INTERNATIONAL JOURNAL OF AUTOMOTIVE SCIENCE AND TECHNOLOGY

2023, VOL. 7, NO: 2, 154-166 www.ijastech.org e-ISSN: 2587-0963

Comparing the Performance of Different Methods for Estimation in Inertial Navigation Systems

Bekir Göğüş[1], G. Begüm Cangöz[1] and Murat Üçüncü[1*]

0000-0002-5902-3464, 0000-0001-8469-5484, 0000-0002-2113-1398

[1] *Electrical Electronics Engineering Department, Faculty of Engineering, Baskent University, Ankara, 06790, Turkey*

Abstract

There are many positioning systems available today. The most prominent of these systems is the Inertial Navigation System, which is increasingly preferred because it works with its own internal system independent of external stimuli. This system detects position, orientation and velocity information by means of accelerometer and gyroscope sensors. Using this information, it is possible to make predictions for the next position, orientation and speed with various algorithms. In the studies conducted so far, Kalman Filter (KF) algorithms have been predominantly used for prediction. In this study, Long Term-Short Memory (LSTM) neural network architecture, Bidirectional Long Short-Term Memory (BLSTM), Gated Recurrent Unit (GRU) and Kalman Filtering methods, which are among the deep learning algorithms that have proven themselves as prediction algorithms, are examined in detail and a comparative study is presented. Here, LSTM, BLSTM and GRU deep learning networks were first trained with IMU sensor data and speed estimation was performed. Root Mean Squared Error (RMSE) values were obtained as 2.5547, 2.7592 and 2.5414, respectively. Furthermore, the same deep learning network methods were trained with GPS data. The prediction data obtained through LSTM, BLSTM and GRU provided RMSE values of 0.42542, 1.91122 and 0.32274, respectively. We see that the prediction with GPS data have higher accuracy since deep learning networks trained with GPS were less affected by noise during the training phase.

Keywords: Accelerometer, Rotation meter, Deep learning, Kalman filter, LSTM

Research Article

https://doi.org/10.30939/ijastech..1174226

Received 12.09.2022
Revised 04.04.2023
Accepted 03.05.2023

* Corresponding author

Murat ÜÇÜNCÜ

murat.ucuncu@yahoo.com.tr

Address: Başkent University, Faculty of Engineering, Electrical and Electronics Engineering Department, Etimesgut / Ankara, Turkey

Tel: +905337746924

1. Introduction

There are estimation methods that estimate the speed and position of a pedestrian, automobile, aircraft or any transportation vehicle in indoor and outdoor environments. These methods can use Inertial Measurement Unit (IMU) data, camera images or Global Positioning System (GPS) data in the estimation phase. GPS, which is one of the most widely used navigation systems in our daily life, becomes useless in closed areas or in places where the phone signal strength is weak. Therefore, alternative navigation systems to GPS are being developed. One of the alternative navigation system to GPS is the Inertial Navigation Systems (INS). INS are capable of providing speed and position information without an extra stimulus or any external signal [1]. INS consist of internal accelerometers, gyroscope and magnetometers. This group of sensors is called the Inertial Measurement Unit. The data coming from the IMU is processed by combining the data with sensor fusion algorithms and the result is obtained by estimating speed or position information by using various estimation algorithms. The Kalman Filter (KF) [2] is commonly used in IMU-assisted navigation applications. In addition to the Kalman Filter, which is a traditional estimation method, deep learning networks such as Long Term-Short Memory (LSTM) [3] are also used in the field of estimation.

In this study, Kalman Filter, Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BLSTM), Gated Recurrent Unit (GRU) algorithms are used for speed estimation of an automobile. With the help of these algorithms, acceleration data from sensors were processed and state estimation was performed. As a result of the velocity estimation, the state estimation performances of the algorithms were compared, the effects of the Q (process noise covariance) and R (measurement

noise covariance) parameters of the Kalman filter on the estimation results were examined, and in addition, the optimizers in the deep learning network algorithms were changed and their effects on the velocity estimation results were observed. The speed estimation values obtained as a result of the studies are compared with GPS speed data and the performances of RMSE estimation algorithms are compared.

2. Related Works

In the literature, there is a large number of estimation studies based on IMU. These studies generally include running speed estimation for sports people, aircraft speed estimation, or gait speed estimation for gait analysis of sick individuals.

In their study [4], Liu et al. used multi-sensor data to estimate the speed and position of a pedestrian. They used the Kalman filter combined with the Zero Velocity Update (ZVO) algorithm.

They also tried to increase their estimation accuracy by adding some adjustable parameters to their filters. They have demonstrated the accuracy of their proposed methods in a pedestrian test environment. They also created a test environment under long-haul conditions and showed that their proposed methods gave successful results here as well.

In their study [5], Svacha et al. took UAV engine speed measurements at the PERCH laboratory in the indoor test environment at the University of Pennsylvania. The conventional RANSAC algorithm was used to estimate the UAV roll rate and Unscented Kalman Filter (UKF) was used to estimate the UAV speed and altitude. Smaller RMS values were obtained by using camera images.

In their study [6], Wang et al. collected acceleration data from a wearable device while exercising on a treadmill. A deep convolutional neural network (CNN) model was used for motion speed estimation. The model outputs were shown to be accurate in RMSE within 7% and 18% of the actual running and walking test speeds respectively.

In their study [7], Gençoğlu et al. had people shoot balls at one-minute rest intervals. They used the Generalized Linear Model, Gradient Augmented Trees, and Support Vector Machine to predict the ball flight. They compared the ball velocities recorded with a radar velocity gun with the prediction results. The performance measures of the Generalized Linear Model, Gradient Augmented Trees and Support Vector Machine were calculated in terms of root mean square errors, absolute errors and correlation coefficients. As a result, it was shown that the data obtained from accelerometers accurately predicted the ball firing speed with the help of machine learning models.

In their study [8], Jain et al. collected global satellite navigation system (GNSS) and IMU data in an urban area for two hours with a vehicle. Velocity estimation was performed by using least squares (LS) and LKF (Linearized Kalman Filter). The velocity estimates were compared with a reference trajectory and the deviations of the variance models were evaluated. They found that the estimated velocity root mean square error (RMSE) was a minimum of less than 16% and a maximum of about 41%.

3. Data Set

During the creation of the data set, the acceleration data of a car was obtained with the help of the sensors of a smartphone. Before starting to create the data set, a suitable measurement environment was created. While the smartphone was placed on the console between the driver's seat and the side seat of the car for measurement, the roll, pitch and yaw angles were adjusted to 0º with the help of the smartphone's spirit level application and a suitable measurement environment was created by fixing the smartphone. While creating the data set, an intercity road with a road length of 30.4 km was used where the slope was not very variable, and measurements were also taken on winding and sloping roads and measurements were made under different road and driving dynamics. Figure 1 shows a representative view of the smartphone on which the sensor readings were taken.

Fig.1. Smartphone sensor axes.

The data set includes time-series sensor data from the phone's accelerometer and rotation meter sensors. GPS data at a frequency of 1 Hz (every 1 second) was also recorded during the measurement to compare the estimation results in this study. During the measurements, Apple Iphone 13 Pro running IOS 15.0 was used as a smartphone. Data from the sensors were sampled at a frequency of 20 Hz (every 50 milliseconds). The details of the data set are presented in Table 1.

