculation, it is important to derive the position and velocity equations. The velocity of the vehicle is given by [20,22]:

$$\boldsymbol{v}_{eb}^{n}(k) \approx \boldsymbol{v}_{eb}^{n}(k-1) + \left(\boldsymbol{C}_{b}^{n}\boldsymbol{f}_{ib}^{b} + \boldsymbol{g}^{n} - 2\boldsymbol{\Omega}_{in}^{n}\boldsymbol{v}_{eb}^{n}(k-1)\right)\tau \tag{5}$$

Where $\boldsymbol{v}_{eb}^{n}$ is the UAVs velocity with respect to the earth in the local navigation frame, $\boldsymbol{C}_{b}^{n}$ is the direction cosine matrix from the body frame to the local navigation frame, $\boldsymbol{f}_{ib}^{b}$ is the specific force in the body frame, $\boldsymbol{g}^{n}$ is the gravitational vector in the local navigation frame and $\boldsymbol{\Omega}_{in}^{n}$ is the skew-symmetric matrix of the earth's angular velocity. To obtain the position of the UAV, the equation above needs to be integrated once more. The position of the vehicle is given by:

$$\boldsymbol{p}_{eb}^{n}(k) = \boldsymbol{p}_{eb}^{n}(k-1) + \left(\boldsymbol{v}_{eb}^{n}(k-1) + \boldsymbol{v}_{eb}^{n}(k)\right)\frac{\tau}{2} \tag{6}$$

To understand the influence of the measurement noises experienced, the actual velocity and position are given by the sum of the true measurement and the biases. The current position and velocity error are related to the vehicle dynamics,

previous INS error, and the environment. The relationship between these factors and the current position and velocity error is highly non-linear thus current techniques may not be enough to provide the accuracy required by UAVs when operating in an urban environment and experiencing GNSS outages. Therefore, a system is required that can find the non-linear relationship.

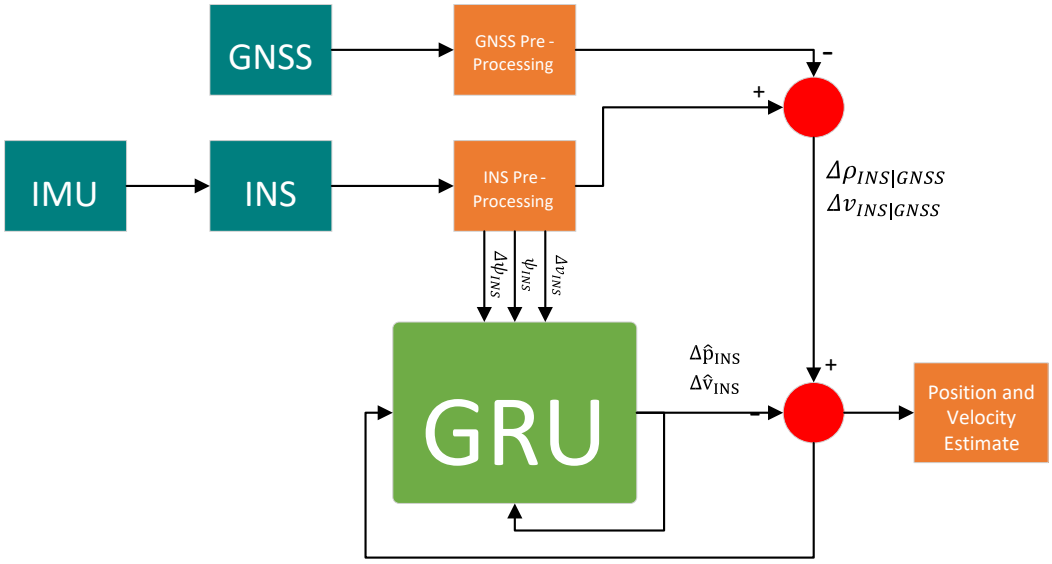

Figure 2. Fusion architecture for training and testing

The proposed system architecture is shown in Figure 2. Consisting of two modes, the first mode is used for the training of the system. Data generated by the INS sensor and GNSS receiver are used to gather positioning and velocity data. Furthermore, the vehicle's specific force and angular velocity measured by the accelerometer and gyroscope are used to calculate the input for the GRU. Therefore, the input to the GRU is the vehicle change in velocity and the vehicle change in attitude. The output of the system is the position and velocity error generated by the INS during normal operations. To measure the progress of the training, estimated residuals for positioning and velocity are compared to the GNSS standard deviation from each information source. Once enough confidence is provided by the GRU, the system is ready to move to its second mode.

The second mode is the prediction phase of the neural network. The prediction mode is used when GNSS information is not available. The trained GRU is used to predict the positioning and velocity error for the given INS information input. The predicted errors are then subtracted from the INS velocity and position readings to provide the system prediction. This will be carried out until GNSS information is available again.

