Condition-Based LSTM for Residual Calculation of Optical Flow

Mohammad Mahdi Mahdavi[*, 1], Mohammad Mahdi Sakhaee[*, 2], Aria Alasty[*, 3]

[*]Advance Research Lab for Control and Agricultural Robotics (*Sharif AgroLab*),
School of Mechanical Engineering, Sharif University of Technology, Tehran, Iran.
[1] mmmahdavi213@yahoo.com , [2] mahdi.sakhaee78@sharif.edu , [3] aalasti@sharif.edu

Keywords: Optical Flow, LSTM, Extended Kalman Filter, Visual–inertial fusion, Monocular Vision.

Abstract

Accurate state estimation for autonomous unmanned aerial vehicles (UAVs) is essential in GPS-denied environments, such as in indoor tasks. In these situations, Inertial Navigation Systems (INS) perform well in the short term with good accuracy, but they suffer from accumulated error (drift) caused by sensors' noise and bias. Optical flow (OF) is often utilized to correct this drift by estimating lateral velocity from the motion of images; however, OF is degraded by gyroscope errors when the UAV performs aggressive turns or high-acceleration maneuvers. This paper proposes a lightweight, condition-based Long Short-Term Memory (LSTM) neural network to correct these errors. The LSTM model will identify and correct systematic OF velocity errors when the UAV is performing aggressive maneuvers and it is triggered by applying a hysteresis gate that monitors the magnitude of gyroscope readings. This way, the LSTM is only active in scenarios where OF measurements are less reliable. This approach preserves the efficiency of classical methods and prevents any degradation of system performance during stable flight. Several test scenarios were conducted in the Gazebo simulation environment to evaluate the performance of the algorithm. Overall, the INS/OF/LSTM framework reduces trajectory root mean square error (RMSE) by 63.77 % and velocity RMSE by 28.94 % compared to a typical INS/OF pipeline on average across all test scenarios. This paper demonstrates that condition-based LSTM networks increase the accuracy and reliability of classical optical flow-based velocity estimation used in UAV applications.

1. Introduction

Autonomous unmanned aerial vehicles (UAVs) have become integral in a variety of applications, particularly in environments where GPS signals are weak or unavailable, such as indoors. In these situations, Inertial Navigation Systems (INS) can provide short-term accuracy, but they suffer from drift over time due to sensor errors and biases. To reduce this drift, Optical Flow (OF) methods are commonly used to estimate UAV velocity by analyzing the movement of image features. However, as the gyroscope and accelerometer errors increase during quick turns, velocity estimations from OF become less accurate.

The use of Long Short-Term Memory (LSTM) networks in UAV navigation systems has recently gained attention for improving state estimation in GPS-denied areas. LSTM models can analyze time-series data and capture both fast and slow variations in motion signals through their memory states, making them suitable for learning systematic errors or residuals in sensor readings. For example, Fang et al. (2020) trained an LSTM to generate pseudo-GPS position increments based on recent inertial data, reducing navigational errors during GPS outages by about 95% compared to baseline INS. Similarly Li et al. (2025) introduced a PSO-optimized LSTM fused with a fading-memory adaptive EKF, which achieved over 20% improvement in long GNSS-denied durations.

Taghizadeh and Safabakhsh (2023) proposed a hierarchical attention-based LSTM using IMU data alone, showing a 70% improvement in position and velocity prediction accuracy during long GNSS blackouts. Similarly, Wang et al. (2025) developed a self-attention enhanced LSTM (SALSTM) integrated within an EKF framework for small UAVs, which showed 50% higher accuracy compared to previous integrated navigation systems. Xiaoyu et al. (2024) also combined an LSTM with a Right-Invariant Error-State EKF for fixed-wing UAVs, achieving near drift-free navigation over extended GPS-denied flights.

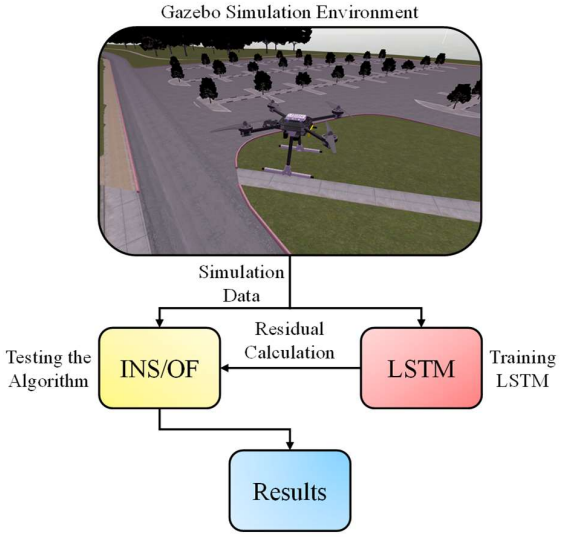

Figure 1. Overall workflow of the method.

