Optical Flow for Drone Horizontal Velocity Estimation without GPS

Djedjiga Belfadel*, David Haessig†

*Electrical Engineering, Fairfield University, CT
Email: dbelfadel@fairfield.edu
†AuresTech Inc., Bridgewater, NJ
Email: dave@aurestech.com

Unmanned aerial vehicles (UAVs) are widely used in military and civilian applications, such as surveillance, disaster monitoring, and rescue missions. They rely heavily on global positioning systems (GPS) for navigation. However, the accuracy and availability of GPS positioning, navigation, and timing (PNT) data can suffer losses due to jamming and spoofing.

To address overreliance on GPS infrastructure, this paper presents a vision-aided system to enable UAVs to estimate velocity and continue operation when GPS signal is intermittently lost. The system fuses data from onboard sensors including the inertial measurement unit (IMU), barometer, and optical flow camera using an extended Kalman filter (EKF). The efficiency of this approach was validated through simulation experiments involving a UAV navigating in a circular and square motions. Statistical analysis confirmed the proposed system can provide state estimation and smooth trajectory control for UAVs during GPS signal outages.

***Index Terms*—Navigation, UAVs, IMU, Optical Flow**

I. Introduction

In robotics, localization refers to determining an object's position and attitude in 2D or 3D space. This is crucial, especially in aerospace applications [1]. Using sensors, robots can maintain estimates of their state, including velocity, attitude, and position [2]. Accelerometers within IMUs, which also incorporate gyroscopes, are commonly employed to calculate position by twice integrating acceleration data [3]. IMU sensor readings inherently contain error, such as random noise and sensor bias which negatively impact navigation data [4]. Moreover, numerical integration introduces additional noise through time discretization, further impacting drift [5]. Euler integration, for example, causes local truncation errors like that of Taylor series approximations [6]. Thus, the double integration of accelerometer data to generate velocity and position leads to compounded drift errors [7], posing a significant challenge in autonomous UAV navigation.

Recent experiments have shown the potential of fusing various sensors with IMUs to match the accuracy of IMU/GNSS systems [8]. This includes novel hardware combinations and algorithmic approaches to enhance sensor fusion accuracy in UAVs. Another approach in GPS-denied environments is using multiple UAVs, each with a ranging sensor and an IMU [9]. Drift is corrected by comparing distances from other UAVs in a swarm using trilateration.

Cameras in UAVs can aid in navigation through object recognition and tracking, utilizing landmarks. However, this approach is computationally demanding, and issues like unknown obstacles or lighting changes can cause data inaccuracies. To address these, SLAM (simultaneous localization and mapping) uses multiple cameras and LIDAR sensors to handle environmental lighting changes [10]. SLAM combines multiple sensors to generate a 3D map of the surroundings [11], offering accurate position estimation but at a high computational cost.

Optical flow sensors are increasingly being explored for relative positioning. Effective in short-term stabilization, their applicability for long-term positioning is still under active research [12]. Studies have indicated the potential of optical flow sensors in positional estimation, particularly when fused with other sensor systems [13], [14]. These advancements lay the foundation for the growing significance of optical flow sensors in UAV systems.

This paper presents a vision-aided UAV velocity estimation approach using an EKF to integrate an on-board IMU, altitude sensor, and an optical flow sensor, greatly improving the filter's estimate of platform horizontal velocity during intermittent periods of GPS signal loss. Consequently, the system significantly reduces navigation error relative to that of dead reckoning alone, over brief GPS outages. Our method utilizes pixel displacement measurements from the optical flow sensor to compute horizontal velocity measurements in the platform body frame. These are transformed onto a North-East-Down locally level navigation frame using platform attitude developed using the IMU rate sensor, accelerometer, and magnetometer data. Velocity measurements in the NED frame are then used in the EKF's estimation of UAV horizontal velocity. To evaluate the effectiveness of this approach for this aided inertial navigation, we applied it to several simulated scenarios. The trajectories evaluated in the simulation include a circular trajectory, and square paths. We assess estimation performance by calculating the error in the estimated platform location and velocity. Our results indicate that the proposed

design reduces state estimation error and improves UAV trajectory control during GPS signal outages.

Section II presents the problem formulation and Section III the solution in detail. Section IV describes the simulations performed and the results. Finally, Section V gives the conclusions and future work.