Table 1. The details of the data set used.

The number of activity during measurement	4
Application used	MATLAB mobile
The distance of measurements	30,4 km
Sensor sampling frequency	20 Hz
Used smartphone	Apple Iphone 13 Pro
Total number of raw measurement	176274

In this study, only the accelerometer sensor data of the smartphone was used from the created data set, and the measurements of the car in the -y axes were processed. The unit for the accelerometer sensor is specified as m/s^2 in the MATLAB mobile program output. The data set information used in the studies is given in Table 2. While creating the data set, 4 different scenarios were followed.

In the first scenario, the smartphone was held without acceleration to determine the constant drift error of the sensors.

In the second measurement scenario, measurements were taken with the cruise control feature of the car and the data was recorded to obtain sensor noise by comparing the data under speed with the speed data obtained with GPS. In the third measurement scenario, acceleration data was measured and recorded under variable speeds to be used for speed estimation within the scope of this study.

In the final measurement case, the fourth measurement scenario, a driving environment with sudden increases and decreases in acceleration was created in order to observe how the speed estimation algorithms designed within the scope of this study would react to acceleration changes even more drastic than the third measurement scenario. The data set information used in the studies is given in Table 2.

Table 2. Data set information used in the studies.

Driving Style	Axis information	Number of measurements
Scenario 1	-y	56580
Scenario 2	-y	6534
Scenario 3	-y	56580
Scenario 4	-y	56580

4. Method

4.1 System Model

In this study, velocity estimation was performed by Kalman Filter, LSTM, BLSTM and GRU algorithms by using the measurements in the data set. For the estimation methods, the mathematical model of the system was first created. Newton's equations of motion and the discrete white noise model (Gaussian White Noise) were used to create the mathematical model of the system. Since the data in x and y axis are used in this study, the system model consists of the position and velocity parameters in x and y axis. In this case, the system dynamics equations are given by equations (1-5).

Equations (1) and (2) show the comparison between the current value of the velocity and its value in the next step for the x and y axis. In the equations, k is the time index. When formulating the formulas, the velocity is assumed to be constant and the system is modeled accordingly. In Equations (3) and (4), the position information for the x and y axes in the next step is given in terms of current position, current velocity and time information.

$$V_x(k+1) = V_x(k) \tag{1}$$

$$V_y(k+1) = V_y(k) \tag{2}$$

Here, V_x is the velocity on the x-axis and V_y is the velocity on the y-axis.

$$X_x(k+1) = X_x(k) + \Delta t V_x(k) \tag{3}$$

$$X_y(k+1) = X_y(k) + \Delta t V_y(k) \tag{4}$$

Here, X_x is the position on the x-axis, X_y is the position on the y-axis and Δt represents the sensor sampling period

Based on the equations obtained above, the state matrix of the system is obtained as in equation (5). The term Δt in the equations indicates the periods at which the data were received from the sensors. In the data set described in Section 3.1, the measurement frequency of both smartphone and smartwatch sensors is 20 Hz. In other words, data is acquired at every 50 ms. Therefore, the Δt value is 0.05 s. When the GWN model (discrete white noise model) in Equation (5) is added, Equation (6) is obtained. The output response of the system is then given by equation (7).

$$\begin{bmatrix} X_x(k+1) \\ V_x(k+1) \\ X_y(k+1) \\ V_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_x(k) \\ v_x(k) \\ x_y(k) \\ v_y(k) \end{bmatrix} + G w_n \tag{5}$$

$$\begin{bmatrix} X_x(k+1) \\ V_x(k+1) \\ X_y(k+1) \\ V_y(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_x(k) \\ v_x(k) \\ x_y(k) \\ v_y(k) \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ \Delta t & 0 \\ 0 & \frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} w_x(n) \\ w_y(n) \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} V_x(k) \\ V_y(k) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_x(k) \\ V_x(k) \\ X_y(k) \\ V_y(k) \end{bmatrix} + v_n \tag{7}$$

4.2 Rotation Matrix

Smartphones express the data obtained through their sensors using a 3-axis coordinate axis. When the smartphone is in its default posture, the axes are defined according to the screen of the device. In the data set created for use in this study, the sensor axes of the smartphone from which the measurements were taken are described in Chapter 3 under the data set heading.

While creating the data set, it is difficult to take a linear acceleration measurement, even though the path where the measurements are taken should be straight and without slopes. While taking measurements for the data set, the automobile is under the influence of roll angle, pitch angle and yaw angle due to the disturbances on the measured road. When the smartphone sensors in the car exposed to these angles measure acceleration data, the effect of gravity will be observed in axis other than the z-axis where gravitational acceleration is measured. In this case, the measurements in other axes must be gravity-free before the

acquired acceleration data can be used. If this is not done, velocity estimation from acceleration data will give erroneous results. As an example, Figure 2 shows the representation of the angles on the vehicle

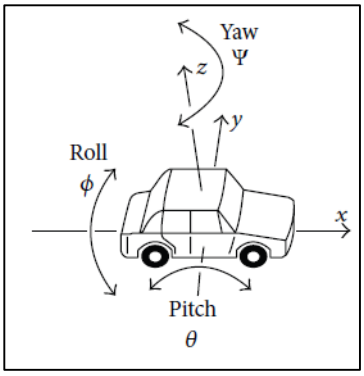

Fig. 2. Representation of angles on a car (9).

In the data set obtained with the help of smartphone sensors, the acceleration on the x and y axes, which is distorted by the acceleration of gravity, is corrected with the help of data rotation matrices. The rotation matrix is denoted by the letter R in Equation (8) (10).

$$R=\begin{bmatrix} E_x & E_y & E_z \\ N_x & N_y & N_z \\ G_x & G_y & G_z \end{bmatrix} \quad (8)$$

x, y and z in matrix R represent the axis of the smartphone. E and N are unit vectors pointing east and north respectively. G is the gravity vector.

The Euler angles used in the rotation matrix are denoted by ψ, θ and φ. The azimuth angle is denoted by the letter ψ and refers to rotation about the vector G. The pitch angle is denoted by the letter θ and refers to the rotation around the E vector. Finally, the roll angle is denoted by the letter φ and refers to the rotation around the N vector.

When smartphone sensors are exposed to any angle during the measurement and are affected by gravity, the measurements are de-gravitated with the help of the rotation matrix. For this purpose, there is a rotation matrix for azimuth, pitch and roll angles. These matrices are given in equations (9), 10) and (11) respectively.

$$R_\psi = \begin{bmatrix} cos\ \psi & sin\ \psi & 0 \\ -sin\ \psi & cos\ \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$R_\theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\ \theta & sin\ \theta \\ 0 & -sin\ \theta & cos\ \theta \end{bmatrix} \quad (10)$$

$$R_\varphi = \begin{bmatrix} cos\ \varphi & 0 & sin\ \varphi \\ 0 & 1 & 0 \\ -sin\ \varphi & 0 & cos\ \varphi \end{bmatrix} \quad (11)$$

The transformation of the data measured by the smartphone sensors from the phone coordinate system to the world coordinate system is given in equation (12).

