# Enhancing Navigation Systems with Recurrent Neural Network-Enhanced Kalman Filters

Alex Hostick
Dept. of Electrical and Computer Engineering
University of New Mexico
Albuquerque, New Mexico
ahostick@unm.edu

***Abstract*—GNSS jamming and spoofing have become commonplace worldwide in electronic warfare. Conflicts in the last decade have employed electronic attacks to disrupt, deny, and employ cyber attacks against innovative weaponry and drone warfare. Domestically, technology for amateur use of systems disrupts GPS in the magnitude of hours to days, which is becoming cheaper and widely available. Kalman filters are integral to navigation sensor fusion to hold position information when navigation signals are degraded or unavailable. These filter algorithms provide the mathematical framework for the recursive estimation of state variables. They are widely employed in various industries and engineering projects, giving optimal real-time estimates for systems with noisy measurements and dynamic changes. However, low fidelity, older technology, and less costly GNSS and INS devices are prone to significant drift and noise due to clock errors, bias drift, temperature-induced variations, and accumulated errors from unmodeled system dynamics. This paper investigates the use of Recurrent Neural Networks to enhance a discrete Kalman filter, addressing the limitations in the standard Markovian approach of state predictions. Research into RNN-augmented Kalman filters improves state estimation accuracy, characterizes and subverts drift, and enhances low-fidelity navigation system's robustness.**

***Index Terms*-Kalman Filters, machine learning, RNN, INS, navigation, linear approach, quaternion, complexity, MEMS**

## I. Introduction

In 2021, at 158.8 billion market shares, the installed base of global navigation satellite system (GNSS) devices worldwide stood at 6.5 billion units, with forecasts predicting that this is to rise to 10.6 billion devices by 2031 [1]. In 2024, nearly 1000 pilots near the Black Sea and Cyprus are reporting their GPS either jammed or spoofed as they fly near conflicts in Israel and Ukraine [2]. While the United States is moving towards more GPS-resilient signals for electronic warfare attacks, the satellite architecture is slow to develop. Drone warfare has driven the requirement for jamming and spoofing systems, with Russia, Iran, and China reportedly normalizing the use of weaponized GNSS jamming [3].

As jamming technology becomes available and low cost to state and non-state actors, more robust sensor fusion tactics are required to maintain attitude and direction during periods where GNSS is unavailable. Dallas Airport reported a jamming incident that lasted 44 hours in 2022 [4]. The same year, DHS reported that Denver Airport saw similar jamming for 33 hours [5]. Long periods and widespread jamming affect civilian GPS systems that do not have military-grade protection. Vehicles and aircraft have difficulty keeping their receivers locked on a signal strength that averages -130dBm or $10^{-16}$ Watts [6] in tunnels and mountainous regions. Inertial navigation systems (INS), used to keep track during short periods of GNSS outage, are sensor-fusion technology. INS systems for military-grade systems can last without GPS for several hours, while industrial grades are typically less than a minute [7].

Kalman filters, named after Rudolf E. Kalman, indirectly estimate states using measurements observed over time and produce statistically predicted state estimates. Sensors provide the Kalman filter with enough data to estimate the system's state by combining different sources that may be subject to noise but become more accurate by estimating a joint probability distribution over the variables every time step [8]. The filter is constructed as a mean squared error minimizer, but an alternative filter derivation is also provided, showing how the filter relates to maximum likelihood statistics [9]. Large industries Kalman filters are employed include:

- Navigation and Guidance
- Robotics and Industrial Controls
- Signal Processing
- Finance, Healthcare, and Biomedical Engineering
- Communications and Networks

However, excess sensor noise and drift accumulating over time create long-term deviations from actual measurements. How can we solve this problem when size, weight, price, and cost (SWAP-C) are required for an application?

A Recurrent Neural Network (RNN) leverages its inherent properties to augment the biases that accumulate in sensors the Kalman filter relies upon. RNNs are machine-learning algorithms utilizing a neural network to learn temporal or sequential data patterns. RNN applications are unique to other feedforward neural network applications as they have

connections, or "memory," that loop back on themselves, allowing information to persist over time [10]. Like Kalman filters, RNNs use a recurrent process of capturing information about previous inputs. However, RNNs capitalize backpropagation through time (BPTT) to learn patterns in sequential data [11]. RNNs blend their use cases with Kalman filters and are employed in industries that use time-series analysis, including:

- Natural Language Processing
- Anomaly Detection
- Speech Processing
- IoT, Wearables
- Music and Audio

To meet reliability of system measurements, RNNs can be employed to learn sensor drift accumulation and Gaussian system noise and detect system anomalies to augment the Kalman filter process for more robust predictions. Utilizing an RNN-enhanced network can assist with navigation applications, where the quality of GPS signals and sensor biases can affect the overall position accuracy.

Historically, navigation has relied on the Global Navigation Satellite Systems (GNSS). Augmented or enhanced navigation can utilize inertial navigation systems (INS) mixed with dead reckoning, Wi-Fi and cellular positioning, radio, and an assortment of sensor-fusion for improved accuracy. More sources of known, accurate systems and structures and real-time kinematics, mixed with the geometry of spaced-based signals, provide centimeter accuracy when blended [12] for faster-moving systems like cars, drones, and aircraft.

When satellite signals are weak in urban environments, degraded by multipath effects, or outright jammed, navigation systems blend Kalman filters with sensor measurements to keep an accurate position. Not all systems have every sensor available, and rural locations may not have cell tower reliability. Utilizing the RNN-Kalman enhanced approach will create greater longevity of sensor accuracy until GNSS is available instead of a standard Kalman filter approach.

## II. Kalman Filters

Kalman filter keeps track of the estimated state of the system and the variance of the estimates made. The process includes two main steps: prediction and update, weighted by Kalman gain. The process starts with an initialization when no prior state knowledge is known, or $x_k$ and initial uncertainty $P_k$. After initialization, the filter uses the previous state transition model $\hat{x}_{k|k-1}$ and the previous prediction $P_{k|k-1}$, combined with sensor measurements to forecast the system's current state.

Sensor measurements are combined into the prediction using the Kalman gain, or a weighted average, to adjust the influence of predictions based on their respective uncertainties. The recursive process ensures more reliable estimates are given higher weight. Figure 1 denotes the process of the Kalman filter. One can use this process loop for the Kalman filter and break it down into its sequences, understanding where improvements are made to reduce the impact of sensor biases and noise, which impact the final Kalman filter prediction.