V. Simulation Setup

To evaluate the proposed architecture, training of the architecture must be carried out. To obtain GNSS and IMU data, Spirent GSS7000 simulator is used with SimSENSOR to provide realistic training scenarios. The route is created to represent different maneuvers typically done by a drone. This is shown in Figure 5. Ground truth, GPS L1, gyroscope, and accelerometer data are collected and processed to obtain the velocity and position data. The GNSS signal is interrupted for a random duration between 1 and 20 seconds to simulate typical GNSS outages. This is then used to calculate the input and output data for the GRU. The input is obtained by taking the readings from the accelerometer and gyroscope and calculating the attitude and velocity. Periodically, GNSS updates are received to correct the IMU error when available to emulate outages. The input also includes information on how long ago the last GNSS update was received by the system. The output from the GRU is the Position and Velocity Error predicted. This is shown visually in Figure 3. The predicted errors are then subtracted from the INS readings to obtain the estimated true Position and Velocity data. This is then compared to the ground truth to evaluate the architecture performance.

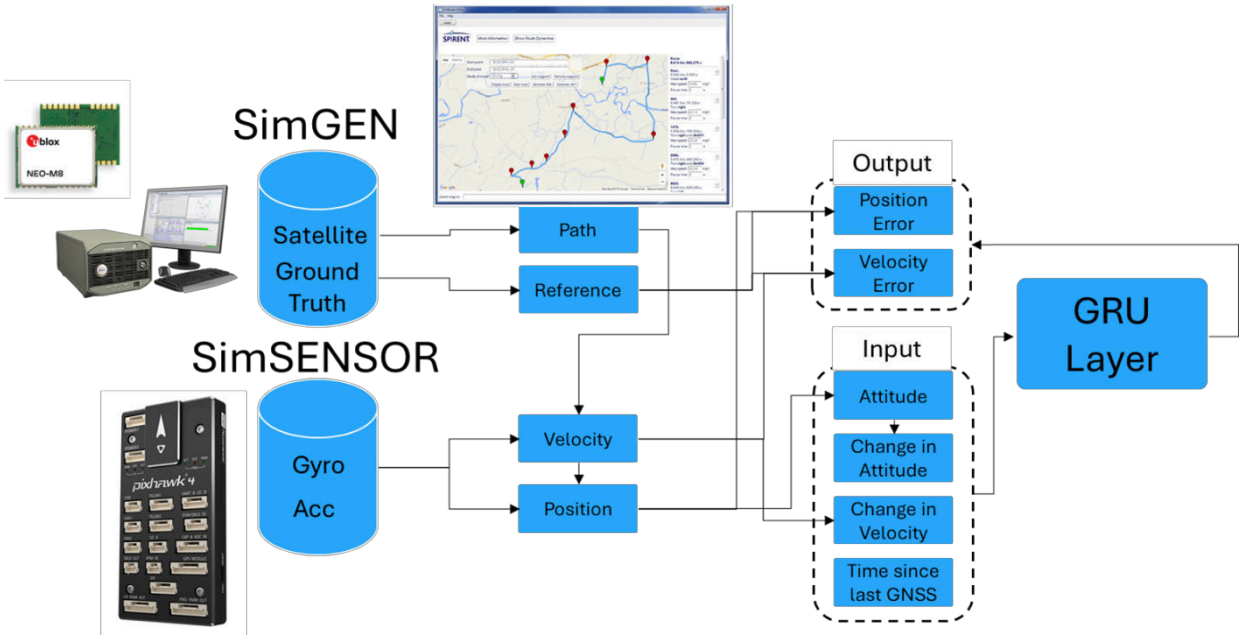

Figure 3. Data model from recording to input/output for neural network architecture [23-25]

The machine learning algorithm is optimized using random search to find the optimal hidden units, number of GRU layers, and best-performing activation function. Random Search only uses a small percentage of the total data to determine the best parameters. Priority is placed on both reducing the computational cost and improving the prediction accuracy. The final layers used are shown in Figure 4. It consists of one hidden layer with 275 units and a dropout of 20%. The GRU is then trained from the dataset provided until the training has been run sufficiently.