II. Problem Formulation

A. Coordinate system

The coordinate system used in this paper is a North-East-Down (NED) system, which serves as the navigation frame (N-frame). The NED system has an origin at a fixed location on the surface of the Earth. The N-frame is located near the region in which the drone will operate, with the x-axis pointing towards geographic north, the y-axis pointing towards the east, and the z-axis pointing downwards towards the center of the Earth. The body frame (B-frame), on the other hand, is a reference frame attached to the UAV itself, with the x-axis pointing forwards along the body roll axis, which is aligned normally with the direction of the flight, the y-axis pointing to the right side of the drone platform, and the z-axis pointing downwards.

To estimate the UAV's state, a coordinate transformation is required between the N and B frames. This transformation relies on an attitude matrix, which plays a critical role in transforming vectors measured in the B-frame to the N-frame, where the navigation solution is generated. The attitude matrix is typically obtained from Euler angles, specifically roll, pitch, and yaw, provided by the IMU.

Assuming that the roll angle is denoted by ϕ, the pitch angle by θ, and the yaw angle by ψ, the attitude matrix can be computed as a product of three rotation matrices:

$$C = \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi) \tag{1}$$

where $\mathbf{R}_z(\psi)$, $\mathbf{R}_y(\theta)$, and $\mathbf{R}_x(\phi)$ represent the rotation matrices about the z-axis, y-axis, and x-axis, respectively.

The rotation matrices can be expressed as:

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3}$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{4}$$

The matrices $\mathbf{R}_z(\psi)$, $\mathbf{R}_y(\theta)$, and $\mathbf{R}_x(\phi)$ are orthogonal matrices, hence C_{BN} is orthogonal and has the property:

$$C_{NB} = C_{BN}^{-1} = C_{BN}' \tag{5}$$

If $\boldsymbol{a}^B$ is the acceleration vector provided by the IMU and expressed in the B-frame, then the corresponding acceleration vector in the N-frame, corrected for gravity and denoted as $\boldsymbol{a}^N$, can be obtained by transforming $\boldsymbol{a}^B$ using the attitude matrix $\mathbf{C}$. This transformation is essential for accurately estimating the UAV's state during navigation.

$$a^N = C \times a^B + g + a_n \tag{6}$$

where:

- a^N is the acceleration in the navigation frame,
- a^B is the acceleration measured in the body frame,
- $C = C_{NB}$ is the rotation matrix from the body frame to the navigation frame,
- g is the acceleration due to gravity, considered constant and acting in the negative direction in the navigation frame, and
- a_n is the noise in the acceleration measurement.

B. Measurement Model

1) Inertial Measurement Units. An Inertial Measurement Unit (IMU) measures the linear accelerations and angular rotational rates that occur along a coordinate frame that is attached to the IMU body. This body frame, like the B-frame described earlier, rotates with the IMU itself. Since the IMU senses 3 axes of rate and acceleration it is referred to as a 6-axis sensor. The IMU can be used for dead-reckoning navigation, a process depicted in Figure 1. Dead-reckoning generates an object's position, velocity, and orientation by integrating the IMU outputs over time. To perform this integration, the objects initial position, velocity, and orientation must be known. Any error present in these initial conditions will introduce error in the resulting solution. Noise present in the IMU outputs will also add error to the integration results, introducing what has been defined as Random Walk [15].

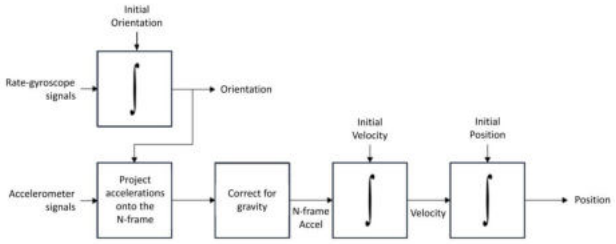

Fig. 1: Basic block diagram for a Strap-down inertial navigation system [15]

.

To constrain the attitude drift errors associated with dead-reckoning, a magnetometer is often added to the basic IMU. The IMU and magnetometer data are combined often using an EKF, resulting in a device designated an Attitude and Heading Reference System (AHRS). The magnetometer provides a measure of compass angle to avoid heading drift. The accelerometer data includes the measured force and vertical direction of gravity. This gravity vector is aligned nominally to the N-frame z-axis and enables drift in platform roll and pitch attitudes to be constrained. The magnetometer measures the 3 components of the magnetic field and thus this type of

IMU-magnetometer device is referred to as a 9-axis sensor. A baro-altimeter is often also added to provide a measure of altitude and thereby constrain altitude drift error. Products that combine an IMU, magnetometer, and baro-altimeter are designated 10-axis sensors.