Beyond methods in which LSTMs are fused into frameworks, LSTMs have also been used for residual correction. Deraz et al. (2023) trained a five-layer LSTM to learn the velocity error between OF velocity measurements and GPS ground truth, reducing forward velocity error by 54% after 113 seconds of GNSS loss. This highlighted the LSTM's capability to predict residuals and stabilize navigation performance.

These studies show the clear advantage of combining LSTM networks with model-based estimators like the Extended Kalman Filter (EKF). However, most existing works apply the LSTM continuously or focus on general state prediction rather than specifically correcting optical flow degradation during aggressive maneuvers.

This paper introduces a lightweight, condition-based LSTM neural network that learns and predicts systematic OF velocity errors based on recent motion and image features. The LSTM operates only when the UAV performs rapid turns or accelerations. To detect these situations, a hysteresis gate has been implemented which monitors the angular velocity magnitude of the UAV. This ensures the network is active only in situations where OF becomes unreliable, preserving computational efficiency and avoiding unnecessary corrections during stable flight, where the OF performance is accurate. The corrected OF velocities are then fused with the INS using an EKF for improved state estimation (see Figure 1).

The proposed method enhances both velocity and trajectory accuracy while maintaining real-time feasibility. It demonstrates that conditionally activating an LSTM to correct OF residuals significantly improves UAV state estimation in aggressive maneuvers, offering an efficient solution for UAV navigation in GPS-denied environments.

2. Methodology

This section describes the technical aspects of the proposed method. The first part discusses the EKF used in the method. The details of the optical flow are discussed next, and the final part explains the LSTM network and how it is implemented.

2.1 Baseline EKF

In order to fuse the measurements of different sensors, an 11-state EKF has been used. The state vector of the EKF contains the below states:

$$\vec{X} = \left\{X, Y, Z, V_x, V_y, V_z, \phi, \theta, \psi, b_{g_x}, b_{g_y}\right\}^T, \quad (1)$$

where
- X, Y, Z position components,
- V_x, V_y, V_z velocity components,
- ϕ, θ, ψ roll, pitch and yaw angles
- b_{g_x}, b_{g_y} gyroscope biases in x and y direction.

The EKF propagates the state vector during the prediction phase by using accelerometer and gyroscope data. In this procedure, gyroscope measurements are used to estimate attitude angles, while accelerometer values are integrated to determine position and velocity.

During the update phase, measurements from several sensors are used to update the propagating state vector during the prediction phase. Roll (ϕ) and pitch (θ) angles are corrected using accelerometer data, the yaw angle (ψ) is refined using magnetometer readings, the altitude (Z) is adjusted using LiDAR measurements, and the velocity components (V_x, V_y) are corrected using optical flow data (see Figure 2).

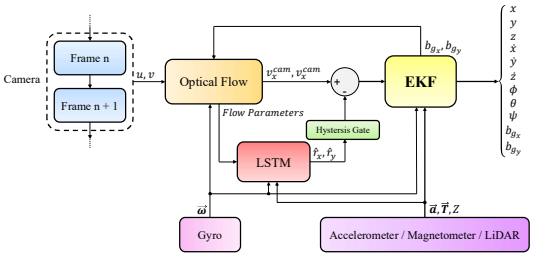

Figure 2. Data flow of the proposed method.

2.2 Sparse Optical Flow

To obtain the velocity of the camera frame from OF, The perspective projection equation for the pinhole camera model is used to determine how a real-world point is projected onto the camera picture plane:

$$s\vec{u} = \boldsymbol{K}\,[\,\boldsymbol{R}\,|\,\vec{T}\,]\,\overrightarrow{X_w} \quad (2)$$

where
- $\vec{u}$ 3×1 homogeneous pixel coordinate,
- $\overrightarrow{X_w}$ 3×1 position vector of a point in world coordinate,
- $\boldsymbol{R}, \vec{T}$ 3×3 rotation matrix and 3×1 translation vector from world coordinate to camera coordiante
- $\boldsymbol{K}$ 3×3 matrix of intrinsic camera parameters
- s scale factor representing depth of $\overrightarrow{X_w}$.

By setting the world coordinate equal to the camera frame, equation (2) reduces to the following equation, where the subscript c in $\overrightarrow{X_c}$ denotes that the position vector is defined with respect to the camera frame:

$$s\vec{u} = \boldsymbol{K}\,\overrightarrow{X_c} \quad (3)$$

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}, \quad (4)$$

where
- c_x, c_y principal point coordinate components,
- f_x, f_y focal length along x and y axis in pixels,

From equation (4) it can be seen that:

$$s = Z_c \quad (5)$$

By calculating the derivative of the equation (3), a relationship between optical flow of a point and its real world velocity with respect to camera frame can be obtained:

$$Z_c\vec{u} = \boldsymbol{K}\,\overrightarrow{X_c}, \quad (6)$$

To calculate the derivative of $\overrightarrow{X_c}$, following equation can be used:

$$\overrightarrow{X_c} = \vec{V} + \vec{\omega} \times \overrightarrow{X_c}, \quad (7)$$

where
- $\vec{V}$ 3×1 translational velocity vector of the point $\overrightarrow{X_c}$,
- $\vec{\omega}$ 3×1 angular velocity vector of the point $\overrightarrow{X_c}$.