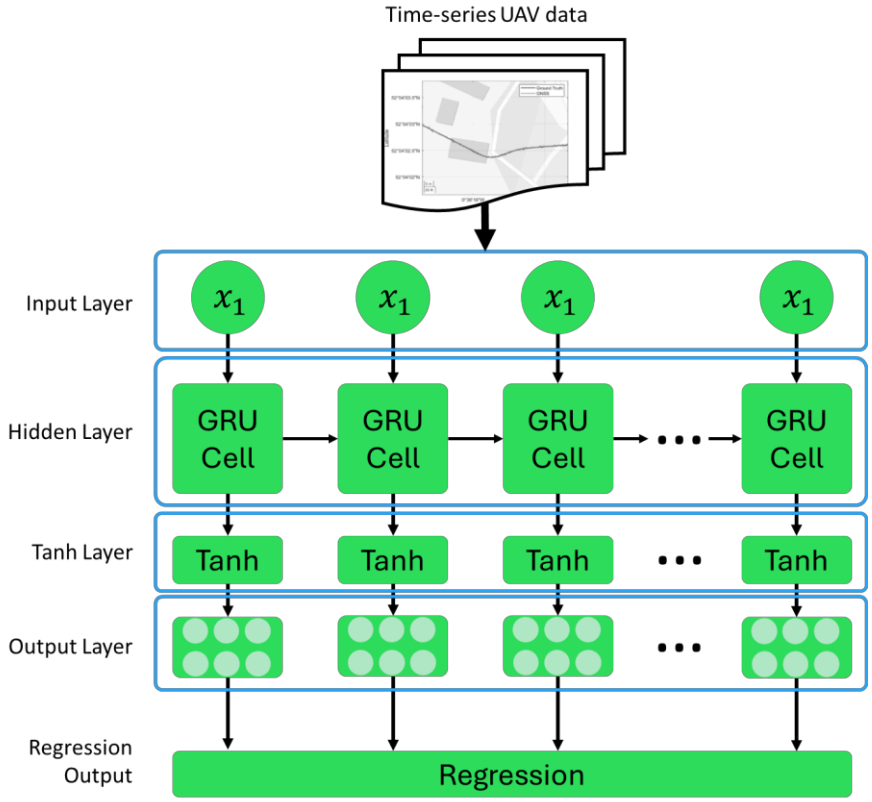

Figure 4. Machine learning layers configuration

VI. Results

A. Data collection

Data collection is carried out using Spirent's SimGEN and SimSENSOR software. In SimGEN, a route is selected that will test the fusion approach's ability by including typical UAV maneuvers such as large straight sections and turning points. The data collected represents the raw GNSS readings consisting of satellite position, velocity, azimuth, elevation, pseudo ranges and rates, delays due to the transition of signal from satellite and receiver, and doppler shift. For the IMU, this consists of accelerometer and gyro readings (angular velocity in all axes). The duration of the data recorded is 8200 seconds.