The VN-100, manufactured by VectorNav Technologies, is an example of an AHRS that is commonly used in applications such as UAVs, drones, and robotics. The VN-100's compact size and low power consumption make it an attractive option for use in small or portable devices where space and power are limited.

Linear accelerations measured by the IMU in the B-frame must be transformed into the navigation frame, corrected for gravity, and integrated to produce platform velocity and position in the N-frame. As noted, IMU sensor noise causes the dead-reckoning navigation solution to deviate from the correct solution with a random walk type of error that grows with time and is unbounded. To constraint the growth of these drift errors, an aided inertial navigation algorithm will utilize measurement data provided by additional sensors to improve or aid the solution produced by the IMU alone. Sensor data may include, for example, optical flow, measurement of platform ground speed, air speed, altitude above ground, range to a specific location, etc.

An aided inertial navigation mechanization in discrete time form is shown in Figure 2. Inputs include those from the IMU that provides the incremental change in body angle, $\Delta\theta$, and in velocity, ΔV. Both are 3 elements vectors containing the change that has occurred along the platform body axes over the most recent time step of duration, δt. Also provided in the case of an AHRS is the platform attitude matrix C of equation 5, shown as $\hat{C}_k$ in the figure.

C. Altitude Sensor

A barometer is used as an altitude sensor to determine the z-coordinate, or vertical position, by measuring atmospheric pressure and applying the principle that pressure decreases with increasing altitude. This approach provides an estimate of altitude above sea level based on the correlation between measured pressure and atmospheric pressure at a reference point, typically using the barometric formula. Altitude is derived by subtracting the barometer's current altitude from the barometric altitude that occurred when on the ground. This method therefore assumes that the ground is level.

D. Optical Flow Sensor Model

The Optical Flow Sensor is an imaging device that generates pixel values from the scene entering the camera, with each pixel represents the angular location of the elements in the scene. By comparing pixel displacement between successive frames of an image, the direction and speed of motion of the image can be determined. This method of calculating motion from frame changes can be achieved with smart cameras along with software libraries, or with embedded light-tracking optic sensors. Unlike software-based optical flow approaches, embedded versions of these sensors work by using a digital signal processor with a lens to detect changes in shadow levels in each pixel.

The Pimoroni PMW3901 optical flow sensor is an example of an embedded light-tracking optical sensor, and gives this collection of pixel displacement values, known as a "flow field", in an averaged x and y direction designated "pixel vectors". From the sensor data sheet we know that the sensor has a Field of View (FOV) of 42 degrees, (±21 degrees). From a series of tests measuring sensor output at varying velocities, it was observed that the PMW3901 must move a minimum of 0.0015 radians to register a new output with a change of 1 pixel count. The flow vectors produced within the sensor in either the x or y direction must separately average to this quantization level to produce a new pixel count of 1.

A key feature of the sensor is that it is not measuring velocity but is measuring a change in the location of the scene. Assume for this discussion that the scene is moving by several pixels in distance from one frame to the next, and that the frame time is 20 msec (i.e.50 Hz). Then the delta-angle output from the sensor will be an integer count value representing the angular rotation measured between the two frame images, quantized to the pixel size. The fractional part of the motion is removed from the output by quantization. However, it is not lost. Consider the case when the motion is very slow so that it takes several frames, e.g. 10, to move by a single pixel. In this case the output will sit at zero (0) counts until the spot finally reaches a distance of one pixel, and at that sample time it will output a change of 1 pixel, then it will drop back to outputting zeros while moving to the next pixel quantum distance. This has a distinct advantage over a sensor that measures angular velocity and outputs a quantized value of that measurement. A velocity sensor could experience an angular rate that remains within one quantum and that motion would never be evident in the sensor's output. But the Optical Flow Sensor instead is sensing angular displacement, or the integral of the angular rate, and producing the change in that angle from frame to frame, so that even the smallest angular rate input is eventually detected when it reaches an angular displacement the size of one pixel. Thus, when the sensor is moving slowly and the scene is moving slowly over the image plane, the sensor produces a stream of 0's punctuated with a periodic value of 1 count that occurs when the accumulated distance traversed reaches a quantization step equaling one pixel size.