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = R(\psi,\theta,\varphi)\begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T \quad (12)$$

With the last step, the acceleration data is transformed from the phone coordinate system to the world coordinate system and only the z-axis of the transformed acceleration data contains gravitational acceleration. Equation (13) is applied to subtract the gravitational effect from the z-axis component. The expression g in the equation represents gravity and its value is 9.81m/s2.

$$A_{linear} = \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} - g[0 \quad 0 \quad 1]^T \quad (13)$$

The linear matrix A obtained in the last step consists of gravity-adjusted acceleration data and is made available for processing.

4.3 Kalman Filter

The method used to estimate the values of some variables from data observed in a certain period is called Kalman filtering [11]. The Kalman filter has been a fundamental method for analyzing and solving estimation problems with different variations [12]. Basically, the Kalman filter is a set of mathematical formulas that minimizes the estimated error covariance under certain conditions and corrects the estimates. When making predictions, it also supports past, present and future state predictions. One of the strength of the filter is that it can do this even when the structure of the system model is not known in an absolute way [13].

The process model of the linear dynamic system in state space format is given in equation (14). The state variables of the system are assumed to have zero mean (equation (15,17)). The measurement model is also given in equation (16).

$$x_k = Fx_{k-1} + Gu_{k-1} + w_{k-1} \quad (14)$$

$$w_k \sim (0, Q) \quad (15)$$

$$z_k = Hx_k + v_k \quad (16)$$

$$v_k \sim (0, R) \quad (17)$$

Here, F is the state transition matrix, x_{k-1} is the previous state vector, G is the control input matrix and u_{k-1} is the input control vector. w_{k-1} represents the process noise vector and Q and R are covariance matrices.

4.4 Long-Short Term Memory (LSTM)

It is a recurrent neural network (RNN) model introduced to the literature by Sepp Hochreiter and Jürgen Schmidhuber in

1997. These structures differ from traditional feed-forward neural networks in that they have feedback structures. LSTM has applications in many fields such as audio processing [14], image processing [15], language processing [16]. A standard LSTM block consists of four gates. Figure 3 shows an illustration of an LSTM block.

4.5 Bidirectional Long Short-Term Memory (BLSTM)

BLSTM was introduced in 1997 by Schuster and Paliwall, BLSTM is a type of feedback neural network. These neural network structures allow processing data from both the next layer and the previous layer. In other words, the difference from a traditional RNN (Recurrent Neural Network) structure is that calculations are made according to the values coming from two layers. One layer receives its input from the forward direction and the other from the backward direction. In this way, applying the LSTM twice leads to better learning for long-term data and therefore leads to improved model accuracy [18]. Figure 4 shows the general structure of BLSTM.

4.6 Gated recurrent unit (GRU)

The GRU neural network, proposed by Kyunghyun Cho and colleagues in 2014, draws attention with its similarity to LSTM. However, it has fewer parameters compared to LSTM and has been shown to perform better in smaller data sets [20]. There are also studies in the literature where the GRU neural network achieves more successful results than the LSTM neural network [21], [22]. The general representation of the GRU deep learning network is given in Figure 5.

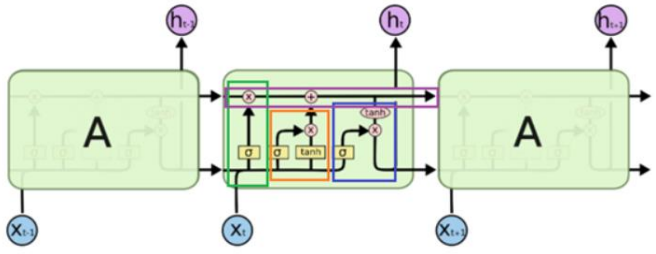

Fig. 3. A standard LSTM block [17].

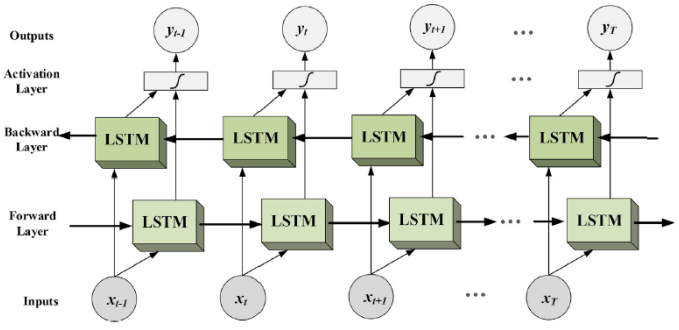

Fig. 4. BLSTM structure [19].

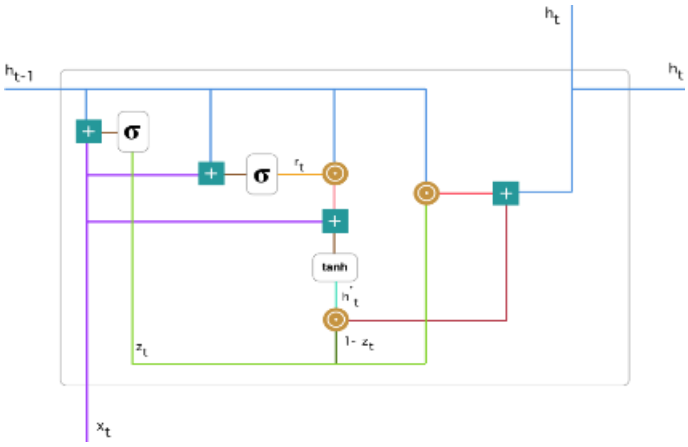

Fig. 5. A standard GRU structure [20].

5. Results

5.1 Kalman Filter Simulation Results

The data set used in this study is given in Figure 6.

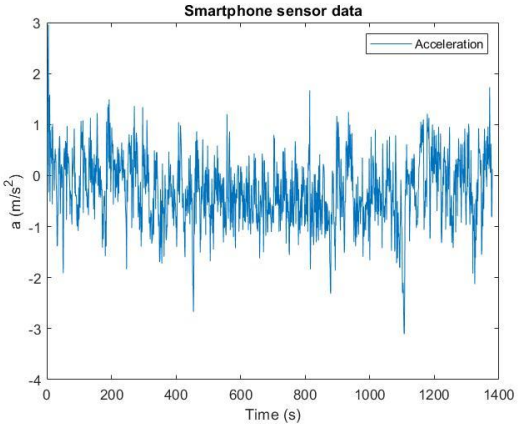

Fig. 6. Y-axis acceleration data obtained from smartphone accelerometer sensors.

To compare the Kalman filter speed estimates with the deep learning-based estimation results, the estimation results with the last 10% of our data set are given in Figure 7. When the graph is analyzed, the Kalman filter is subject to a cumulative error due to sensor measurement errors as time passes and after a while there is a large difference between the speed estimation and GPS values.

When the velocity estimation results obtained with the Kalman Filter are examined, as can be seen in Figure 7, it is thought that there may be a constant shift in the acceleration data due to temperature changes and sensor calibration errors during the measurement from the sensors, as well as the effect of noise coupling. For this reason, we tried to remove the constant drift of the smartphone sensor data as much as possible. The constant drift of the acceleration data was removed by using the moving average of the smartphone acceleration data. The cleaned version of the acceleration data is given in Figure 8.