If the chosen point is at a fixed position, the velocity vector of the camera can be calculated with respect to the fixed point by taking the opposite of equation (7). This resulting velocity vector is the velocity of the camera frame with respect to any fixed coordinate. To calculate the velocity components (v_x, v_y) from the optical flow, equation (7) can be substituted into equation (6). After decomposing vectors into their components and rewriting equation (6), optical flow components can be calculated as follows:

$$\dot{u} = \dot{u}_{trans} + \dot{u}_{rot} \tag{8}$$

$$\dot{v} = \dot{v}_{trans} + \dot{v}_{rot} \tag{9}$$

Where translational and rotational components of optical flow can be calculated from the following equations:

$$\dot{u}_{trans} = f_x \left(-\frac{V_x}{Z_c} + \frac{u - c_x}{f_x}\frac{V_z}{Z_c} \right) \tag{10}$$

$$\dot{u}_{rot} = f_x \left(\omega_x \frac{(u - c_x)(v - c_y)}{f_x f_y} - \omega_y \left(1 + \left(\frac{u - c_x}{f_x} \right)^2 \right) + \omega_z \frac{v - c_y}{f_y} \right) \tag{11}$$

$$\dot{v}_{trans} = f_y \left(-\frac{V_y}{Z_c} + \frac{u - c_y}{f_y}\frac{V_z}{Z_c} \right) \tag{12}$$

$$\dot{v}_{rot} = f_y \left(\omega_x \left(1 + \left(\frac{v - c_y}{f_y} \right)^2 \right) - \omega_y \frac{(u - c_x)(v - c_y)}{f_x f_y} + \omega_z \frac{u - c_x}{f_x} \right) \tag{13}$$

Since a single camera cannot directly measure the depth of each point in the image, a flat ground plane has been assumed to calculate Z_c. This assumption holds true for high-altitude flights and many indoor flight scenarios. With this assumption, the depth of each feature can be calculated using the known altitude h, which is obtained from the LiDAR sensor and the UAV's orientation (roll and pitch):

$$h = z_b \cos(\theta) \cos(\phi), \tag{14}$$

where h height of the camera relative to ground,
z_b measured depth from the LiDAR.

In order to calculate the depth of a detected feature, the unit direction vector of a light ray corresponding to that feature must be calculated. To do so, normalized coordinates of that feature in the image sensor must be obtained from the following equation:

$$x = \frac{u - c_x}{f_x}, y = \frac{v - c_y}{f_y} \tag{15}$$

where x, y normalized coordinates.

Then the unit direction vector of a light ray can be defined as:

$$\overrightarrow{d_c} = \frac{1}{\sqrt{1 + x^2 + y^2}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \tag{16}$$

By representing $\overrightarrow{d_c}$ in world coordinates, the light ray that connects the detected feature to the camera can be parameterized as follows:

$$\vec{P}(t) = \overrightarrow{C_w} + t\overrightarrow{d_w} \tag{17}$$

where $\overrightarrow{C_w}$ position of the camera frame origin with respect to world coordinate,
$\overrightarrow{d_w}$ $\overrightarrow{d_c}$ representation in world coordinate.

Then the length of the line that connects the detected feature to the camera can be calculated by setting the third component of $\vec{P}(t)$ to zero and solving for t. It should be noted that if $h = 0$ for the ground is assumed, the height of camera frame origin will be equal to h, no matter the xy position of the origin.

$$t = -h/d_{z_w} \tag{18}$$

By multiplying the calculated length t with the z component of the unit direction vector for a detected feature in camera coordinate, Z_c can be obtained from the following equation:

$$Z_c = -\frac{h}{d_{z_w}} u_z \tag{19}$$

where u_z z component of $\overrightarrow{d_c}$,
d_{z_w} z component of $\overrightarrow{d_w}$.

By measuring optical flow vector $(\vec{u})$ for detected features between two consecutive frames, reading $\vec{\omega}$ from gyroscope and calculating Z_c from equation (19), V_x, V_y can be calculated for each feature from equations (8) and (9).

The FAST method has been chosen to detect features in images due to low computational costs and high accuracy (Rosten and Drummond, 2006). To track detected features in consecutive frames and measure the optical flow, pyramidal Lucas-Kanade is used (Bouguet, 2001). This choice makes the tracking algorithm robust against aggressive maneuvers where the features have large displacements between two frames.

The maximum number of detected features is hard-capped at 50 and redetection takes place where the number of tracked features drops below 10. Finally, the median is taken across all calculated velocities from each detected feature to obtain a robust velocity estimation.

2.3 LSTM Residual Correction

Since optical flow depends on gyroscope readings to estimate velocity, its performance degrades when gyroscope errors increase during aggressive turns or high-acceleration maneuvers. Because all sensor measurements are sequential and in time series format, these systematic errors can be identified and corrected using an LSTM network, which takes advantage of temporal dependencies in the data.