The model created to represent the Optical Flow Sensor output is therefore one that involves the integral of the angular rate of the scene. That angular rate equation as defined earlier is:

$$\Omega = -\omega - \frac{V}{A} \tag{7}$$

where Ω is the scene angular velocity, ω is the angular velocity of the sensor body, V is the sensor translational velocity, and A is the sensor distance (or altitude) from the scene being

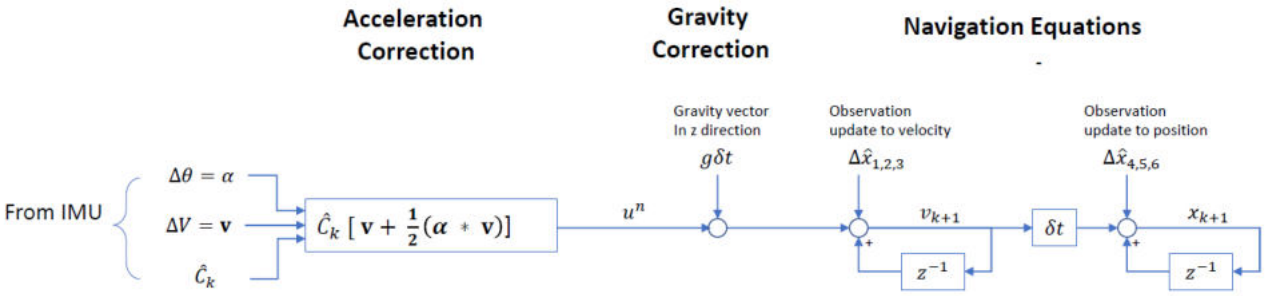

Fig. 2: Aided Navigation Algorithm Block Diagram

observed (i.e. the ground). Integrating this to produce an angle:

$$\phi(t) = \int_{t_0}^{t} (-\omega - \frac{V}{A})\, dt \tag{8}$$

where ω, V, and A are functions of the continuous time variable t. To produce the sensor output we integrate $\phi(t)$ to the frame time kT at frame index k , quantizing to the quantization step size Q, and generate the difference in the quantized result:

$$\phi(kT) = \int_{t_0}^{kT} \left(-\omega - \frac{V}{A}\right) dt \tag{9}$$

$$X_k = \text{floor}\left(\frac{\phi_k}{Q}\right) \tag{10}$$

$$V_{Bx} = X_k - X_{k-1} \tag{11}$$

This value, V_{Bx}, is what comes out of the sensor to represent motion of the scene in one direction. It is an integer value and has units of counts. There is an equivalent equations for V_{By}, representing motion in the other orthogonal direction.

III. Sensor Model for Conversion to Velocity

To convert from the sensor output's change in scene angle V_{Bx} to linear velocity we integrate the scene angle over a single frame time period:

$$\Delta\phi_k = \int_{(k-1)T}^{kT} \left(-\omega - \frac{V}{A}\right) dt \tag{12}$$

To generate this integrated result, we assume that the inputs are constant over the very short time period T:

$$\Delta\phi_k = -\omega_k T - \frac{V_k}{A_k} T \tag{13}$$

The angular rate ω_k is the pitch angular velocity and is provided by the IMU. The altitude A_k is assumed to be known, provided by an altitude sensor which can be a barometric altimeter's measure of the distance above the take-off altitude, assuming that the ground is relatively flat. Equation 11 is converted from units of counts to units of radians (angle) using the quantization step size:

$$\Delta\phi_k = V_{Bx} Q \tag{14}$$

Combining Equations 12 and 13 and solving for velocity V_k:

$$V_k = \left[-\omega_k - V_{Bx}\left(\frac{Q}{T}\right)\right] A_k \tag{15}$$

This is the equation that will be used to produce a velocity measurement that will serve as the feedback signal for velocity control.