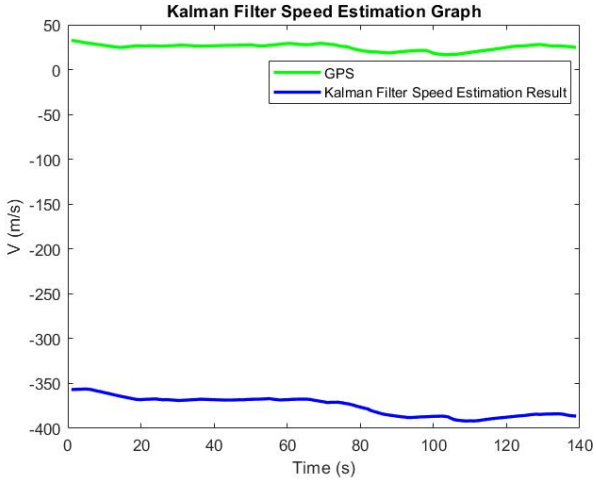

Fig. 7. Kalman Filter speed estimation graph (138 seconds).

The acceleration data, which is free of constant drift, is applied to the Kalman Filter velocity estimation algorithm. Velocity estimation results are given in Figure 9. When the velocity estimation graph with the cleaned acceleration data is analyzed, it is seen that much more successful estimation results are obtained compared to the velocity estimation in Figure 7. The linear drift in Figure 7 is largely eliminated.

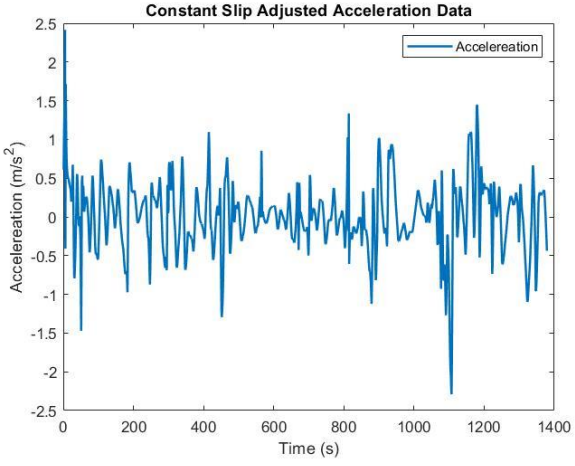

Fig. 8. Acceleration data adjusted for constant drift.

In Figure 9, there are small shifts in the speed estimation outputs. Although there is a shift in the speed estimation outputs, the speed estimation outputs also respond to changes in GPS speed data. As a result, the large shifts in the estimation results previously obtained through the Kalman Filter were removed by taking a moving average of the smartphone acceleration data, and more successful results were obtained.

The velocity estimations carried out by the Kalman Filter using the acceleration data free of fixed drifts are given in Figure 9. When the graph is analyzed, there are shifts in the Kalman filter velocity estimation results. However, compared to the first velocity estimation case in Figure 7, it provided more successful results.

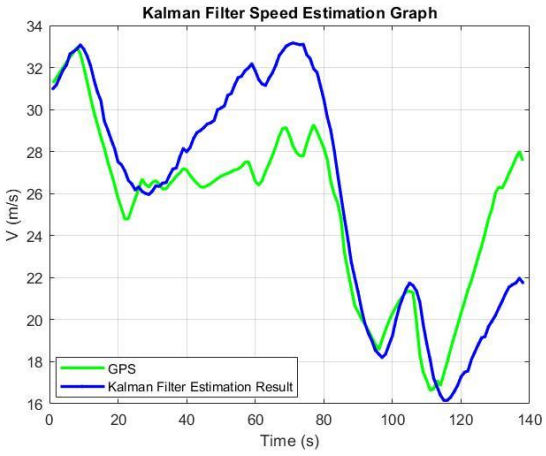

Fig. 9. Kalman Filter velocity estimation with purified acceleration data (138 seconds).

In the last 138 seconds of the velocity estimation result obtained by the Kalman filter using the acceleration data adjusted for the constant drift, the RMSE value was 2.9322 and its graph is given in Figure 10. As can be seen, more successful results were obtained as compared to the estimation results of the previous case given in Figure 7. The RMSE plot of the velocity estimation using the acceleration data adjusted for the constant drift is given in Figure 10.

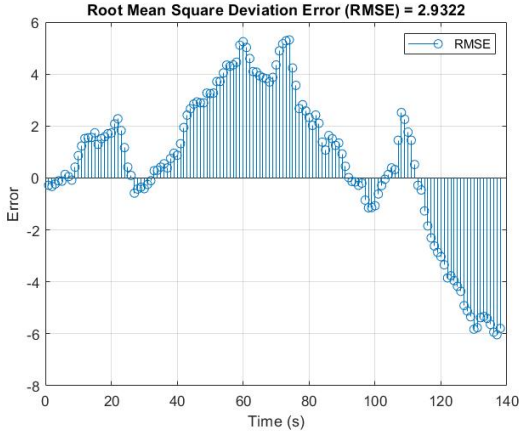

Fig. 10. RMSE values obtained from estimation (138 seconds).

5.2 LSTM Deep Learning Network Simulation Results

LSTM requires a training set to learn the long-term dependencies in the data to start the estimation process. The data set used in this study is divided into training and test sets as 9:1, respectively. 1380 samples were taken from the data set and 1242 out of these samples were used to build the LSTM model, while the remaining 138 were used to test the model.

The standardized velocity data are given as input to the LSTM layer, transformed into a single vector with the fully coupled layer, and in the final stage, the regression layer is used to produce the model outputs. In the algorithm, maximum training round (Max. Epoch), gradient threshold, initial learning rate,

number of hidden units (NumHiddenUnits), mini batch size, shuffle, L2 regularization, validation frequency, gradient threshold method and momentum parameters are kept constant. In addition, Adam, RMSprop and SGDM optimizers were used to optimize the model by changing the learning step size (Learn Rate Drop Period) and the learning rate drop factor (Learn Rate Drop Factor), which affect the algorithm success the most.

There are 3 types of optimization algorithms in the LSTM method used in this study. For each optimization algorithm, four parameter groups were selected and RMSE values of the deep learning network were obtained under different conditions. The RMSE values obtained in this study are given in Table 3. In order to examine the effects of the learning rate on the prediction success of the algorithm, the Initial Learn Rate was chosen as 0.005 and this value was kept constant for each parameter state. The initial learning rate was varied in certain steps with the help of the parameters Learn Rate Drop Period and Learn Rate Drop Factor in order to have an updatable structure throughout the training phase of the LSTM network.

Considering the LSTM speed estimation RMSE values of the first parameter case, which is considered as the best success obtained with the Adam optimizer, and the estimation graph with RMSE values in this scenario are given in Figure 11. When the graph is analyzed, there is a speed estimation consisting of 138 seconds. Since the velocity data used to train the LSTM deep learning network is calculated from acceleration data containing noise residuals and shifts, the velocity estimation graph also contains shifts as a result of these noise residuals and shifts.