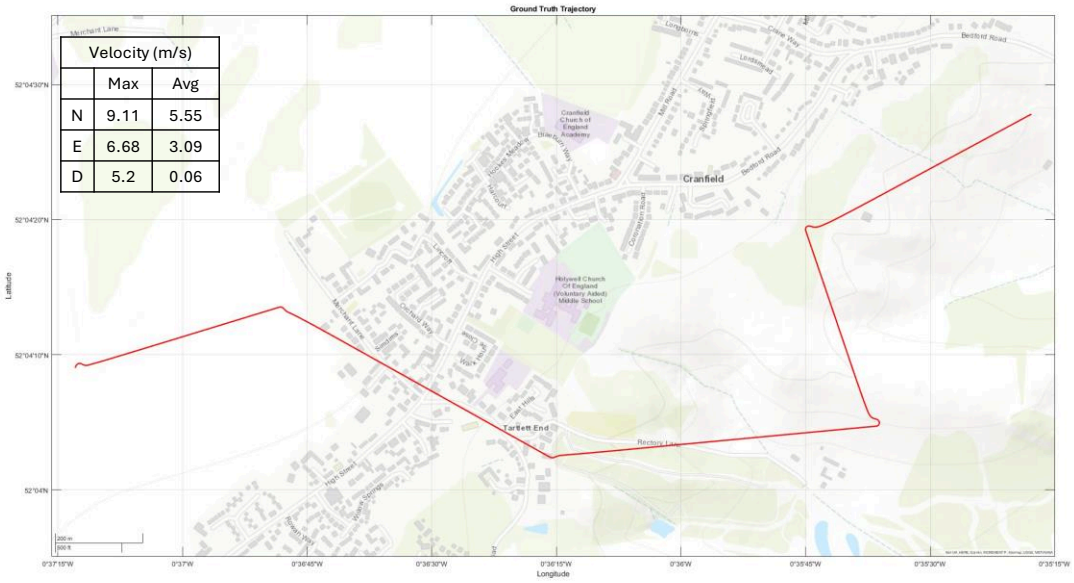

Figure 5. Drone testing trajectory with velocity information

The route trajectory is generated using SimROUTE [23] which is then loaded into SimGEN. The route is shown in Figure 5. Thlimitsed limit pre-defined so that the simulator does not exceed realisticwhilstds wilst the acceleration for each section and curve is random with non-linear acceleration/deceleration ranging between 0 to 2 m/s^2 . Once the maximum speed is reached, the UAV will hold this speed until it reaches close to a curve, where the UAV then starts to decelerate to the defined turning speed. No environmental elements (such as wind or rain) are modelled in the simulator to isolate position and velocity estimation errors for analysis. To generate realistic IMU readings, SimSENSOR is used to simulate IMU readings. The IMU is modeled from a BMI055 IMU [25] developed by Bosch which is used in the popular Pixhawk 4. No additional noises, other than the receiver/sensor specification are applied to SimSENSOR. To provide a realistic reading, sensor deterministic and stochastic errors from a real sensor are used. For GNSS, the NEO-M8 by ublox is used as a model receiver [24] with the simulated errors shown in Figure 6. The key sensor specifications for the gyroscope and the accelerometer are shown in Table 1. In Table 2, the GNSS receiver specification is presented. The dataset is split into 80% training and 20% testing data sequentially to evaluate GRU performance. Mean squared error (MSE) is used as a benchmark to determine the proposed architecture performance whilst training. Once the training is complete, testing data is used to evaluate the performance of the system. The testing results are used to compare with existing methods like Extended Kalman Filters and Recurrent Neural Networks to understand its performance improvements over the traditional methods.

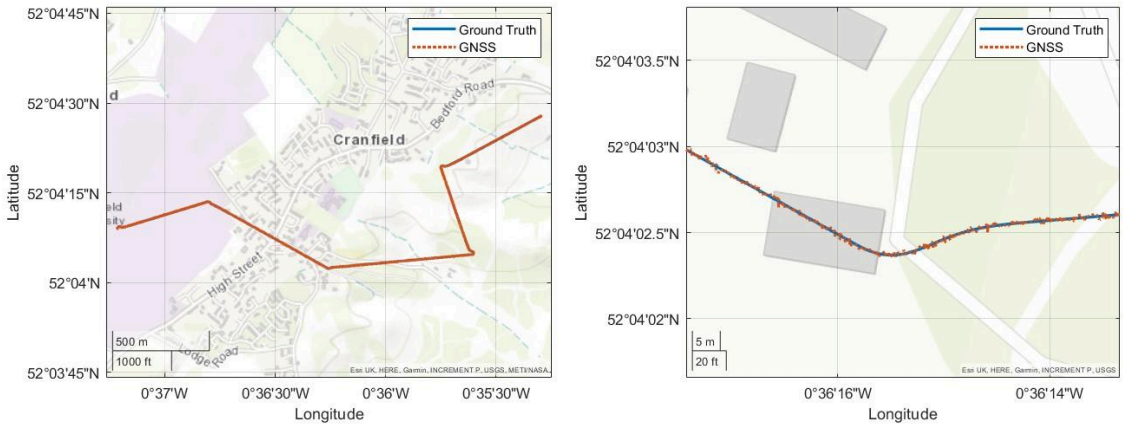

Figure 6. Comparison of drone trajectory with GNSS output from simulator a) whole map b) zoomed in

INS sensor specification			
Accelerometer		Gyroscope	
Scaling factor (ppm)	500	Scaling factor (ppm)	500
Bias (mg)	0.1	Bias (deg/h)	0.001
Accelerometer Random Wallk (ARW) (m/s/sqrt(h))	0.003	Gyroscope Random Wallk GRW (deg/sqrt(h))	0.003
Update rate (Hz)	100		

Table 1. INS sensor specification

GNSS receiver specification	
Pseudo range accuracy (m)	3
Pseudo range rate accuracy (m/s)	0.5
Update rate (Hz)	10

Table 2. GNSS receiver specification

B. Performance Evaluation

In Figure 7, the true position error is compared to the predicted position error in the North (N), East (E), and Down (D) directions. Unlike ground-based vehicles, UAVs depend on obtaining information in all axes to maintain safe flying conditions. It can be shown that INS errors that accumulate due to integration cause the errors to increase rapidly. Whenever GNSS signal is provided again (randomly between 1 to 20 seconds) a sharp correction is observed. This trend continues throughout the journey of the drone in all axes. Comparing the true error to the predicted error, visually it can be seen that most of the errors generated by the INS are predicted by the architecture. There are some instances where this is not the case. These correlate with turning maneuvers that can be seen in Figure 8.

In Figure 8, the ground truth is compared to the INS data and the predicted ground truth. The predicted truth is calculated by taking the INS measurement and subtracting the predicted INS error. The majority of the errors generated by the IMU sensor that led to the predicted path deviating from its actual trajectory due to its deterministic and stochastic errors are mitigated by the GRU architecture. This is especially true for the straight segments of the trajectory. In more challenging circumstances like right or left turns this is also true. However, there are still some rotations that the GRU struggles with predicting the measured errors. This may indicate that more training data is required to improve the performance of the GRU. This can be solved by training the GRU on multiple routes for a prolonged time. Furthermore, in certain circumstances, the architecture overcompensates for the INS reading errors for the position. However, these overcorrections are very minor compared to the errors from the INS data.