The proposed LSTM block contains a minimal design stateless LSTM, which predicts the systematic OF residual from an input feature vector. Stateless models are better for sequences that have short-term patterns and don't need information from states far away, whereas stateful models perform better for sequences that have long-term dependencies, like text or speech. For the case of optical flow error estimation, a stateless LSTM is preferred because these errors are mostly based on recent motion patterns, not long-term trends. Limiting the hidden state to a fixed sequence length forces the model to focus on short-term dynamics. By doing so, the model will be more accurate and robust.

The LSTM model contains 2 layers with 128 hidden units and has a sequence length of $T = 10$. The last hidden state then passes through a small head that contains 2 fully connected layers, each with 128 units and the ReLU activation function, to predict the 2×1 residual vector (see Figure 3). The input feature set has 20 states that include:

$$\vec{x}_i = \left\{\overrightarrow{v^{cam}}, v_z, \vec{\Psi}, z, \vec{\omega}, \vec{a}, \Delta t, \overrightarrow{u_{mean}}, \overrightarrow{u_{rot}}, |\vec{u}|_{RMS}, n\right\}^T, \quad (20)$$

where
- $\overrightarrow{v^{cam}}$ 2×1 velocity vector estimated by OF,
- v_z velocity in the z axis from EKF state vector,
- $\vec{\Psi}$ attitude vector,
- z altitude,
- $\vec{\omega}, \vec{a}$ gyroscope and accelerometer outputs,
- Δt time interval,
- $\overrightarrow{u_{avg}}$ average OF vector,
- $\overrightarrow{u_{rot}}$ average OF vector caused by rotation,
- $|\vec{u}|_{RMS}$ RMS of OF vectors,
- n No. of detected features.

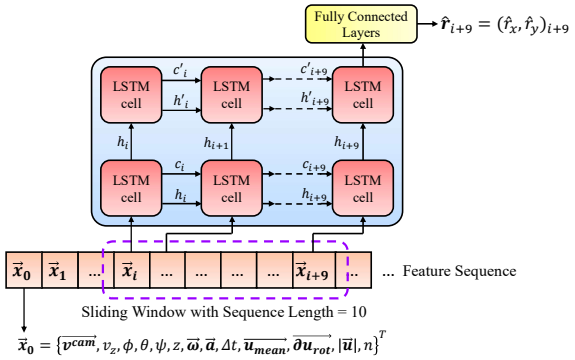

Figure 3. Architecture of the LSTM block.

The hysteresis gate activates the LSTM block based on angular velocity magnitude. Across all scenarios, the flight speed of the UAV and its minimum turning radius were set to 3 m/s and 3 m, respectively. Under these conditions, the hysteresis gate can be defined as a constant angular velocity magnitude that can accurately identify aggressive maneuvers. Based on empirical analysis of angular velocity magnitudes in several different scenarios, the gate thresholds were set to 20 deg/s (ON) and 10 deg/s (OFF).

The model was trained on 90 minutes of feature-rich flight data in Gazebo, including hover states and diverse turning maneuvers for strong generalization. The recorded flight logs were split 70%-15%-15% into train-validation-test datasets. In order to select a valid sequence length, a model with a sequence length of 30 was trained during 60 epochs. To analyze the importance of timesteps in the input, each timestep was set to 0, and the increase in loss has been measured (see Figure 4).

It can be seen from Figure 4 that the hidden state of the model is not dependent on the data earlier than the most recent 10 time steps. Because of this result, the sequence length of the model was set to 10. This decision makes the model more optimized and reduces the average model inference time from 7.93 ms to 5.81 ms, without reducing its accuracy (see Figure 5).

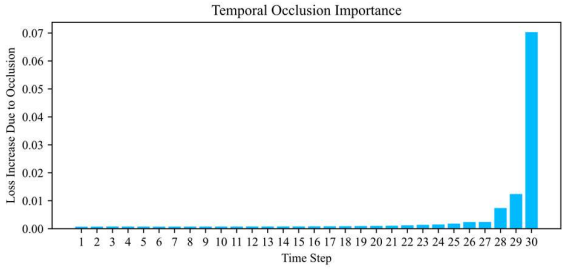

Figure 4. Temporal occlusion importance of the LSTM model with sequence length of 30.

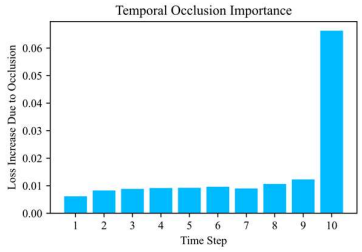

Figure 5. Temporal occlusion importance of the LSTM model with sequence length of 10

The training and validation loss of the model have been plotted in Figure 6. The illustrated plot shows that the loss of the model for training and validation reduces concurrently, and the gap between them is negligible; thus, the model has not overfitted to the training data.

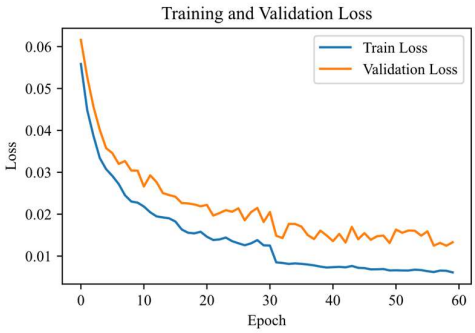

Figure 6. Loss of the model for training and validation datasets.