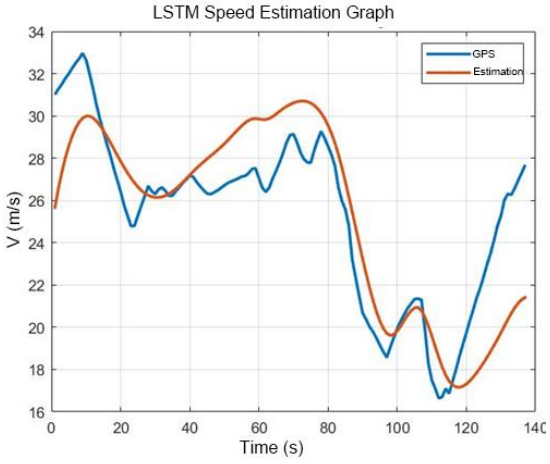

Fig. 11. LSTM speed estimation graph with IMU.

Figure 12 shows the RMSE value per second. Significant drifts are observed especially at the endpoints.

In the first part of the study, the speed estimation was done with the data obtained from the IMU sensors. We also found it useful to carry out the same speed estimation with GPS data. Therefore, the same deep learning algorithms were trained with GPS data. This will enable us to compare the speed estimation based on IMU data and GPS data. The speed estimation results obtained by training deep learning networks with GPS data are given in Table 4 below in terms of RMSE.

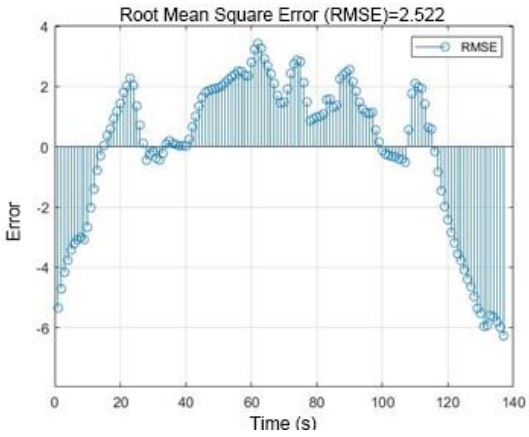

Fig. 12. LSTM speed estimation RMSE values graph

Table 3. LSTM deep network speed estimation results with IMU.

Optimizer	Parameter Status	Parameter	Value	RMSE
Adam	1	LDP	2	2,5547
		LDF	0,99	
	2	LDP	5	2,81425
		LDF	0,89	
	3	LDP	10	2,84424
		LDF	0,5	
	4	LDP	none	2,9675
		LDF	none	
RMSProp	1	LDP	2	3,32687
		LDF	0,99	
	2	LDP	5	2,9655
		LDF	0,89	
	3	LDP	10	2,8963
		LDF	0,5	
	4	LDP	none	2,7691
		LDF	none	
SGDM	1	LDP	2	2,56546
		LDF	0,99	
	2	LDP	5	2,57395
		LDF	0,89	
	3	LDP	10	3,4473
		LDF	0,5	
	4	LDP	none	2,85125
		LDF	none	

Table 4. LSTM deep network speed estimation results with GPS data.

Optimizer	Parameter Status	Parameter	Value	RMSE
Adam	1	LDP	2	1,98357
		LDF	0,99	
	2	LDP	5	0,42542
		LDF	0,89	
	3	LDP	10	0,89424
		LDF	0,5	
	4	LDP	none	0,95758
		LDF	none	
RMSProp	1	LDP	2	0,986875
		LDF	0,99	
	2	LDP	5	1,22855
		LDF	0,89	
	3	LDP	10	0,86193
		LDF	0,5	
	4	LDP	none	1,31475
		LDF	none	
SGDM	1	LDP	2	0,54666
		LDF	0,99	
	2	LDP	5	0,71395
		LDF	0,89	
	3	LDP	10	1,0101
		LDF	0,5	
	4	LDP	none	0,94712
		LDF	none	

We see that best estimate having the minimum RMSE value was obtained with Adam optimizer number 2. The estimation results are given in Figure 13.

The prediction results of the deep learning network trained with GPS provided estimates with higher accuracy as compared with the results trained with IMU data. The prediction errors are also given in Figure 14.

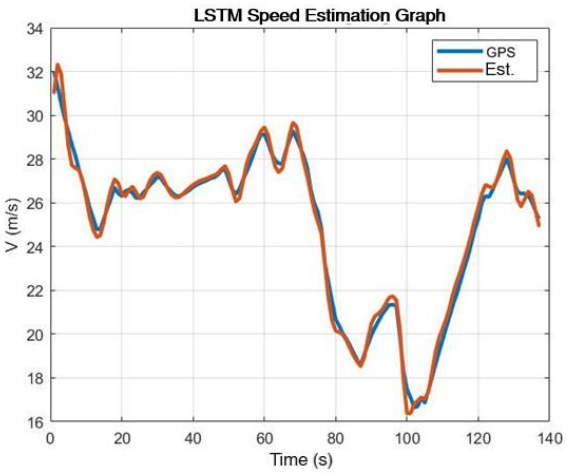

Fig. 13. LSTM speed estimation versus GPS data.

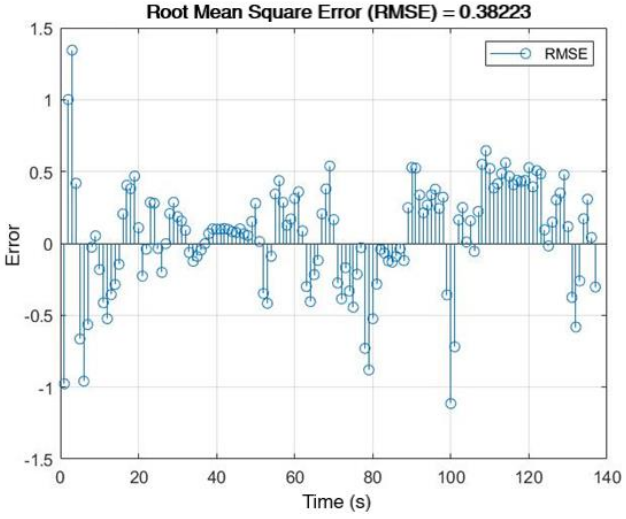

Fig. 14. LSTM speed estimation RMSE values.

5.3 BLSTM Deep Learning Network Simulation Results

As with the LSTM algorithm, the BLSTM method also requires a training and test set. In order to make a comparison with the success obtained from the LSTM model, the data set and parameters used in the LSTM algorithm were preferred. As applied to the LSTM deep learning network, three existing optimizers were used to estimate the speed for four different parameter cases and the results were tabulated. While the outputs are given in Table 5, the SGDM (Stochastic Gradient Descent with Momentum) optimizer with the first parameter case achieved the lowest RMSE value. However, although the Adam optimizer in the first parameter case obtained a value close to the SGDM speed estimation, the LSTM deep learning network is more advantageous because it requires longer training time compared to the LSTM deep learning network. Another remarkable aspect of the BLSTM deep learning network is that it achieves more successful results when the parameter values are not updated.

When RMSE values for the BLSTM velocity estimation are considered, the velocity estimation graph for the first parameter state, which was found to be the most successful and made with

the SGDM optimizer, is given in Figure 15. A shift is observed in the speed estimation values compared to GPS. When the RMSE values of LSTM deep learning network and BLSTM deep learning network are compared in general, BLSTM is more inconsistent than LSTM deep learning network and BLSTM deep learning network takes longer training time. Therefore, it is not reasonable to use the BLSTM deep learning network for this kind of prediction problems.