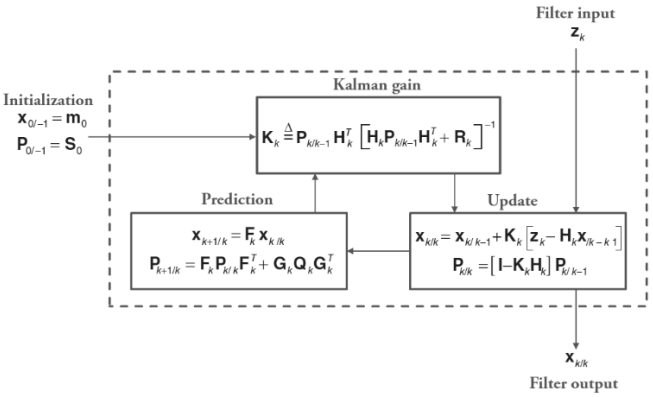


**Fig. 1.** Kalman Filter Process [12]

### A. Kalman Gain

Higher Kalman gain indicates that measurements are trusted, while low Kalman gain will note higher levels of uncertainty in measurements, typically due to sensor or process noise. The output of the Kalman gain determines how much weight is given to the new measurement compared to the prediction with (1):

$$K_k \triangleq P_{k|k-1}H_k^{\top}\left[H_kP_{k|k-1}H_k^{\top} + R_k\right]^{-1} \tag{1}$$

where $H_k$ is the measurement model from the state to the measurement, $R_k$ is the measured noise covariance for measurement uncertainty and $P_{k|k-1}$ is the predicted state covariance from the previous iteration.

### B. Update Step

The update step ($x_{k|k}$) will utilize the new measurements $z_k$ to predict a new state. Measurements are used to correct predicted states with (2)

$$x_{k|k} = x_{k|k-1} + K_k\left[z_k - H_kx_{k|k-1}\right] \tag{2}$$

with the term $[z_k - H_kx_{k|k-1}]$ as the measurement residual, or "innovation," between the predicted and actual measurements. The update step includes the update to the covariance ($P_{k|k}$) for uncertainty after measurements are incorporated. The updated covariance is defined by (3) while using the identity matrix ($I$) for proper dimensionality.

$$P_{k|k} = [I - K_kH_k]P_{k|k-1} \tag{3}$$

### C. Prediction Step

The prediction state $\left(x_{k+1|k}\right)$ uses the system's model with the state transition model matrix $F_k$ and the process noise

covariance matrix $Q_k$ for the next prediction ($P_{k|k+1}$). Updates from the measurements are used in the predicted state with (4):

$$x_{k+1|k} = F_k x_k \tag{4}$$

and the next predicted covariance is calculated with (5):

$$P_{k+1|k} = F_k P_k F_k^{\top} + G_k Q_k G_k^{\top} \tag{5}$$

where $G_k$ is the process noise model, leading to $G_k Q_k G_k^{\top}$ accounting to how process noise affects the state uncertainty and $F_k P_k F_k^{\top}$ updates the covariance matrix based on state transition dynamics for the system's inherent behavior and the existing uncertainty.

After each interval, the filter will use the update step and the prediction step iteratively as new measurements are received. In this research, the innovation vector of device measurements $z_k$ where measurement model $H_k$ will be reduced by an RNN augmentation for the measurements and predictions. GPS receiver measurements, which contribute to the noise covariance for measurement uncertainty $R_k$. The variability in process noise covariance $Q_k$ will also be dynamically updated with assistance from the RNN. Finally, the drift patterns learned by the RNN in process dynamics $x_{k+1|k}$, which accumulate from several sources, including biases from slowly varying offsets in accelerometer and gyroscope, temperature-induced fluctuations, small errors in acceleration or angular velocity, and sensitivity to calibration for the dynamic models.

## III. Recurrent Neural Networks, LSTM, and GRU

In 1943, neurophysiologists Warren McColloch and mathematician Walter Pitts wrote a paper on how neurons might work using a simple neural network using electrical circuits [13]. In 1960, the concept of "close-loop cross-coupled perceptrons," which use the recurrent connection, was published by Frank Rosenblatt [14]. The theory, application, and statistical mechanics of RNNs related to the mid-1900s research developed the fundamental aspects of RNN variants used today.

RNNs have recurrent loops that allow information to persist. The recurrent aspect of RNNs differs in its applications from Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN). ANNs are the simplest form of neural networks, typically used in simple regression tasks and pattern recognition. At the same time, CNNs are specialized neural networks for processing grid data-like images to detect features [14]. An RNN can utilize its hidden state to represent past knowledge. The hidden state from the previous step, together with the input of the current step, is used to derive the current hidden state and then to derive the current step's output [15]. The recurrent nature is noted in Fig. 2.

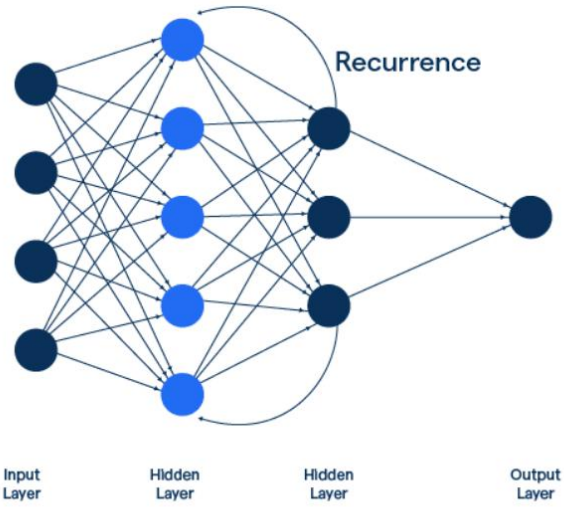


Fig. 2 General Recurrent Neural Network Structure [16]

Each neural network sequence can be noted in Fig. 3, where the BPTT and series of time steps, weights, and hidden inputs are noted. The variable description is available in Table I.

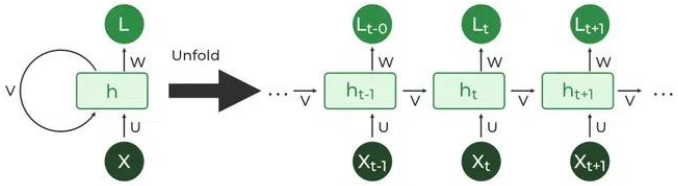


Fig. 3. Unfolded Recurrent Neural Network [17]

Table I Recurrent Neural Network BPTT Variables

| Variable | | Description |
|---|---|---|
| $X_t$ | $X_t$-1 | Input data from previous step |
| | $X_t$ | Current input data |
| | $X_t$+1 | Next input data parameter |
| $h_t$ | $h_t$-1 | Hidden state from previous step |
| | $h_t$ | Hidden state at time t, computed on input $X_t$ and $h_t$-1 |
| | $h_t$+1 | Hidden state at time t+1 |
| $L_t$ | $L_t$-0 | Output layer/vector |
| | $L_t$ | Derived from hidden state $h_t$ |
| | $L_t$+1 | RNN prediction or output at time t |
| U, V, W | | Weight matrices for the RNN transforms |

The process of Fig. 3 can also be described mathematically by the function of the current state in the abstract form (6):

$$h_t = f(h_{t-1}, X_t) \quad (6)$$

with a non-linear activation function (7):

$$h_t = \tanh\left(W_{hh}h_{t-1} + W_{xh}x_t\right) \quad (7)$$

and the output state in (8):

$$L_t = W_{hL}h_t \quad (8)$$

### A. Long Short-Term Memory

An inherent problem of RNNs is the vanishing and exploding gradient. The vanishing gradient problem in RNNs occurs because the gradients are propagated backward through time, and they can become very small due to the repeated multiplication of gradients in each step [18]. This leaves the network unable to learn from past iterations and thus ignore long-term dependencies. If the gradients explode, the network becomes unstable and sensitive to small changes in the input, which can lead to numerical overflow, erratic behavior, and overfitting [19].