The R^2 accuracy of the model for training, validation, and test datasets has been plotted in Figure 7.

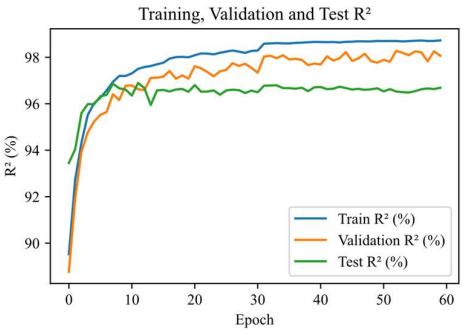

Figure 7. Accuracy of the model for training, validation and test datasets.

3. Experimental Simulation

All training and test flight logs were gathered in the Gazebo simulation environment using PX4 Autopilot. The sensor models were modified to represent their real-world characteristics. The simulated UAV was fitted with a downward-facing monocular camera that has 800×600 resolution, 3 mm focal length, and a 62.2° field of view that matches the specifications of the Raspberry Pi 5 V2 Camera Module. The IMU noise parameters were fine-tuned to replicate the MPU9250, based on previous work done to simulate this sensor (Sakhaee, 2022). The range sensor was modelled as a single-ray LiDAR operating at 20 Hz to mimic an ultrasonic altimeter.

The EKF prediction runs at a frequency of 250 Hz, matching the sampling rates of both the accelerometer and gyroscope. In the update phase, the yaw estimate is adjusted using magnetometer data at a rate of 100 Hz. The OF front end operates at 30 Hz, with each OF step taking an average of 7.39 ms. Additionally, LSTM inference requires 5.81 ms, which keeps the total latency for OF and LSTM under 33 ms, allowing the 30 Hz OF rate to be maintained. All timing measurements were recorded on a 2.3 GHz laptop-class CPU (Intel i7-12700h).

Four metrics have been used for evaluation: RMSE values for v_x, v_y, $|v_{xy}|$ and xy trajectory, which are calculated after start-point alignment. Both INS/OF and INS/OF/LSTM were compared with the same EKF and OF front-end settings.

4. Results and Discussion

In order to evaluate the algorithm, multiple test scenarios have been considered. First scenario is a hover situation that verifies the claim regarding the necessity of using a Conditional LSTM. It can be seen that constant usage of LSTM in certain scenarios (especially the hover scenario) degrades the performance of the algorithm (see Figure 8). The reason is that the LSTM model was constantly producing a small residual and injecting it into the EKF; creating extra noise in the velocity and bias estimation.

In the second scenario including consecutive 90-degree turns, the standard INS/OF algorithm accumulates error at each turn, whereas the LSTM model, accurately compensates for these errors, leading to significantly more precise velocity and trajectory estimates (see Figure 9). Furthermore, by examining the velocity plots, it is evident that the base algorithm has random spikes that introduce errors (mostly in turns), while the proposed method removes these biases (see Figure 10 and Figure 11).

Beyond these two scenarios, the proposed approach has been evaluated on other flight scenarios involving more aggressive maneuvers, mixed sequences of turns, and circular trajectories made of multiple small corrective turns. In all cases, the conditional LSTM showed consistent improvements in velocity and trajectory accuracy over the base INS/OF pipeline (see Figure 12 to Figure 21). Detailed comparisons for all scenarios are shown in Table 1.

Scenario Name/No.		$\sigma_{v_x}\ (m/s)$	$\sigma_{v_y}\ (m/s)$	$\sigma_{\lvert v_{xy}\rvert}\ (m/s)$	$\sigma_{xy}\ (m)$
Hover	Uncond. LSTM	0.0200	0.0141	0.0183	0.465
	Cond. LSTM	**0.0163**	**0.0083**	**0.0124**	**0.188**
Improvement		18.50 %	41.13 %	32.24 %	59.57 %
1	No LSTM	0.2616	0.3221	0.1971	3.826
	LSTM	**0.0824**	**0.0966**	**0.0802**	**0.856**
Improvement		68.50 %	70.01 %	59.31 %	77.63 %
2	No LSTM	0.3027	0.1595	0.1558	2.039
	LSTM	**0.1915**	**0.1028**	**0.1362**	**0.689**
Improvement		36.73 %	35.55 %	12.58 %	66.21 %
3	No LSTM	0.1945	0.2324	0.1251	4.066
	LSTM	**0.0954**	**0.1134**	**0.0974**	**1.993**
Improvement		50.95 %	51.20 %	22.14 %	50.98 %
4	No LSTM	0.1735	0.2089	0.1176	1.572
	LSTM	**0.0737**	**0.1090**	**0.0781**	**0.639**
Improvement		57.57 %	47.82 %	33.59 %	59.35 %
5	No LSTM	0.1642	0.2316	0.1012	2.505
	LSTM	**0.0867**	**0.1148**	**0.0675**	**0.774**
Improvement		47.20 %	50.43 %	33.3 %	69.10 %
6	No LSTM	0.2121	0.2391	0.1339	2.721
	LSTM	**0.1808**	**0.1531**	**0.1169**	**1.106**
Improvement		14.75 %	35.97 %	12.70 %	59.35 %

Table 1. RMSE values of v_x, v_y, $|v_{xy}|$ and xy compared to ground truth.