The RMSE values of the speed estimation results of the BLSTM deep learning network at each second are given in Figure 16. When Figure 16 is analyzed, there are quite high shifts especially in the initial speed estimations. The same shifts also reached high values at the end point. However, the error values are low at the midpoints of the velocity estimation graph. In general, results that can be considered successful were obtained for the velocity estimation, except at the start and end points.

Table 5. BLSTM deep network speed estimation results with IMU

Optimizer	Parameter Status	Parameter	Value	RMSE
Adam	1	LDP	2	2,9296
		LDF	0,99	
	2	LDP	5	2,7709
		LDF	0,89	
	3	LDP	10	3,0256
		LDF	0,5	
	4	LDP	none	2,7812
		LDF	none	
RMSProp	1	LDP	2	2,84423
		LDF	0,99	
	2	LDP	5	4,08124
		LDF	0,89	
	3	LDP	10	3,6455
		LDF	0,5	
	4	LDP	none	2,82871
		LDF	none	
SGDM	1	LDP	2	2,7592
		LDF	0,99	
	2	LDP	5	4,3457
		LDF	0,89	
	3	LDP	10	5,81412
		LDF	0,5	
	4	LDP	none	2,9105
		LDF	none	

The GPS data set was used for training purposes in the BLSTM deep learning network as well as in the LSTM deep learning network and speed estimation was performed. The results are given in Table 6.

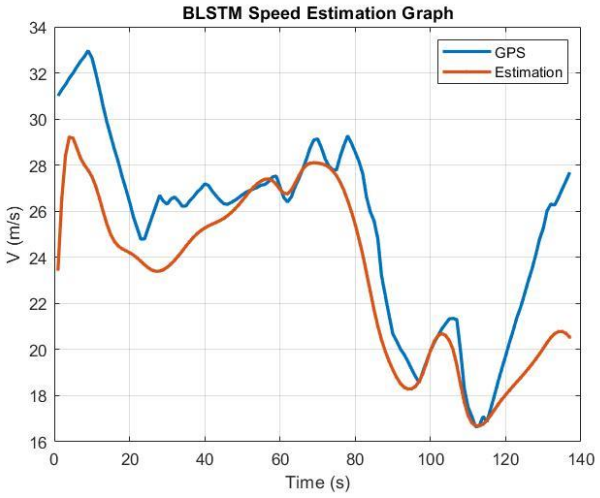

Fig. 15. BLSTM speed estimation prediction values graph with IMU.

Table 6. BLSTM deep network speed estimation results with GPS

Optimizer	Parameter Status	Parameter	Value	RMSE
Adam	1	LDP	2	2,0296
		LDF	0,99	
	2	LDP	5	2,3709
		LDF	0,89	
	3	LDP	10	2,5865
		LDF	0,5	
	4	LDP	none	1,99435
		LDF	none	
RMSProp	1	LDP	2	2,34423
		LDF	0,99	
	2	LDP	5	2,8124
		LDF	0,89	
	3	LDP	10	2,1055
		LDF	0,5	
	4	LDP	none	3,2871
		LDF	none	
SGDM	1	LDP	2	1,91122
		LDF	0,99	
	2	LDP	5	2,3457
		LDF	0,89	
	3	LDP	10	2,86412
		LDF	0,5	
	4	LDP	none	2,0105
		LDF	none	

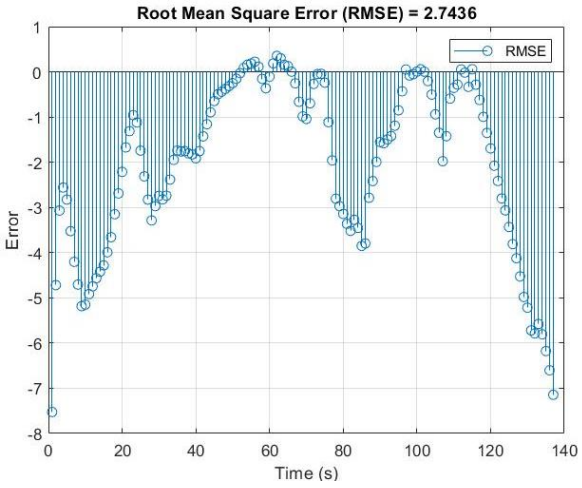

Fig. 16. BLSTM speed estimation RMSE values graph.

When the BLSTM velocity estimation RMSE values are considered, SGDM optimizer with number 1 parameter provides the best estimate which is also illustrated in Figure 17. Although the speed estimation values are able to predict the moments of speed change compared to GPS, there are shifts in the graph. When the RMSE values of LSTM deep learning network and BLSTM deep learning network are compared in general, BLSTM fails as compared to LSTM deep learning network and there are shifts in the velocity estimation results.

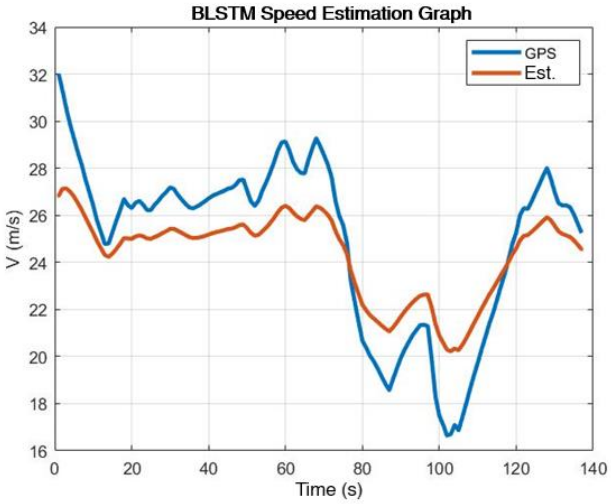

Fig. 17. BLSTM speed estimation prediction values graph with GPS

The RMSE values of the speed estimation results of the BLSTM deep learning network at each second are given in Figure 18. When Figure 18 is analyzed, there are shifts at higher values throughout the graph compared to the LSTM deep learning network.

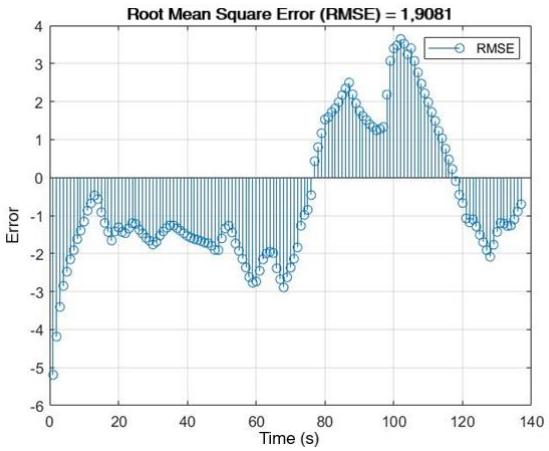

Fig. 18. BLSTM speed estimation RMSE values graph.

5.4 GRU Deep Learning Network Simulation Results

In the literature, GRU is a deep learning network model that is presented as an alternative to the LSTM algorithm and structurally differentiated from LSTM. For this reason, we wanted to make a performance comparison with the LSTM and BLSTM methods used in this study.