IV. The Solution of the Estimation Problem

The EKF is a widely used technique for state estimation in nonlinear systems, making it particularly suitable for drone navigation where the process and measurement models are often nonlinear. The EKF works by linearizing the nonlinear model around the current state estimate and then applying the standard Kalman Filter update equations. The prediction and update steps of the EKF are represented by the following equations:
Prediction step:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \tag{16}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{17}$$

Update step:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \tag{18}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - h(\hat{x}_{k|k-1})) \tag{19}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{20}$$

In these equations, $\hat{x}_{k|k-1}$ and $\hat{x}_{k|k}$ represent the predicted and updated state estimates, respectively, while $P_{k|k-1}$ and $P_{k|k}$ denote the predicted and updated state covariance matrices. The process and measurement models are represented by $f(\cdot)$ and $h(\cdot)$, with F_k and H_k being the corresponding Jacobian matrices. Q_k and R_k are the process and measurement noise covariance matrices, and K_k is the Kalman gain. The system dynamics model and the state transition matrix $\mathbf{F}_k$ are adjusted accordingly:

A. EKF Equations Applied to Vision-Nav

State Vector: The state vector $\mathbf{x}_k$ to be estimated now includes the position, velocity of the drone done in the N-frame i,e. the body Frame, while the orientation is estimated separately using the AHRS:

$$\mathbf{x}_k = \begin{bmatrix} v_N & v_E & v_D & p_N & p_E & p_D \end{bmatrix}_k^T \tag{21}$$

where the NED velocities are designated v_N, v_E, v_D, the NED positions are designated p_N, p_E, p_D.

The optical flow sensor produces measurements ΔX_k and ΔY_k that are associated with the platform x and y body axes. The measurements in the body frame must be related to the states that are given in the N-frame. Equation 15 of velocity in term of ΔX_k and ΔY_k can be used to generate velocity observations for the x and y velocities in the B-frame

$$V_x^B(k) = \left[-\omega_y(k) - V_{Bx}\left(\frac{Q}{T}\right)\right] A_k \tag{22}$$

$$V_y^B(k) = \left[\omega_x(k) - V_{By}\left(\frac{Q}{T}\right)\right] A_k \tag{23}$$

and the take $V_x^B(k)$ and $V_y^B(k)$ as the sensor measurements rather than ΔX_k and ΔY_k. These are related to the state as fellows

$$\begin{bmatrix} v_N \\ v_E \end{bmatrix}_N = \left[C_{NB}\right]_{2\times 2} \begin{bmatrix} V_x \\ V_y \end{bmatrix}_B \tag{24}$$

In this step of the observation generation, we utilize the top-left 2×2 submatrix of C_{NB}, assuming that the platform's roll ϕ and pitch θ angles are negligible and thus are approximated to be zero.

State Dynamic Equations:

$$u^n = C_k(\Delta V_k + \frac{1}{2}[\Delta\theta_k \times \Delta V_k]) \tag{25}$$

$$v_{k+1} = v_k + (u^n + g\delta T) + w_k \tag{26}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \delta T v_k \tag{27}$$

where δT is the time step, u^n is a known input with components provided by the AHRS, b_a is the accelerometer bias and $\boldsymbol{w}_k$ is the process noise, an additive white Gaussian noise sequence. C_k is the attitude matrix generated by the AHRS, $g\delta T$ is the contribution of the gravity vector g over time interval δT, and where ΔV_k and $\Delta\theta_k$ are the velocity and angular increment vectors given by the AHRS at each sample time. The vector u^n is the change in velocity over the interval δT in the N-frame. A description is given in [15].

Observation Equation:

Given the measurement vector $\mathbf{z}_k$ as:

$$\mathbf{z}_k = \begin{bmatrix} v_{N,k} \\ v_{E,k} \\ -p_D \end{bmatrix} \tag{28}$$

where $v_{E,k}$ and $v_{N,k}$ are the North and East components of the velocity as measured by the optical flow sensor, and $p_{D,k}$ is the altitude or Down position from an altitude sensor. The measurement model relates the current state to the measurements:

$$\mathbf{z}_k = h(x_k) + m_k \tag{29}$$

where m_k is the measurement noise, modeled as an additive white Gaussian noise sequence.

Assuming that the UAV is flying over a flat surface, the function h mapping the state vector $\mathbf{x}_k$ to the measurement vector $\mathbf{z}_k$ is defined based on the relationships:

$$h(\mathbf{x}_k) = \begin{bmatrix} \left(-\omega_{yk} + V_{Bx}\left(\frac{Q}{T}\right)\right) p_{D,k} \\ \left(\omega_{xk} + V_{By}\left(\frac{Q}{T}\right)\right) p_{D,k} \\ -p_{D,k} \end{bmatrix} \tag{30}$$

where:

- ω_k represents the angular velocity of the sensor body.
- V_{Bx} and V_{By} are the sensor outputs from the optical flow sensor, representing the change in scene angle in the North and East directions, respectively.
- Q is the quantization step size of the optical flow sensor.
- T is the time between frames.
- $p_{D,k}$ is the altitude of the sensor from the ground, measured as the change in altitude given by the barometer.