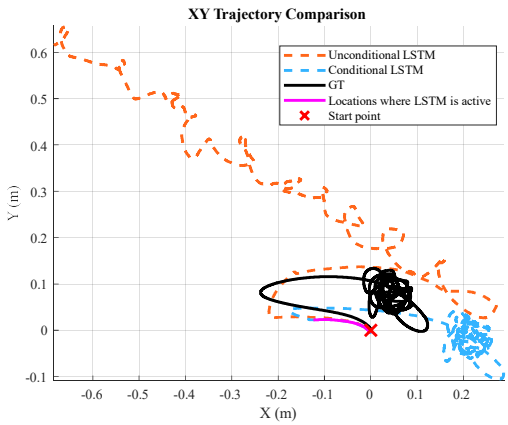

Figure 8. Estimated trajectory in hover scenario.

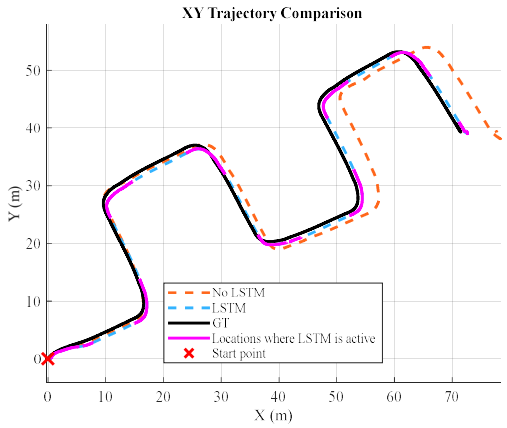

Figure 9. Estimated trajectory in scenario 1.

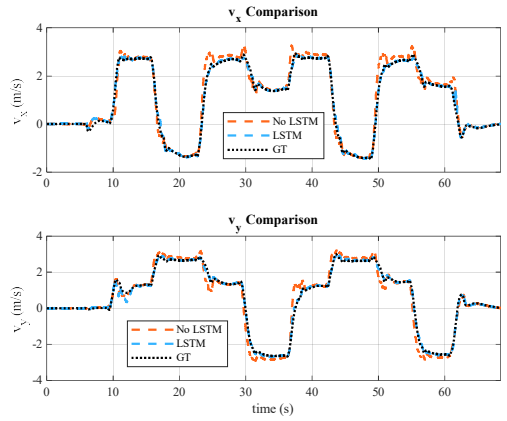

Figure 10. Estimated velocity in scenario 1.

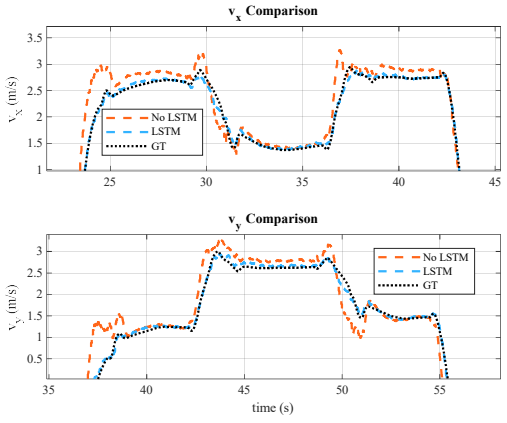

Figure 11. Zoomed-in view of Figure 5 highlighting regions affected by optical flow errors during turns. The proposed method successfully corrects these errors.

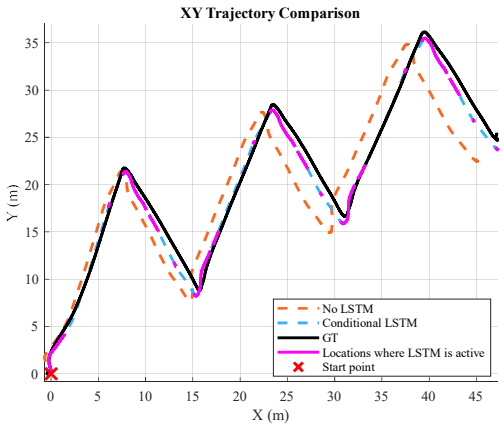

Figure 12. Estimated trajectory in scenario 2.

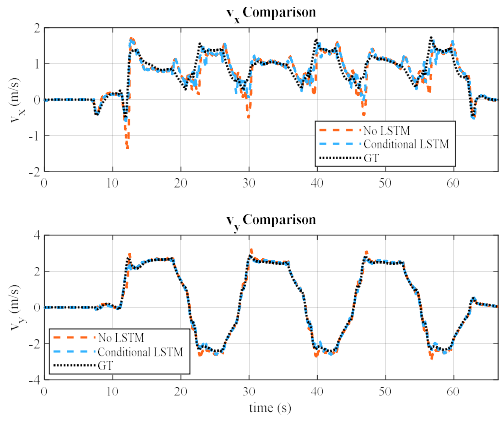

Figure 13. Estimated velocity in scenario 2.