The parameters used in previous deep network models are also used in this structure. Adam, RMSprop and SGDM optimizers used in LSTM and BLSTM deep learning networks are also used in GRU neural network. In order to be able to compare the RMSE values with the previous deep learning networks, the parameter conditions were also chosen to be the same. The results are presented in Table 7.

The velocity estimation results with the best RMSE values obtained with the GRU deep learning network are given in Fig.19. While the LSTM deep learning network has speed estimation errors in the initial states, the GRU deep learning network eliminates this problem. The GRU deep learning network outperformed the LSTM and BLSTM deep learning networks in the overall speed estimation of 138 seconds.

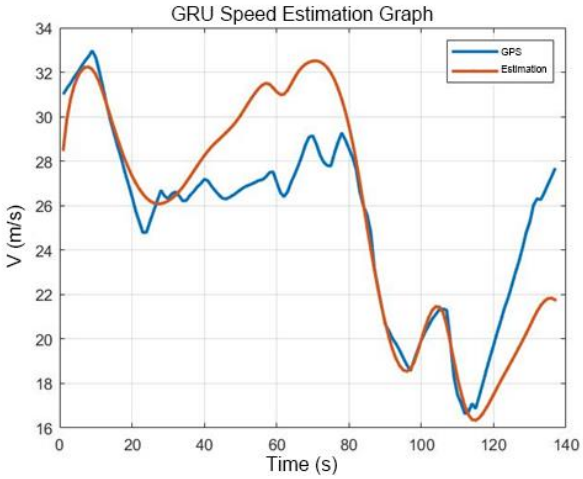

Fig. 19. GRU speed estimation prediction values graph with IMU.

Table 7. GRU deep network speed estimation results with IMU.

Optimizer	Parameter Status	Parameter	Value	RMSE
Adam	1	LDP	2	2,81824
		LDF	0,99	
	2	LDP	5	2,626
		LDF	0,89	
	3	LDP	10	2,6324
		LDF	0,5	
	4	LDP	none	2,6847
		LDF	none	
RMSProp	1	LDP	2	5,0164
		LDF	0,99	
	2	LDP	5	2,8642
		LDF	0,89	
	3	LDP	10	3,2396
		LDF	0,5	
	4	LDP	none	2,9462
		LDF	none	
SGDM	1	LDP	2	2,5414
		LDF	0,99	
	2	LDP	5	2,5576
		LDF	0,89	
	3	LDP	10	3,1982
		LDF	0,5	
	4	LDP	none	2,6726
		LDF	none	

The speed prediction RMSE graph of the GRU deep learning network is given in Figure 20. When the graph is analyzed, although the predictions were successful at the starting point, there were shifts over time. However, the prediction outputs are generally in line with GPS speed values.

Finally, GPS data was also employed as the training data set in the GRU deep learning network. The results are given in Table 8.

The speed estimation results with the best RMSE values obtained with the GRU deep learning network are given in Figure 21. While the LSTM deep learning network has speed estimation errors in the initial conditions, the GRU deep learning network eliminates this problem. In general, the GRU deep learning network is more successful than the LSTM and BLSTM deep learning networks in terms of speed estimation.

Table 8. GRU deep network speed estimation results with GPS.

Optimizer	Parameter Status	Parameter	Value	RMSE
Adam	1	LDP	2	0,95824
		LDF	0,99	
	2	LDP	5	0,726
		LDF	0,89	
	3	LDP	10	0,65246
		LDF	0,5	
	4	LDP	none	1,5647
		LDF	none	
RMSProp	1	LDP	2	1,2481
		LDF	0,99	
	2	LDP	5	0,7112
		LDF	0,89	
	3	LDP	10	0,75496
		LDF	0,5	
	4	LDP	none	0,6521
		LDF	none	
SGDM	1	LDP	2	0,32274
		LDF	0,99	
	2	LDP	5	0,35624
		LDF	0,89	
	3	LDP	10	0,56402
		LDF	0,5	
	4	LDP	none	0,35087
		LDF	none	

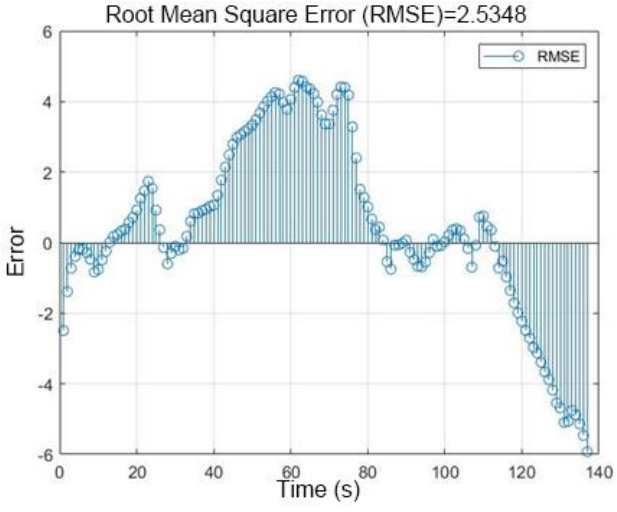

Fig. 20. GRU velocity estimation RMSE values graph.

The RMSE values for speed estimation of the GRU deep learning network is given in Figure 22. When the graph is analyzed, it can be seen that the LSTM and BLSTM deep learning networks have lower RMSE values even though they make incorrect speed predictions at the starting point. The GRU deep learning network obtained the most successful results among the three deep learning networks, even if there are small shifts in the speed transitions, which is a common problem associated with all the three deep learning networks.

All of the utilized deep learning networks overall RMSE values is given in Table 9.

Table 9. Deep learning networks overall RMSE values.

Training Data Set	LSTM	BLSTM	GRU
IMU	2,5547	2,7592	2,5414
GPS	0,42542	1,91122	0,32274

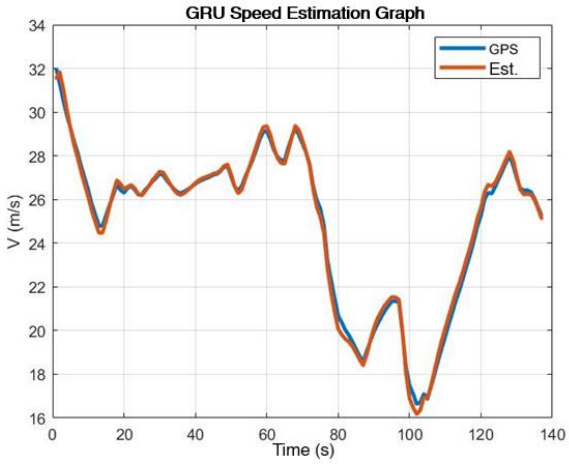

Fig. 21. GRU speed estimation prediction values graph with GPS.