### B. Memory Cell State

To overcome vanishing and exploding gradients in legacy RNNs, the process Long Short-Term Memory (LSTM) method was published in 1997 [20] by Sepp Hochreiter and Jürgen Schmidhuber, introducing memory cells and gates. An LSTM memory cell stores information over long periods of sequential data. The memory cell state $c_t$ store the long-term LSTM information [21] and is updated by (9):

$$C_t = (f_t \cdot C_{t-1}) + (i_t \cdot \tilde{C}_t) \quad (9)$$

where $(f_t \cdot C_{t-1})$ is the forgotten portion of the previous cell state and $(i_t \cdot \widetilde{c_t})$ is the new memory from the input gate. Fig. 4 denotes the core aspects of the LSTM memory cell, noting the internal Forget, Input, and Output gates.

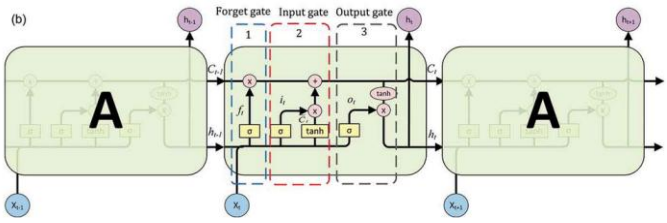


Fig. 4 LSTM Memory Cell Overview with Gates [22]

### C. Forget Gate

LSTM gates can control which information can be added and which can be discarded from the current time series information. The Forget gate reads $x_t$ and $h_{t-1}$, and outputs a value between 0 and 1 from the sigmoid function in (10). A 1 will indicate keeping the information, and 0 will discard it.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (10)$$

where $f_t$ is the forget gate output between 0 and 1, $\sigma$ is the sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$, $W_f$ is the weight matrix, $[h_{t-1}, x_t]$ is the concatenation of the hidden state $h_{t-1}$ and the current input $x_t$, and $b_f$ is the bias vector that allows the model to adjust output.

### D. Input Gate

LSTM input gates $i_t$ consists of a sigmoid layer and a tanh layer, deciding what new information will be stored in the cell state $C_t$. The input gate uses the current input $x_t$ and the previous hidden state $h_{t-1}$ to ensure the LSTM updates with relevant new information after the Forget gate clears irrelevant information. Gate activation $i_t$ is noted in (11) and outputs a value between 0 and 1 for determining the weight of the new information.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (11)$$

From Fig. 4, we also see the tanh layer, which creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state and is denoted in (12)

$$\widetilde{C_t} = tanh(W_C[h_{t-1}, x_t] + b_C) \quad (12)$$

where the tanh function is a hyperbolic tangent activation function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, squashing the values to the range of -1 and 1. The Forget Gate and Input gate outputs are used in (9) to control how much information is used to update the LSTM's memory.

### E. Output Gate

Lastly, the Output gate protects subsequent cells by only passing the relevant parts of the updated cell sent forward as the hidden state $h_t$. The hidden state passes information to the next time step and is the output of the memory cell. A sigmoid function (13) decides which value from the Input gate to pass, and the tanh decides the level of importance multiplied with output of the sigmoid for the hidden state noted in (14).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (13)$$

$$h_t = o_t * tanh(C_t) \quad (14)$$

Only passing the relevant information forward prevents unnecessary information from propagating to the next cell, mitigating and controlling the values of the hidden state. By passing information through the gates, the weighted values ensure vanishing nor exploding gradients heavily influence the training process.

Insert Table II here

### *F. Grated Recurrent Unit and LSTM*

A Grated Recurrent Unit (GRU) is a simplified variant of the LSTM. GRU architecture solves the vanishing gradient problem and lowers the overall complexity compared to RNN and LSTM while retaining long-term memory without removing irrelevant information [23]. The GRU uses the update and reset gates, as shown in Fig. 5.

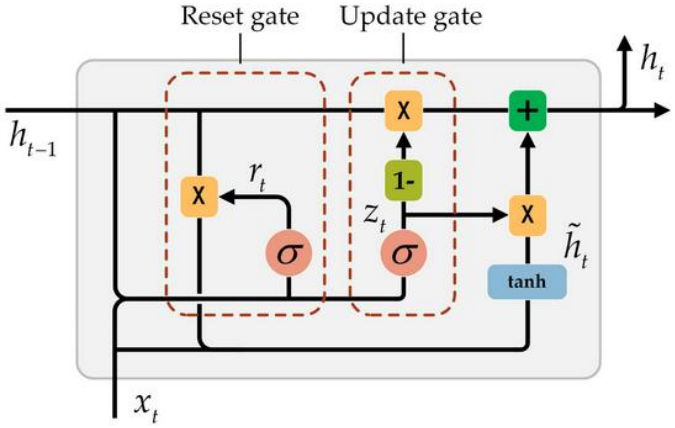


Fig. 5 High-Level Grated Recurrent Unit Structure [24]

The premise of the GRU RNN is that it uses gating mechanisms to selectively update the network's hidden state at each time step. The gating mechanisms are used to control the flow of information in and out of the network [25]. By doing so, the update gate combines LSTM's Forget and Input gates. The GRU network uses a hidden state $h_t$ instead of a cell state $C_t$. GRU theory is generalized in Table II, utilizing the same methods as LSTM RNNs for 0 and 1 outputs from the sigmoid function and -1 and 1 outputs from the tanh activation function.

## IV. RNN-Enhanced Kalman Filters and Navigation Sensor Fusion Setup

For the research, three Kalman filters with 41 vectors were created with Python. The three filters were a 'pure' filter from the filter.py library, a filter.py filter with vanilla RNN enhancement from PyTorch, and a filter with an RNN variant combination of LSTM and GRU. Sensor measurements included in the filter are the output of an LSM9DS1 INS module integrated into an Arduino Nano 33 BLE Sense and a u-blox EVK-F9T GNSS receiver. The GNSS receiver and ELM327 were the control group for research to compare the standard performance of the open-source filter.py library versus the same Kalman filters with RNN enhancements. RNN enhancements included a standard RNN trained on 2 hours of collected INS and GPS data, including 1 hour of stationary measurement and 1 hour of motion while driving. The training consists of nearly 720,000 IMU data points and 180,000 GPS readings. The state model for the system is defined in (15).

$$X = \begin{bmatrix} \psi_E & \psi_N & \psi_D \\ \delta V_E & \delta V_N & \delta V_D \\ \delta L & \delta\lambda & \delta h \\ \nabla b_x & \nabla b_y & \nabla b_z \\ \nabla b_{gyro,x} & \nabla b_{gyro,y} & \nabla b_{gyro,z} \\ \varepsilon_x & \varepsilon_y & \varepsilon_z \\ \varepsilon_{gyro,x} & \varepsilon_{gyro,y} & \varepsilon_{gyro,z} \end{bmatrix} \quad (15)$$

where:

- $\psi$ is the attitude orientation and quaternion errors
- $\delta V$ are the velocity North, East, Down (NED) errors
- $\delta L, \delta\lambda, \delta h$ are latitude, longitude, and altitude errors
- $\nabla b_x, \nabla b_y, \nabla b_z$ are accelerometer biases
- $\nabla b_{gyro,x}, \nabla b_{gyro,y}, \nabla b_{gyro,z}$ are gyroscope biases
- $\varepsilon$ are residual IMU noise components

At present, a motion model for the LSM9DS1 is not available and is assumed generically as the discrete state space equation in (16):

$$\begin{cases} X_k = F_{k,k-1}X_{k-1} + W_k \\ Z_k = H_k X_k + V_k \end{cases} \quad (16)$$

where:

- $F_{k,k-1}$ is the state transition matrix
- $W_k, V_k$ are the process noise and measurement noise
- $H_k$ is the measurement matrix
- $Z_k$ is the measurement vector

Time predictions are implemented with the update steps in (6) for the state error estimate and (7) for the covariance error matrix. Measurement updates for the Kalman gain were implemented with (1), the state vector updated values with (2), and the updated covariance matrix with (3). Table III notes the LSM9DS1 [26] and u-blox EVK-F9T [27] sensor specifications.