In Table 3, a comparison between the Root Mean Square Error performance (RMSE) of the GRU, RNN, and EKF is made. The total position RMSE (the summation of RMSE in the North, East, and Down direction) shows a reduction of nearly 80% for RNN compared to EKF. This is to be expected as the linearization step in EKFs provides estimation errors. Furthermore, assuming Gaussian error distribution may also provide problematic for non-linear problems. RNN also has some short-term memory ability which allows it to use past information to correct the position error. Additionally, looking at the difference between RNN and GRU, we see a further drop in the total position RMSE by 60%. This could be attributed to the GRUs ability to determine whether it keeps or removes the prior information based on their impact on the overall system thus solving the issues related to the gradient loss function observed on RNNs, therefore, allowing GRU to keep more of the useful information on the system.

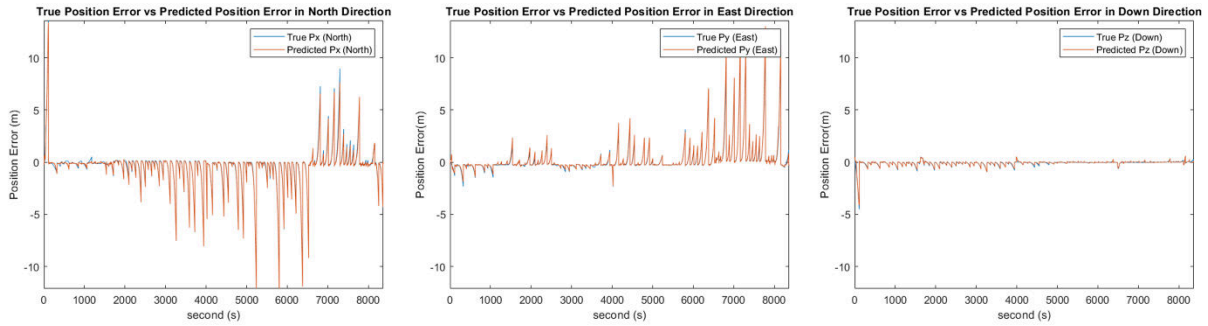

Figure 7. True position error vs predicted error for a) North b) East c) Down

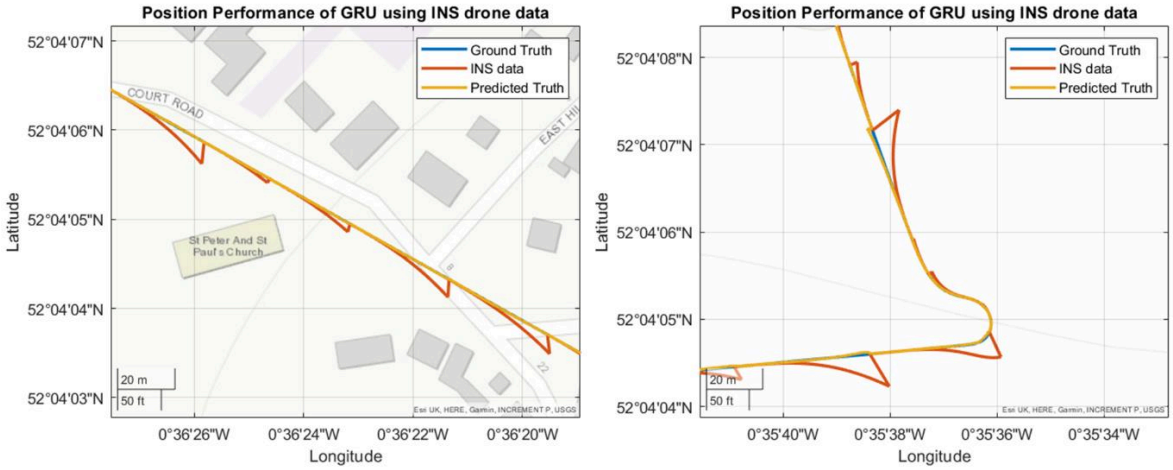

Figure 8. Position performance comparison between Ground Truth, INS data, and Predicted Truth from GRU a) straight segment b) turning point

	GRU errors		RNN errors		EKF errors	
	Position RMSE, (m)	Velocity RMSE, (m)	Position RMSE, (m)	Velocity RMSE, (m)	Position RMSE, (m)	Velocity RMSE, (m)
North	**0.91**	**0.05**	1.95	0.07	6.4	0.45
East	**0.23**	**0.04**	0.53	0.04	5.5	0.2
Down	**0.14**	**0.02**	0.51	0.05	2.2	0.46
Total	**1.27**	**1.1**	2.99	0.15	14.1	1.22

Table 3. RMSE comparison of different fusion architectures

In Figure 9, a comparison of positioning performance is shown between RNN and GRU during a turning point. The proposed architecture reduces the INS errors during GNSS outages whilst RNN struggles with highly non-linear issues. This may be due to the inherent problems RNN has with keeping information for a longer period as compared to GRUs which solve these issues by using update and reset gates to determine which information to keep or discard

therefore reducing long-term errors. However, RNN is faster to train as it does not use gates. Nonetheless, for UAVs higher accuracy is preferred.