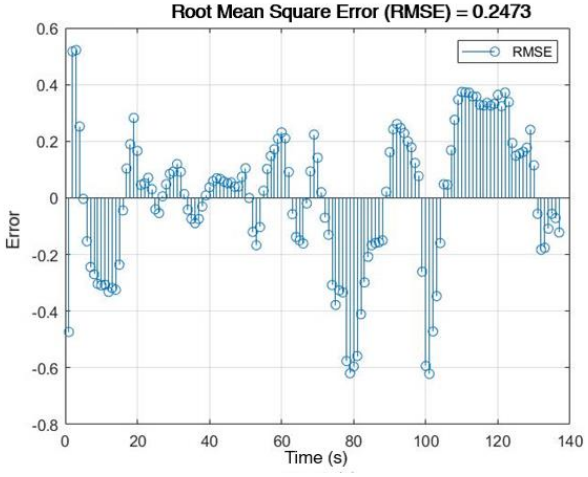

Fig. 22. GRU velocity estimation RMSE values graph.

6. Conclusion

The analysis carried out in this work shows that the Kalman filter, one of the traditional estimation methods, is affected by small errors in the sensor data and these errors results in large shifts in the velocity estimations. In order to prevent these drifts, moving average filtering is used to remove the drift in the acceleration data and make the Kalman filter more successful. LSTM, BLSTM and GRU deep learning networks were used in deep learning-based estimation studies. Deep learning network parameters, which have a direct impact on the prediction accuracy, were tested under different conditions. By using Adam, RMSprop and SGDM optimizers in deep learning networks, it is aimed to examine the effects of optimizers on the speed of estimation results. In this study, two different data sets were used for speed prediction. LSTM, BLSTM and GRU deep learning networks were first trained with IMU sensor data for speed estimation. RMSE values were obtained as 2.5547, 2.7592 and 2.5414, respectively. Furthermore, the same deep learning network methods were trained with GPS data. The prediction data obtained through LSTM, BLSTM and GRU provided RMSE values of 0.42542, 1.91122 and 0.32274, respectively. We see that the prediction with GPS data have higher accuracy since deep learning networks trained with GPS were less affected by noise during the training phase.

Nomenclature

ADAM	Adaptive Moment Estimation
BLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
EKF	Extended Kalman Filter
GRU	Gated Recurrent Unit
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
GWN	Gaussian White Noise
IMU	Inertial Measurement Unit
INS	Inertial Navigation Systems
KF	Kalman Filter
LS	Least Squares
LKF	Linearized Kalman Filter
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
RLG	Ring Laser Gyroscope
RMSE	Root-Mean-Square Error
SGDM	Stochastic Gradient Descent with Momentum
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
ZVO	Zero Velocity Update

Conflict of Interest Statement

The authors declare that there is no conflict of interest in the study.

Credit Author Statement

Murat Üçüncü: Conceptualization, Supervision, Writing - Review& Edit, Literature survey
G. Begüm Cangöz: Conceptualization, Writing-review&editing, Validation, Literature survey
Bekir Göğüş: Data collection, analysis and processing, design, validation, software, Literature survey

7. References

[1] Noureldin A, Karamat TB, Georgy J. Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration; Chapter 4 Inertial Navigation System [Internet]. 2013. 247–271 p. Available from: http://link.springer.com/10.1007/978-3-642-30466-8

[2] Moaveni B, Khosravi Roqaye Abad M, Nasiri S. Vehicle longitudinal velocity estimation during the braking process using unknown input Kalman filter. Veh Syst Dyn [Internet]. 2015 Oct 3;53(10):1373–92. Available from: https://doi.org/10.1080/00423114.2015.1038279

[3] Zhang B, Zhao W, Zou S, Zhang H, Luan Z. Methodology Based on SBI-LSTM During. IEEE Sens J. 2021;21(14):15485–95.

[4] Liu S, Su T, Wang B, Peng S, Jin X, Bai Y, et al. Pedestrian indoor navigation using foot-mounted IMU with multi-sensor data fusion. Int J Model Identif Control. 2018;30(4):261–72.

[5] Svacha J, Loianno G, Kumar V. Inertial Yaw-Independent Velocity and Attitude Estimation for High-Speed Quadrotor Flight. IEEE Robot Autom Lett. 2019;4(2):1109–16.

[6] Wang KL, Xu J. A speed regression using acceleration data in a deep convolutional neural network. IEEE Access. 2019;7:9351–6.

[7] Gençoğlu C, Gümüş H. Standing Handball Throwing Velocity Estimation with a Single Wrist-Mounted Inertial Sensor. Ann Appl Sport Sci. 2020;8(1):2–8.

[8] Jain A, Kulemann D, Schon S. Improved velocity estimation in urban areas using Doppler observations. 2021 Int Conf Localization GNSS, ICL-GNSS 2021 - Proc. 2021;

[9] Yi C, Cho J. Sensor Fusion for Accurate Ego-Motion Estimation in a Moving Platform. Int J Distrib Sens Networks [Internet]. 2015;2015:1–6. Available from: http://www.hindawi.com/journals/ijdsn/2015/831780/

[10] Poulose A, Eyobu OS, Han DS. An Indoor Position-Estimation Algorithm Using Smartphone IMU Sensor Data. IEEE Access [Internet]. 2019;7:11165–77. Available from: https://ieeexplore.ieee.org/document/8606925/

[11] Kim Y, Bang H. Introduction to Kalman Filter and Its Applications. In: Introduction and Implementations of the Kalman Filter [Internet]. IntechOpen; 2019. p. 1–16. Available from: https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications

[12] McGee LA, Schmidt SF. Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry. NASA Tech Memo [Internet]. 1985;(November):21. Available from: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19860003843_1986003843.pdf

[13] Welch G, Bishop G. An Introduction to the Kalman Filter. 1997;

[14] Wang J, Xue M, Culhane R, Diao E, Ding J, Tarokh V. Speech emotion recognition with dual-sequence LSTM architecture. ICASSP 2020 - 2020 IEEE Int Conf Acoust Speech Signal Process. 2020;6469–73.

[15] Wang L, Xu X, Dong H, Gui R, Yang R, Pu F. Exploring convolutional LSTM for polsar image classification. 2018;8452–5.

[16] Zhan Q, Zhang L, Deng H, Xie X. An Improved LSTM For Language Identification. In: 2018 14th IEEE International Conference on Signal Processing (ICSP) [Internet]. IEEE; 2018. p. 609–12. Available from: https://ieeexplore.ieee.org/document/8652445/

[17] Akköse O. Uzun-Kısa Vadeli Bellek(LSTM) [Internet]. 2022. Available from: https://medium.com/deep-learning-turkiye/uzun-kısa-vadeli-bellek-lstm-b018c07174a3

[18] Babüroğlu B, Tekerek A, Tekerek M. Türkçe için Derin Öğrenme Tabanlı Doğal Dil İşleme Modeli Geliştirilmesi. 2019;8.

[19] Yildirim Ö. A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. Comput Biol Med [Internet]. 2018 May;96(March):189–202. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0010482518300738

[20] Kostadinov S. Understanding GRU Networks [Internet]. Available from: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[21] Fu R, Zhang Z, Li L. Using LSTM and GRU neural network methods for traffic flow prediction. In: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC) [Internet]. IEEE; 2016. p. 324–8. Available from: http://ieeexplore.ieee.org/document/7804912/

[22] Yamak PT, Yujian L, Gadosey PK. A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. In: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence [Internet]. New York, NY, USA: ACM; 2019. p. 49–55. Available from: http://dl.acm.org/doi/10.1145/3377713.3377722