Table II Sensor Accuracy

| Sensor | Parameter | Accuracy |
|---|---|---|
| LSM9DS1 IMU Rates: Accelerometer: 100Hz Gyroscope:100Hz Barometer:75Hz Magnetometer: 25Hz | Range | ±2 g (accelerometer), ±245 dps (gyroscope) |
| | White Noise (Gyroscope) | (3.3869, 3.2936, 2.7094) × $10^{-3}$ rad/s/√Hz per axis |
| | Temperature Drift | (7.4197, 4.6638, -0.84739) × $10^{-2}$ °/s/K |
| u-blox EVK-F9T GNSS Receiver Sampled at 2Hz | Horizontal Position Accuracy (CEP) | 2.0 m (standalone mode) |
| | Velocity Accuracy | 0.05 m/s |
| | Dynamic Heading Accuracy | 0.3° (at 30 m/s) |
| | Sensitivity | Tracking: -167 dBm, Cold Start: -148 dBm |

Configuration of sensor locations is essential. Gyroscope readings can pick up vibration from the vehicle roof, adding extra noise to the sensor. Though this is a suitable vector for RNN training, the Arduino Nano 33 BLE was placed on top of

the center console of a 2016 Toyota Rav4, and the GPS antenna is placed 1 meter away on the dashboard and inside the vehicle. The configuration matches the setup in Fig. 6. Fig. 7 notes the hardware setup inside the car, with each device connecting via USB to the control CPU. While driving, the primary receiver under test was disconnected, and the Kalman filters do not have navigation updates to their state vectors. A second u-blox F9T receiver references east velocity, north velocity, longitude, latitude, and altitude errors as a 'truth' source. There are inherent offsets in the primary receiver under test and the truth receiver due to clock biases and processing data. A rolling average of position error relative to each receiver is taken before applying the max and RMSE error calculations.

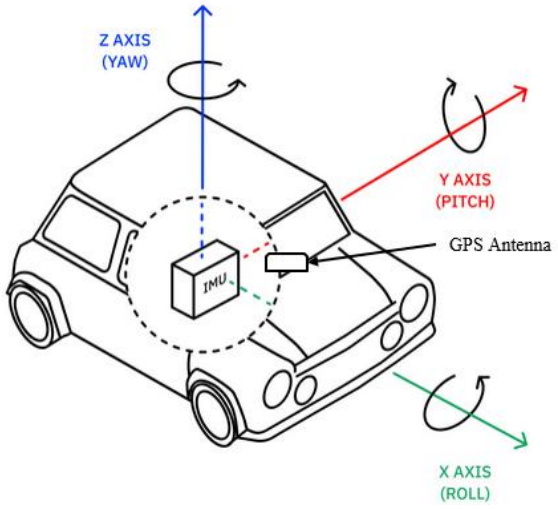


Fig. 6. IMU and GNSS Antenna Vehicle Setup

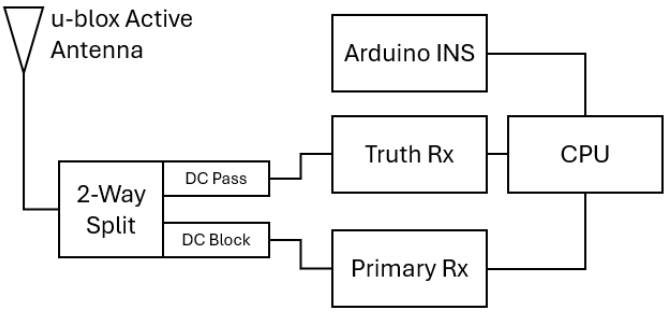


Fig. 7 INS, GPS, and Vehicle Hardware Setup

## V. Performance Results

### A. IMU Corrections

While stationary, the Arduino Nano 33 BLE Sense exhibited significant drift. IMU acceleration in the X direction immediately drifts without external forces applied, as noted in Fig. 8. At the same time, the Y magnitude experienced the same phenomenon in Fig. 9. In both cases, all Kalman filters predicted the stationary state and updated the state vector accordingly during a 20-minute trial. The Hybrid RNN combines LSTM+GRU augmentation, while the standard RNN is a vanilla version from PyTorch's library. The Pure Kalman filter is the standard model from filterpy and utilizes Fig. 1 arithmetic and standard Kalman filter implementation. An adaptive noise model for $G_k$ and a quaternion smoothing algorithm were applied to all Kalman filter models.

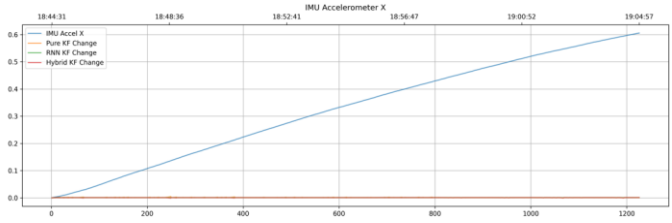


Fig. 8 IMU Accelerometer X Drift

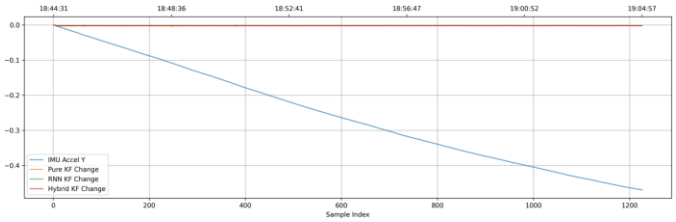


Fig. 9 IMU Accelerometer Y Drift

The z-axis for the accelerometer had significant corrections from the pure, non-RNN augmented predictions. Vibrations and gravity affect minute errors in the z-axis, noting that the vanilla RNN and the Hybrid RNN are less sensitive to anomalies. The performance of the models is noted in Fig. 10.