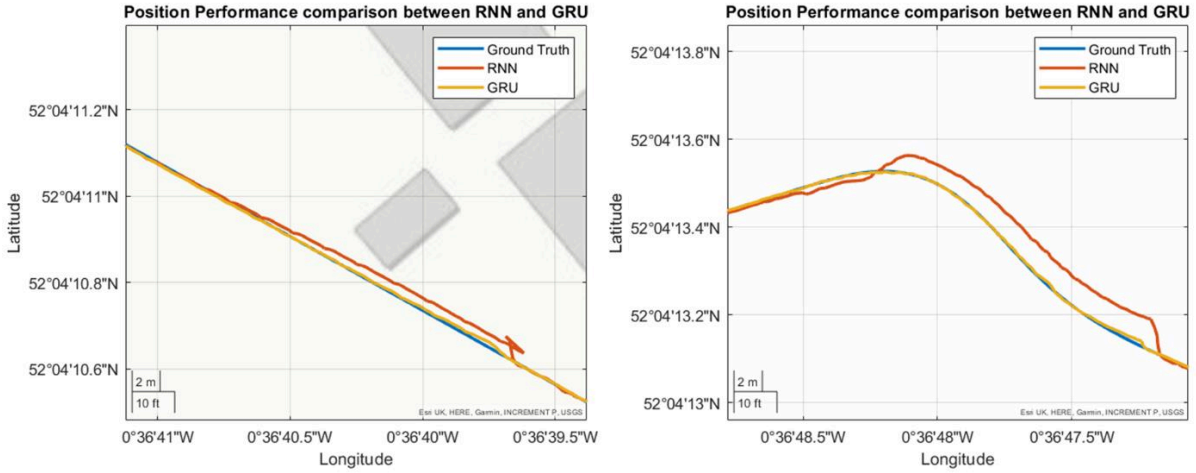

Figure 9. Position performance comparison between RNN and GRU a) straight segment b) turning point

Figure 10 compares the Mean, Standard deviation, and 95th percentile between INS, RNN, and GRU in the horizontal direction. Consistently, GRU outperforms both RNN and GRU in all categories in both the North and East Direction. As expected, RNN and GRU perform significantly better than INS as they can utilize past information. The difference between RNN and GRU is more significant in the North direction as compared to the East direction.

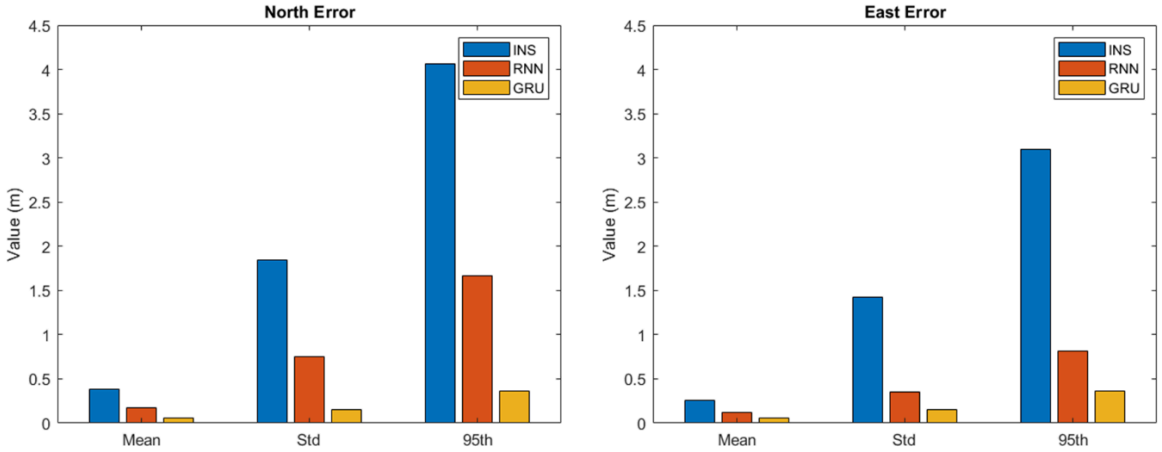

Figure 10. Mean, Standard Deviation and 95th percentile error comparison between INS, RNN, and GRU in a) North direction b) East direction

Looking at Figure 11, an error distribution is presented showing the magnitude of position error of the INS compared with the error distribution when using the proposed method. Comparing the graph, the proposed method reduces the position estimation error caused by the stochastic and deterministic errors exhibited by the accelerometer and gyroscope. After using the GRU, Figure 10 shows that the absolute error at the 95th percentile is less than 0.4 meters This is compared to before using the proposed architecture where the error is about 4 meters at the 95th percentile. Therefore, for GNSS outages lasting 20 seconds, it is reasonable to assume that the position error in INS-only mode using the proposed method will, for most of the time, be less than 0.4 meters.