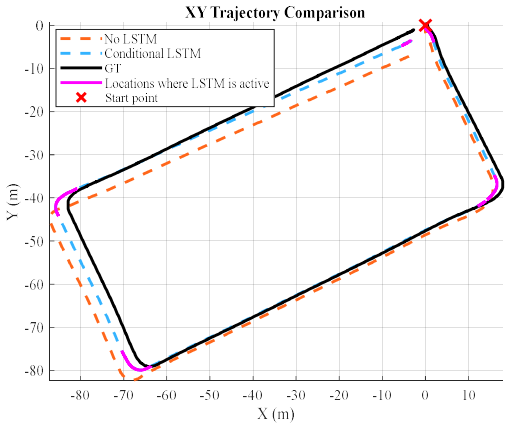

Figure 14. Estimated trajectory in scenario 3.

https://doi.org/10.5194/isprs-archives-XLVIII-4-W22-2025-37-2026

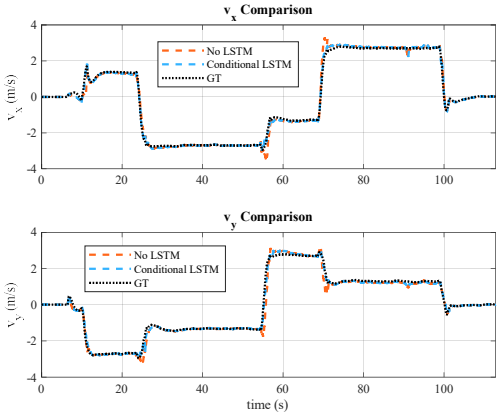

Figure 15. Estimated velocity in scenario 3.

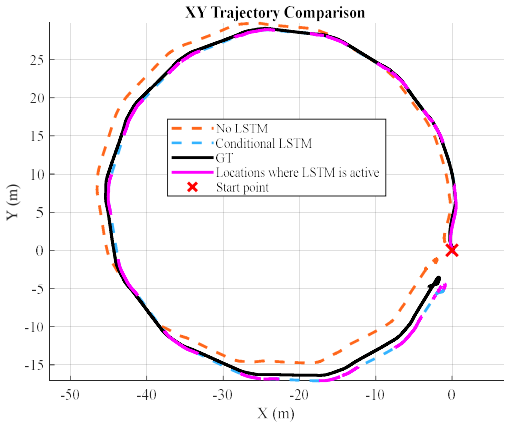

Figure 16. Estimated trajectory in scenario 4.

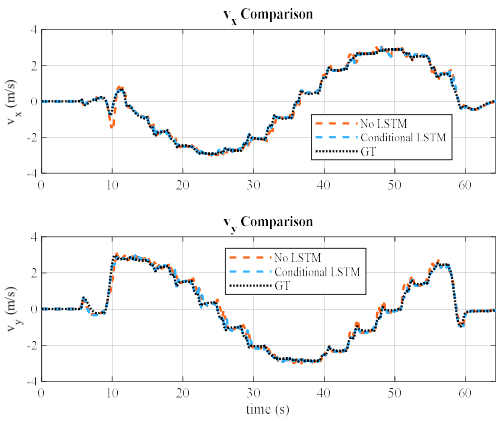

Figure 17. Estimated velocity in scenario 4.

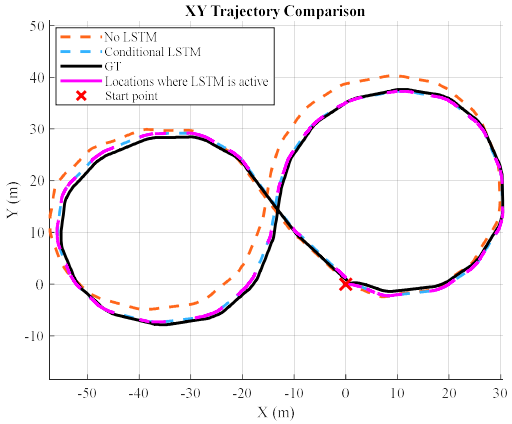

Figure 18. Estimated trajectory in scenario 5.

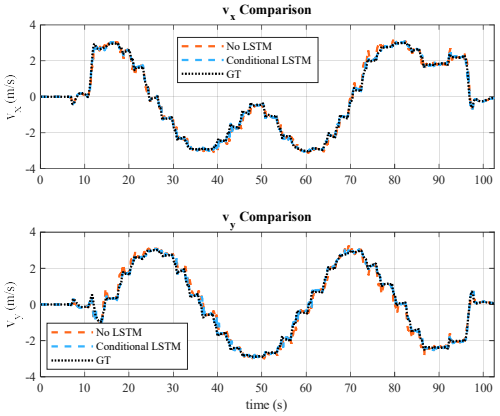

Figure 19. Estimated velocity in scenario 5.