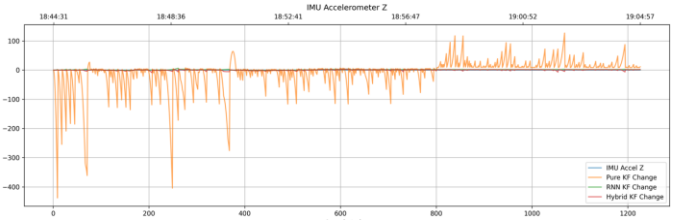


Fig. 10 IMU Accelerometer Z Noise

Surprisingly, all Kalman models were sensitive to gyroscope noise. As noted in Fig. 12, process noise in the gyroscope enacts significant biases on the Kalman filters. The Vanilla RNN the least sensitive to gyroscope noise.

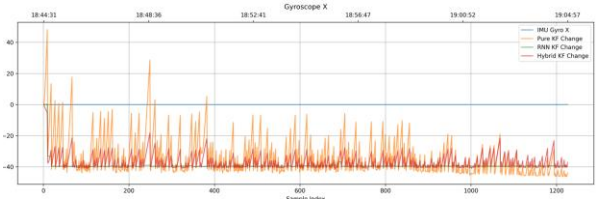


Fig. 11 IMU Gyroscope Noise

### B. *Stationary Tests*

A second trial of disconnecting the primary GPS receiver is conducted. During the stationary test, the velocity and position errors are analyzed during 60 and 100 seconds of GPS outages. Table III notes the error statistics during 30 seconds of GPS outages, with Fig. 12 noting the east velocity and north velocity errors. Fig. 13 notes the longitude and latitude errors. As indicated in Table III, with respect to north velocity, east velocity, longitude, and latitude, the standard RNN was able to reduce RMSE velocity and position error against the standard Pure Kalman filter model by 75%, 0%, 38.2%, and 94.5%, respectively. The Hybrid LSTM+GRU RNN reduced the errors by 61.5%, 33.3%, 28%, and 68.3%, respectively.

Table III Error Statistics During a 60s Stationary GPS outage

| Error | Pure KF | | RNN KF | | Hybrid KF | |
|---|---|---|---|---|---|---|
| | Max | RMSE | Max | RMSE | Max | RMSE |
| East velocity (m/s) | 0.0013 | 0.0004 | 0.0001 | 0.0001 | 0.0005 | 0.001 |
| North velocity (m/s) | 0.0008 | 0.0003 | 0.0003 | 0.0003 | 0.0004 | 0.0002 |
| Longitude (m) | 76.54 | 26.11 | 23.32 | 16.13 | 34.66 | 18.79 |
| Latitude (m) | 145.52 | 46.82 | 7.90 | 5.68 | 51.65 | 14.85 |

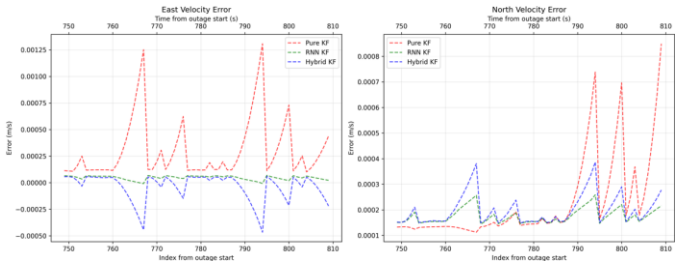


Fig. 12 Velocity Errors during 60s GPS Outage

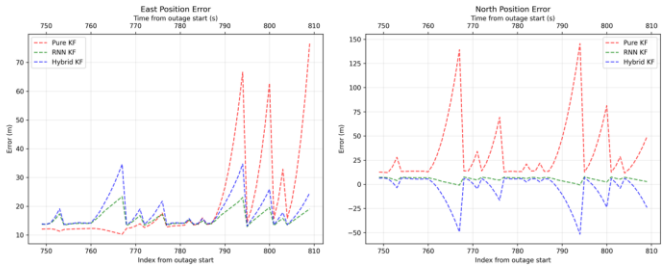


Fig. 13 Position Errors During 60s Outage

An extended test notes how the Kalman filters perform at 100 seconds of GPS outage. As indicated in Table IV, with respect to north velocity, east velocity, longitude, and latitude, the standard RNN was able to reduce RMSE velocity and position error against the standard Pure Kalman filter model by 92.3%, 50%, 54.9%, and 96.2%, respectively. The Hybrid LSTM+GRU RNN reduced the errors by 33.3%, 50%, 45.7%, and 56.7%, respectively.

Table IV Error Statistics During a 100s Stationary GPS Outage Period

| Error | Pure KF | | RNN KF | | Hybrid KF | |
|---|---|---|---|---|---|---|
| | Max | RMSE | Max | RMSE | Max | RMSE |
| East velocity (m/s) | 0.0013 | 0.0003 | 0.0001 | 0.0000 | 0.0006 | 0.0002 |
| North velocity (m/s) | 0.0013 | 0.0004 | 0.0003 | 0.0002 | 0.0005 | 0.0002 |
| Longitude (m) | 117.11 | 36.51 | 26.64 | 16.46 | 40.92 | 19.83 |
| Latitude (m) | 145.53 | 7.90 | 5.55 | 5.55 | 62.90 | 17.04 |

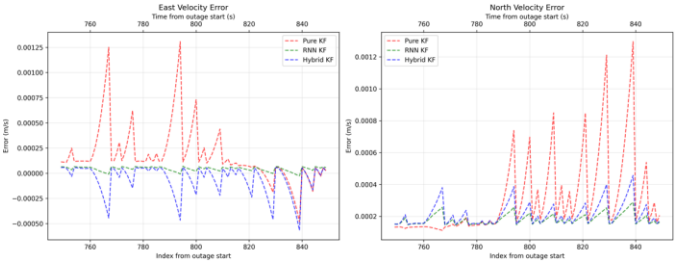


Fig. 14 Velocity Errors during 100s GPS Outage

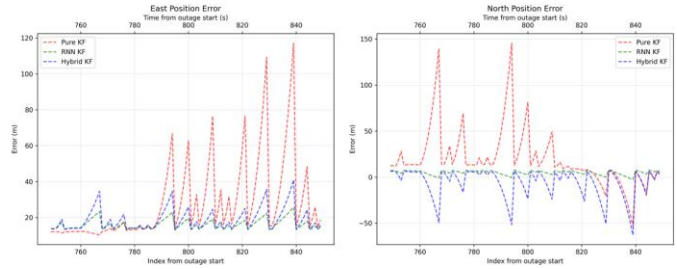


Fig. 15 Position Errors during 100s GPS Outage

### C. *On-Road Testing*

Data is collected during a nearly 35-minute drive. Results from 17 minutes with the primary GPS receiver are tested against the latter 18 minutes of the test with the receiver disconnected from the system. Fig. 16, 17, and 18 denote the latitude, longitude, and altitude position errors, respectively. During the first 30 seconds of GPS not adding to the state vector predictions, Table IV notes that the standard RNN was able to reduce RMSE velocity and position error against the standard Pure Kalman filter model by 0%, 0%, 11.76%, and 0.36%, respectively. The Hybrid LSTM+GRU RNN reduced the errors by 0%, 0%, 10.26%, and 0.48%, respectively.