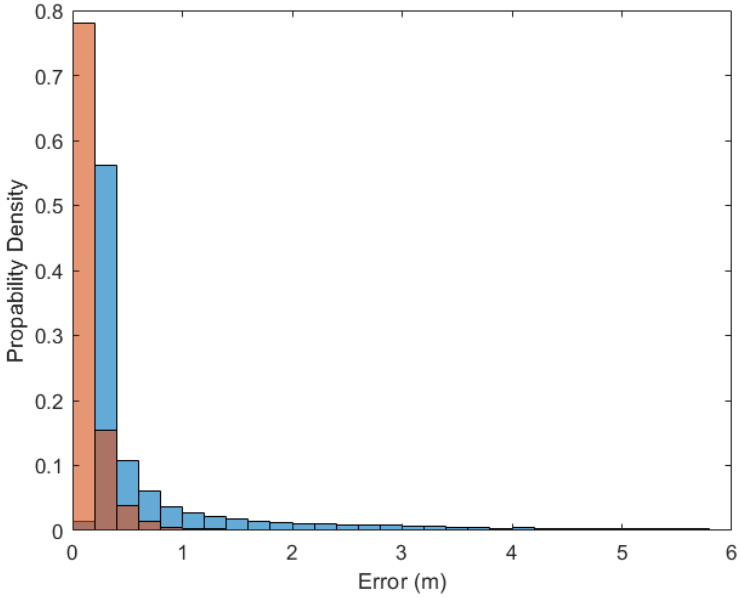

Figure 11. Error distribution comparing INS and GRU performance

VII.Conclusion

The paper aimed to propose a method that aids in situations where GNSS is denied or unreliable for UAVs. Traditional methods use an IMU/GNSS integration method to improve location accuracy. However, these do not aid in situations when GNSS information is unavailable. Existing techniques such as EKFs are not suitable for highly non-linear problems. RNNs have been proposed due to their ability to use past information to determine future output. However, RNNs suffer from gradient loss functions which only allow the system to retain short-term memory. Therefore, it was proposed to use a GRU architecture that solves the issues with RNNs. Analysis of the results shows visually that the proposed method reduces path deviation when GNSS is not available. Furthermore, a comparison of EKF, RNN, and GRU shows a 60% performance improvement for predicting position errors generated by the IMU. Comparing the mean, standard deviation and 95th percentile between INS and the proposed architecture shows an average of 88% improvement in reducing the range of predicted position errors. The method proposed in this paper shows that using GRUs to establish a relationship between the vehicle dynamics and the predicted errors leads to better position and velocity predicting when comparing this to traditional methods when GNSS is not available. However, GRUs are computationally more expensive than EKF and RNN. Future work will be focused on implementing other sensors to the architecture and testing the system in a real-life case.

References

[1] C. Cristodaro, F. Dovis, G. Falco and M. Pini, "GNSS receiver performance in urban environment: Challenges and test approaches for automotive applications," 2017 International Conference of Electrical and Electronic Technologies for Automotive, 2017, pp. 1-6, doi: 10.23919/EETA.2017.7993222.

[2] Chan TK, Chin CS. Review of Autonomous Intelligent Vehicles for Urban Driving and Parking. Electronics. 2021; 10(9):1021. https://doi.org/10.3390/electronics10091021

[3] D. Tang, D. Lu, B. Cai and J. Wang, "GNSS Localization Propagation Error Estimation Considering Environmental Conditions," 2018 16th International Conference on Intelligent Transportation Systems Telecommunications (ITST), 2018, pp. 1-7, doi: 10.1109/ITST.2018.8566771.

[4] Q. Zhang, X. Niu and C. Shi, "Impact Assessment of Various IMU Error Sources on the Relative Accuracy of the GNSS/INS Systems," in IEEE Sensors Journal, vol. 20, no. 9, pp. 5026-5038, 1 May1, 2020, doi: 10.1109/JSEN.2020.2966379.

[5] P. J. Hargrave, "A tutorial introduction to Kalman filtering," IEE Colloquium on Kalman Filters: Introduction, Applications and Future Developments, 1989, pp. 1/1-1/6.