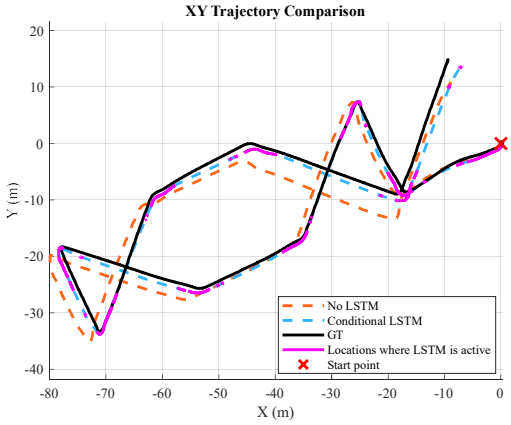

Figure 20. Estimated trajectory in scenario 6.

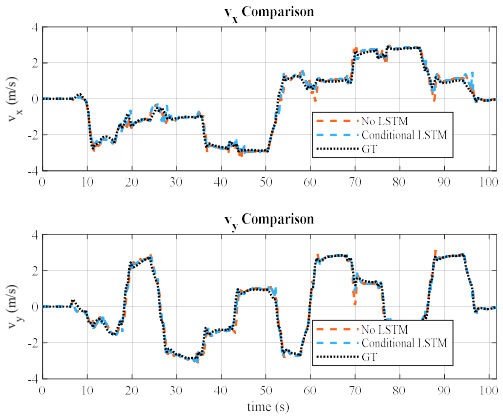

Figure 21. Estimated velocity in scenario 6.

5. Conclusions

Aiming to increase the accuracy of the OF algorithm and its robustness to gyroscope error, this study proposed a condition-based LSTM model to calculate the residual of OF during aggressive maneuvers to reduce the effects of gyroscope errors in velocity calculation. By leveraging a simple hysteresis to activate the LSTM block, the OF algorithm is left untouched during stable flights, where its raw performance is accurate. This way the proposed method is more efficient and also better suited for scenarios like hover, where constant usage of the LSTM block immensely degrades the performance. Results showed that the proposed method greatly increases velocity and pose estimation accuracy. Also, the computational load analysis proves the feasibility of real-time implementation of the proposed method.

This work shows the capability of LSTM models in residual calculation of OF and also its potential use for other sensors, where errors are injected into the system in certain scenarios. In these cases, LSTM models can leverage their pattern recognition in time series data to accurately detect and compensate these errors.

For future work, the training dataset of the model can be expanded to include different velocities; in this case, the gate threshold must be defined as a function of the UAV's angular and linear velocity. Also, different architecture for the models can be examined, like using convolutional neural networks and then passing the output to LSTM, for better feature extraction with the goal of residual calculation. Lastly, gathering data from actual flight logs to train LSTM and implementing the algorithm on hardware for real-time application must be examined.

References

Bouguet, J.-Y., 2001: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm, Microprocessor Research Labs, Intel Corporation.

Deraz, A. A., Badawy, O., Elhosseini, M. A., Mostafa, M., Ali, H. A., El-Desouky, A. I., 2023: Deep learning based on LSTM model for enhanced visual odometry navigation system, *Ain Shams Engineering Journal*, 14(8), 102050, 10.1016/j.asej.2022.102050.

Fang, W., Jiang, J., Lu, S., Gong, Y., Tao, Y., Tang, Y., Yan, P., Luo, H., Liu, J., 2020: A LSTM Algorithm Estimating Pseudo Measurements for Aiding INS during GNSS Signal Outages, *Remote Sensing*, 12(2), 256, 10.3390/rs12020256.

Li, X., Wang, X., Pei, C., 2025: Handling method for GPS outages based on PSO-LSTM and fading adaptive Kalman filtering, *Scientific Reports*, 15(1), 11817, 10.1038/s41598-025-95716-1.

Rosten, E. Drummond, T., 2006: Machine Learning for High-Speed Corner Detection, *Computer Vision – ECCV 2006*, Berlin, Heidelberg, 430–443, 10.1007/11744023_34.

Taghizadeh, S., Safabakhsh, R., 2023: An integrated INS/GNSS system with an attention-based hierarchical LSTM during GNSS outage, *GPS Solutions*, 27(2), 71, 10.1007/s10291-023-01412-w.

Wang, Z., Shen, X., Li, J., Li, J., Wu, X., Yang, Y., 2025: Enhancing Integrated Navigation with a Self-Attention LSTM Hybrid Network for UAVs in GNSS-Denied Environments, *Drones*, 9(4), 279, 10.3390/drones9040279.

Xiaoyu, Y., Fujun, S., Zongyu, Z., Rui, Z., Qinghua, Z., 2024: Semi-Aerodynamic Model-Aided Invariant Kalman Filtering for UAV Full-State Estimation, *IEEE Sensors Journal*, 24(16), 25920–25939, 10.1109/JSEN.2024.3414995.