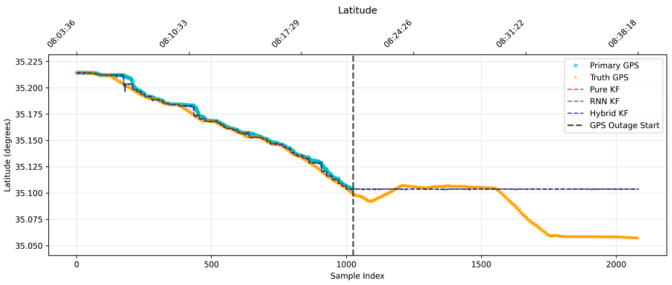


Fig. 16 Latitude Position Errors after GPS Outage

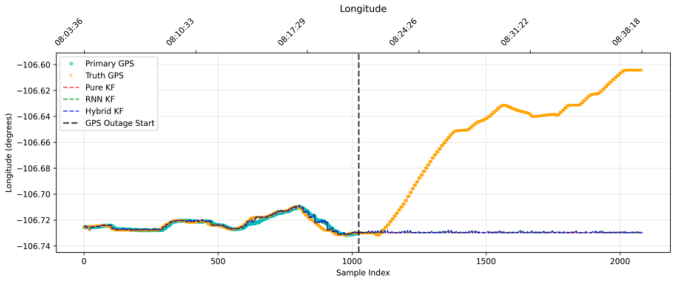


Fig. 17 Longitude Position Errors after GPS Outage

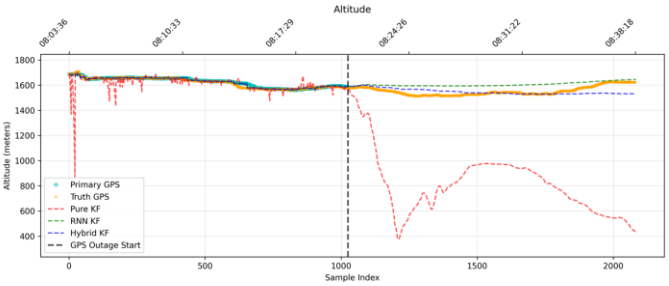


Fig. 18 Altitude Position Errors after GPS Outage

Table V Error Statistics During a 30s Driving GPS Outage

| Error | Pure KF | | RNN KF | | Hybrid KF | |
|---|---|---|---|---|---|---|
| | Max | RMSE | Max | RMSE | Max | RMSE |
| East velocity (m/s) | 0.0068 | 0.0061 | 0.0068 | 0.0061 | 0.0068 | 0.0061 |
| North velocity (m/s) | 0.001 | 0.0004 | 0.001 | 0.0004 | 0.001 | 0.0004 |
| Longitude (m) | 86.98 | 32.63 | 94.16 | 36.47 | 95.09 | 35.98 |
| Latitude (m) | 554.36 | 550.98 | 554.48 | 552.99 | 552.9 | 548.29 |

## *A. 30 seconds*

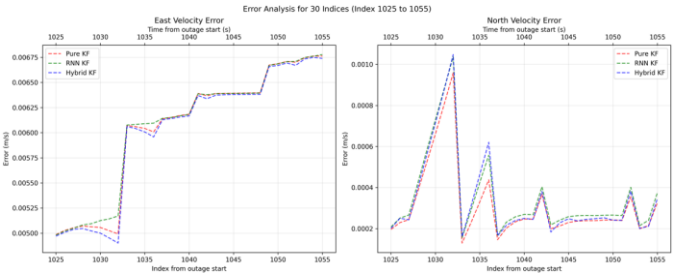


Fig. 19 Velocity Errors During 30s GPS Outage

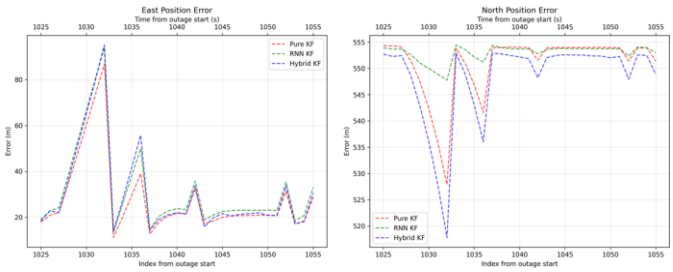


Fig. 20 Position Errors During 30s GPS Outage

## *B. 60 seconds*

Table VI Error Statistics During a 60s Driving GPS outage

| Error | Pure KF | | RNN KF | | Hybrid KF | |
|---|---|---|---|---|---|---|
| | Max | RMSE | Max | RMSE | Max | RMSE |
| East velocity (m/s) | 0.0115 | 0.0078 | 0.0114 | 0.0079 | 0.0114 | 0.0078 |
| North velocity (m/s) | 0.0014 | 0.0006 | 0.0015 | 0.0006 | 0.0016 | 0.0007 |
| Longitude (m) | 127.47 | 52.35 | 132.74 | 55.37 | 143.56 | 58.62 |
| Latitude (m) | 554.35 | 542.24 | 554.47 | 551.76 | 553.05 | 537.51 |

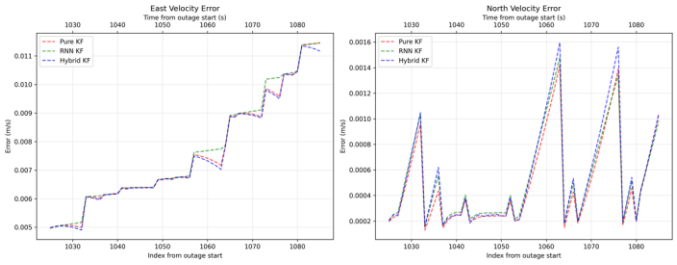


Fig. 21 Velocity Errors During 60s GPS Outage

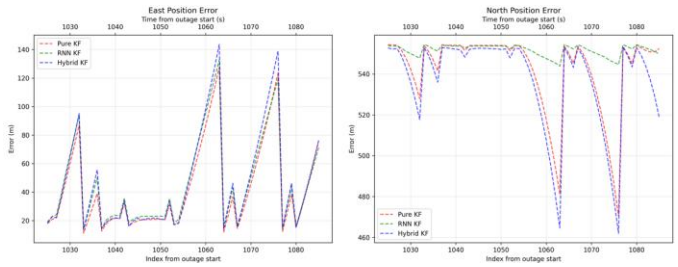


Fig. 22 Position Errors During 60s GPS Outage

### *C. 100 seconds*

Table VII Error Statistics During a 100s Driving GPS outage

| Error | Pure KF | | RNN KF | | Hybrid KF | |
|---|---|---|---|---|---|---|
| | Max | RMSE | Max | RMSE | Max | RMSE |
| East velocity (m/s) | 0.0018 | 0.0089 | 0.0018 | 0.0089 | 0.0018 | 0.0087 |
| North velocity (m/s) | 0.0042 | 0.0013 | 0.0042 | 0.0014 | 0.0042 | 0.0014 |
| Longitude (m) | 127.61 | 53.29 | 172.60 | 68.75 | 193.07 | 74.30 |
| Latitude (m) | 564.29 | 546.78 | 554.52 | 551.06 | 553.21 | 532.49 |