[6] J. Mochnac, S. Marchevsky and P. Kocan, "Bayesian filtering techniques: Kalman and extended Kalman filter basics," 2009 19th International Conference Radioelektronika, 2009, pp. 119-122, doi: 10.1109/RADIOELEK.2009.5158765.
[7] C. Antoniou, M. Ben-Akiva and H. N. Koutsopoulos, "Nonlinear Kalman Filtering Algorithms for On-Line Calibration of Dynamic Traffic Assignment Models," in IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 4, pp. 661-670, Dec. 2007, doi: 10.1109/TITS.2007.908569.
[8] Y. Ban, X. Niu, T. Zhang, Q. Zhang, W. Guo and H. Zhang, "Low-end MEMS IMU can contribute in GPS/INS deep integration," 2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014, 2014, pp. 746-752, doi: 10.1109/PLANS.2014.6851440.
[9] R. Sharaf, A. Noureldin, A. Osman and N. El-Sheimy, "Online INS/GPS integration with a radial basis function neural network," in IEEE Aerospace and Electronic Systems Magazine, vol. 20, no. 3, pp. 8-14, March 2005, doi: 10.1109/MAES.2005.1412121.
[10] A. Noureldin, R. Sharaf, A. Osman and N. El-Sheimy, "INS/GPS data fusion technique utilizing radial basis functions neural networks," PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No.04CH37556), 2004, pp. 280-284, doi: 10.1109/PLANS.2004.1309006.
[11] Noureldin, A., El-Shafie, A. and Bayoumi, M., 2011. GPS/INS integration utilizing dynamic neural networks for vehicular navigation. Information Fusion, 12(1), pp.48-57.
[12] C. H. Chen and L. Yu, "A learning algorithm for improved recurrent neural networks," Proceedings of International Conference on Neural Networks (ICNN'97), 1997, pp. 2198-2202 vol.4, doi: 10.1109/ICNN.1997.614249.
[13] S. Hochreiter and J. Schmidhube, “Long Short-Term Memory,” Neural Comput 1997; 9 (8): 1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735
[14] K. Cho, B. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724-1734, doi: 10.3115/v1/D14-1179
[15] Y. Li and X. Xu, "The Application of EKF and UKF to the SINS/GPS Integrated Navigation Systems," 2010 2nd International Conference on Information Engineering and Computer Science, 2010, pp. 1-5, doi: 10.1109/ICIECS.2010.5678253.
[16] K. Lee, A. Oka, E. Pollakis and L. Lampe, "A comparison between Unscented Kalman Filtering and particle filtering for RSSI-based tracking," 2010 7th Workshop on Positioning, Navigation and Communication, 2010, pp. 157-163, doi: 10.1109/WPNC.2010.5650817.
[17] D. Rumelhart, G. Hinton, and R. Williams, “Learning representations by back-propagating errors.” Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0
[18] M. Jordan, “Chapter 25 - Serial Order: A Parallel Distributed Processing Approach,” in Neural-Network Models of Cognition, vol. 121, J. W.Donahoe and V.Packard Dorsel, Eds.North-Holland, 1997, pp. 471–495.
[19] J.Chung, C.Gulcehre, K.Cho, andY.Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv Prepr. arXiv1412.3555, 2014
[20] H. fa Dai, H. wei Bian, R. ying Wang, and H. Ma, “An INS/GNSS integrated navigation in GNSS denied environment using recurrent neural network,” *Defence Technology*, vol. 16, no. 2, pp. 334–340, Apr. 2020, doi: 10.1016/j.dt.2019.08.011.
[21] H. Kim, T. Bae, “Deep Learning-Based GNSS Network-Based Real-Time Kinematic Improvement for Autonomous Ground Vehicle Navigation,”. J. Sens. 2019, pp. 1–8.
[22] p. Groves, “Principles of GNSS, inertial, and multisensor integrated navigation systems,” in GNSS technology and applications series. Boston: Artech House;2008. pp. 518.
[23] "GNSS Simulation: SimROUTE™ Software Option - route matched trajectory generation and motion visualization", Lange-electronic.com, 2021. [Online]. Available: https://www.lange-electronic.com/en/products/gnss-simulation/simroute-software-option-routen-angepasste-trajektoriengenerierung-und-bewegungsvisualisierung-detail.
[24] "GNSS Simulation: SimROUTE™ Software Option - route matched trajectory generation and motion visualization", Lange-electronic.com, 2021. [Online]. Available: https://www.lange-electronic.com/en/products/gnss-simulation/simroute-software-option-routen-angepasste-trajektoriengenerierung-und-bewegungsvisualisierung-detail.
[25] Cdn.sparkfun.com, 2021. Available: https://cdn.sparkfun.com/assets/d/d/9/9/3/Pixhawk4-DataSheet.pdf.