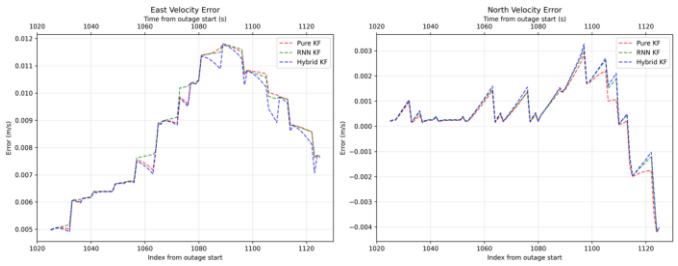


Fig. 23 Velocity Errors During 100s GPS Outage

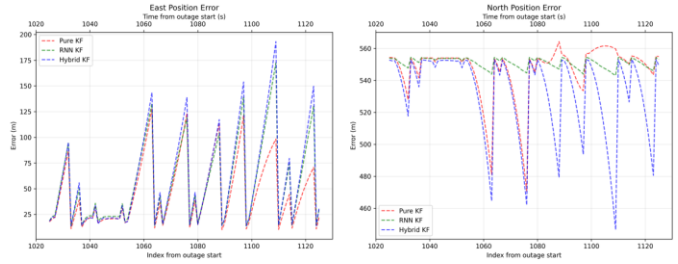


Fig. 24 Position Errors During 100s GPS Outage

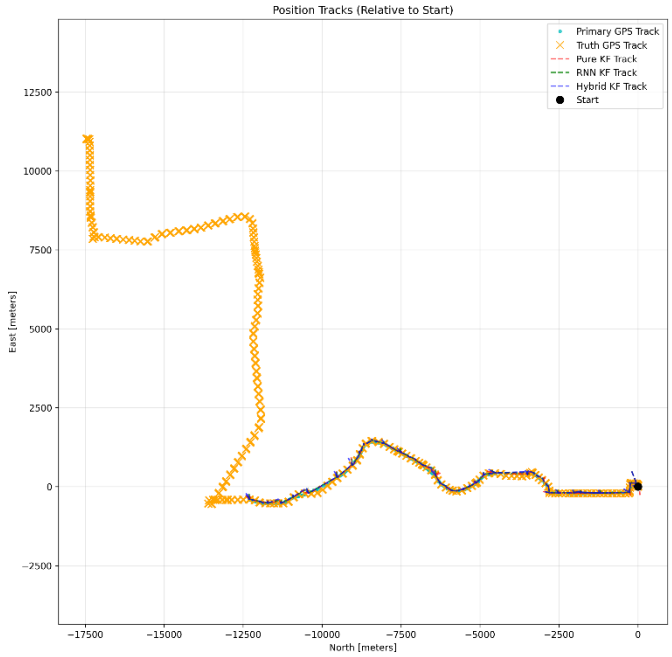


Fig. 25 2D Position Map of Kalman Performance

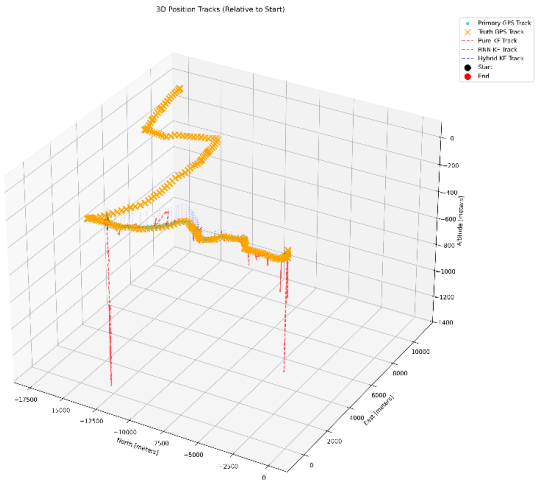


Fig. 26 3D Position Map of Kalman Performance

## VI. Related Work

Topics of RNN, LSTM, GRU, and Kalman filter applications helped craft the experimentation in this paper. Related work on applying RNN and its variants to Kalman filters are found at [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] with advanced applications, bridging the applications to high-velocity vehicles like UAVs. The related works stated have publication dates of 2020 to 2024. As technology advances and the computation complexity of RNNs is reduced, machine learning to augment systems will produce more research into Kalman filters and RNN integration.

## VII. Conclusions

A typical application of integrating RNN variants with Kalman filters is presented. Sensor fusion, high-fidelity INS systems, and more robust algorithms will extend how much time a system relies on navigation to predict its location during GPS and GNSS outages. The system's accuracy resulted in errors referring to the east velocity, north velocity, longitude, latitude, and altitude. Each error was reduced by X, Y, Z, T, respectively, noting the effectiveness of RNN and Kalman filter integration. Comparing the high reduction of errors compared to a vanilla Kalman filter integration showed a decrease in errors. Utilizing a simple model, one can grant enhanced precision to low SWAP-C devices, heightening the accuracy and effectiveness of devices that may inherently have higher gradients of noise and drift. Applying the RNN method with a loosely coupled approach is encouraged by vehicles, aircraft, and UAVs. First responders, military, and researchers benefit from low-cost, small, and highly commercially available INS devices. Vendors with access to a collection of platforms, motion models, and turn tables can produce far more accurate training sets that may one day be part of the sensor's base feature package. Future work will include integration within new platforms and exploring drone position reinforcement during eclipsed or degraded navigation.

## References


[1] Statista. "Global GNSS Device Installed Base 2020-2031." [Online]. Available: https://www.statista.com/statistics/1174544/gnss-device-installed-base-worldwide/. [Accessed: 4-Aug-2024].

[2] M. Hookham, "1,000 Planes a Day Have Signals Jammed as They Fly Over War Zones," The Times, 10-Dec-2024. [Online]. Available: https://www.thetimes.com/uk/technology-uk/article/1000-planes-a-day-have-signals-jammed-as-they-fly-over-war-zones-swtdflkqc?region=global. [Accessed: 4-Aug-2024].

[3] S. Erwin, "Global Navigation Jamming Will Only Get Worse; US Needs to Move Fast," SpaceNews, 4-Aug-2024. [Online]. Available: https://spacenews.com/global-navigation-jamming-will-only-get-worse-us-needs-move-fast/. [Accessed: 10-Oct-2024].

[4] RNT Foundation, "FAA Warns Airline Pilots as GPS Signals Disrupted Around Dallas," 18-Oct-2022. [Online]. Available: https://rntfnd.org/2022/10/18/faa-warns-airline-pilots-as-gps-signals-disrupted-around-dallas-bloomberg/. [Accessed: 04-Aug-2024].

[5] G. Gutt, "DHS Report on Denver Jamming: More Questions Than Answers," GPS World, 10-Dec-2024. [Online]. Available: https://www.gpsworld.com/dhs-report-on-denver-jamming-more-questions-than-answers/. [Accessed: 16-Nov-2024].

[6] Safran Navigation & Timing, "Measuring a GNSS Signal and Gaussian Noise Power." [Online]. Available: https://safran-navigation-timing.com/document/measuring-a-gnss-signal-and-gaussian-noise-power/. [Accessed: 9-Nov-2024].

[7] VectorNav Technologies, "Theory of Inertial Navigation." [Online]. Available: https://www.vectornav.com/resources/inertial-navigation-primer/theory-of-operation/theory-inertial. [Accessed: 15-Sep-2024].

[8] Wikipedia, "Kalman Filter." [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter#cite_note-1. [Accessed: 18-Sep-2024].

[9] T. Lacey, "Chapter 11 Tutorial: The Kalman Filter." [Online]. Available: https://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf. [Accessed: 18-Sep-2024].

[10] IBM, "Recurrent Neural Networks." [Online]. Available: https://www.ibm.com/topics/recurrent-neural-networks. [Accessed: 12-Sep-2024].

[11] D2L.ai, "Backpropagation Through Time." [Online]. Available: https://d2l.ai/chapter_recurrent-neural-networks/bptt.html. [Accessed: 1-Aug-2024].

[12] ArduSimple, "RTK Explained." [Online]. Available: https://www.ardusimple.com/rtk-explained/. [Accessed: 12-Sep-2024].

[13] J. W. Betz, Engineering Satellite-Based Navigation and Timing : Global Navigaton Satellite Systems, Signals, and Receivers. Piscataway, New Jersey: Ieee ; Hoboken, New Jersey, 2016.

[14] E. Roberts, "History of Neural Networks." [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html. [Accessed: 9-Sep-2024].

[15] F. Rosenblatt, "Perceptual Generalization over Transformation Groups," in Self-organizing Systems: Proceedings of an Inter-disciplinary Conference, M. C. Yovitz and S. Cameron, Eds. London: Pergamon Press, 1960, pp. 63–100.

[16] H. Idrees, "ANN vs. CNN vs. RNN vs. LSTM: Understanding the Differences in Neural Networks," Medium, 10-Dec-2024. [Online]. Available: https://medium.com/@hassaanidrees7/ann-vs-cnn-vs-rnn-vs-lstm-understanding-the-differences-in-neural-networks-94486cbb6d5a. [Accessed: 22-Sep-2024].

[17] C. Yanhui, "A Battle Against Amnesia: A Brief History and Introduction of Recurrent Neural Networks," Towards Data Science, 08-Mar-2021. [Online]. Available: https://towardsdatascience.com/a-battle-against-amnesia-a-brief-history-and-introduction-of-recurrent-neural-networks-50496aae6740. [Accessed: 19-Sep-2024].

[18] S. Linda, "Deep Learning Part 4: Recurrent Neural Network (RNN)," Medium, 10-Dec-2024. [Online]. Available:

https://medium.com/@sumbatilinda/deep-learning-part4-recurrent-neural-network-rnn-0714e0852581. [Accessed: 15-Aug-2024].
[19] GeeksforGeeks, "Introduction to Recurrent Neural Network." [Online]. Available: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/#. [Accessed: 23-Aug-2024].
[20] Baeldung, "Prevent the Vanishing Gradient Problem with LSTM." [Online]. Available: https://www.baeldung.com/cs/lstm-vanishing-gradient-prevention. [Accessed: 15-Aug-2024].
[21] LinkedIn, "How Do You Deal with Vanishing/Exploding Gradient?" [Online]. Available: https://www.linkedin.com/advice/3/how-do-you-deal-vanishing-exploding-gradient. [Accessed: 11-Nov-2024].
[22] A. Li and J. Mikhaylov, "A Deep Learning-Based Kalman Filter for GNSS/INS Integration," arXiv preprint arXiv:2210.08244, 2022. [Online]. Available: https://arxiv.org/pdf/2210.08244. [Accessed: 12-Dec-2024].
[23] S. Xing, F. Han, and S. Y. Khoo, "Extreme-long-short term memory for time-series prediction," arXiv preprint arXiv:2210.08244, Oct. 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.08244
[24] S. T. Goh, O. Abdelkhalik, and S. A. (Reza) Zekavat, "A Weighted Measurement Fusion Kalman Filter implementation for UAV navigation," Aerospace Science and Technology, vol. 28, no. 1, pp. 315–323, Jul. 2013, doi: https://doi.org/10.1016/j.ast.2012.11.012
[25] Xuefei Cui, Zhaocai Wang & Renlin Pei. (2023) A VMD-MSMA-LSTM-ARIMA model for precipitation prediction. Hydrological Sciences Journal 68:6, pages 810-839.
[26] A. Tang and B. Jiang, "Understanding GRU Networks," Towards Data Science, 10-Dec-2024. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be. [Accessed: 12-Nov-2024].
[27] P. Li et al., "Bidirectional gated recurrent unit neural network for Chinese address element segmentation," ISPRS International Journal of Geo-Information, vol. 9, no. 11, p. 635, Oct. 2020. doi:10.3390/ijgi9110635
[28] GeeksforGeeks, "Gated Recurrent Unit Networks," 10-Dec-2024. [Online]. Available: https://www.geeksforgeeks.org/gated-recurrent-unit-networks/#. [Accessed: 20-Nov-2024].
[29] STMicroelectronics, LSM9DS1 Datasheet, 2015. [Online]. Available: https://www.st.com/resource/en/datasheet/lsm9ds1.pdf. [Accessed: 1-Dec-2024].
[30] u-blox AG, ZED-F9T Datasheet, 2020. [Online]. Available: https://www.u-blox.com/sites/default/files/ZED-F9T-00B_DataSheet_UBX-18053713.pdf. [Accessed: 1-Dec-2024].
[31] Tang, Y., Jiang, J., Liu, J., Yan, P., Tao, Y., & Liu, J. (2022). A GRU and AKF-Based Hybrid Algorithm for Improving INS/GNSS Navigation Accuracy during GNSS Outage. Remote. Sens., 14, 752. https://doi.org/10.3390/rs14030752.

[32] Y. Zhang, "A Fusion Methodology to Bridge GPS Outages for INS/GPS Integrated Navigation System," IEEE Access, vol. 7, pp. 61296–61306, 2019, doi: 10.1109/ACCESS.2019.2911025.
[33] Jeon, B., Petrunin, I., & Tsourdos, A. (2021). Recurrent Neural Network based Sensor Fusion Algorithm for Alternative Position, Navigation and Timing. 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), 1-7. https://doi.org/10.1109/dasc52595.2021.9594472.
[34] Goh, S., Abdelkhalik, O., & Zekavat, S. (2013). A Weighted Measurement Fusion Kalman Filter implementation for UAV navigation. Aerospace Science and Technology, 28, 315-323. https://doi.org/10.1016/J.AST.2012.11.012.
[35] Tang, Y., Jiang, J., Liu, J., Yan, P., Tao, Y., & Liu, J. (2022). A GRU and AKF-Based Hybrid Algorithm for Improving INS/GNSS Navigation Accuracy during GNSS Outage. Remote. Sens., 14, 752. https://doi.org/10.3390/rs14030752.
[36] Shin, Y., Lee, C., Kim, E., & Walter, T. (2021). Adopting Neural Networks in GNSS-IMU integration: A Preliminary study. 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), 1-7. https://doi.org/10.1109/DASC52595.2021.9594430.
[37] Guang, X., Gao, Y., Liu, P., & Li, G. (2021). IMU Data and GPS Position Information Direct Fusion Based on LSTM. Sensors (Basel, Switzerland), 21. https://doi.org/10.3390/s21072500.