For an EKF, the Jacobian of $h(x)$, denoted as H_k, is computed at each step to linearize about the current estimate:

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k-1}} \tag{31}$$

The Jacobian matrix H_k of the measurement model $h(x_k)$ with respect to the state vector x_k is calculated by taking the partial derivatives of each measurement function with respect to each element of the state vector. Given our assumed measurement model, the Jacobian matrix H_k would be:

$$H_k = \begin{bmatrix} \frac{\partial v_{N,k}}{\partial v_{N,k}} & \frac{\partial v_{N,k}}{\partial v_{E,k}} & \frac{\partial v_{N,k}}{\partial v_{D,k}} & \frac{\partial v_{N,k}}{\partial p_{N,k}} & \frac{\partial v_{N,k}}{\partial p_{E,k}} & \frac{\partial v_{N,k}}{\partial p_{D,k}} \\ \frac{\partial v_{E,k}}{\partial v_{N,k}} & \frac{\partial v_{E,k}}{\partial v_{E,k}} & \frac{\partial v_{E,k}}{\partial v_{D,k}} & \frac{\partial v_{E,k}}{\partial p_{N,k}} & \frac{\partial v_{E,k}}{\partial p_{E,k}} & \frac{\partial v_{E,k}}{\partial p_{D,k}} \\ \frac{\partial p_{D,k}}{\partial v_{N,k}} & \frac{\partial p_{D,k}}{\partial v_{E,k}} & \frac{\partial p_{D,k}}{\partial v_{D,k}} & \frac{\partial p_{D,k}}{\partial p_{N,k}} & \frac{\partial p_{D,k}}{\partial p_{E,k}} & \frac{\partial p_{D,k}}{\partial p_{D,k}} \end{bmatrix} \tag{32}$$

Given our direct mapping from the state vector to the measurement vector, the derivatives simplify to:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \frac{\partial v_{N,k}}{\partial p_{D,k}} \\ 0 & 1 & 0 & 0 & 0 & \frac{\partial v_{E,k}}{\partial p_{D,k}} \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{33}$$

where

$$\frac{\partial v_{N,k}}{\partial p_{D,k}} = -\omega_{yk} + V_{Bx} \tag{34}$$

$$\frac{\partial v_{E,k}}{\partial p_{D,k}} = \omega_{xk} + V_{By} \tag{35}$$

$$\tag{36}$$

The measurement noise covariance matrix R_k, representing uncertainty in the measurements, is defined as:

$$R_k = \begin{bmatrix} \sigma^2_{v_{N,k}} & 0 & 0 \\ 0 & \sigma^2_{v_{E,k}} & 0 \\ 0 & 0 & \sigma^2_{P_{D,k}} \end{bmatrix}$$

The Kalman Gain K_k is calculated as:

$$K_k = P_{k|k-1}H_k^T(H_k P_{k|k-1} H_k^T + R_k)^{-1}$$

The updated state estimate $\hat{x}_{k|k}$ with the incorporation of the measurement z_k is:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \left(z_k - h(\hat{x}_{k|k-1})\right)$$

And the updated covariance estimate $P_{k|k}$ is:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

EKF Predict Step:

$$u_k = [(\boldsymbol{u}^n + g\delta T)' \;\; \mathbf{0}_{1\times 3}]' \tag{37}$$

$$\bar{\mathbf{x}}_{k+1} = F\bar{\mathbf{x}}_k + u_k \quad \text{(State Update)} \tag{38}$$

$$\bar{P}_{k+1} = F\bar{P}_k F' + Q_k \quad \text{(Covariance Update)} \tag{39}$$

where:

$$F = \begin{bmatrix} I_3 & 0_{3\times 3} \\ \delta T I_3 & I_3 \end{bmatrix} \tag{40}$$

$$Q_k = \begin{bmatrix} Q_v & 0_{3\times 3} \\ 0_{3\times 3} & Q_p \end{bmatrix} \tag{41}$$

with I_3 representing the 3x3 identity matrix, $0_{3\times 3}$ representing the 3x3 zero matrix, and Q_v, and Q_p being the velocity and location process noise spectral density matrices respectively.

EKF Update Step:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -\omega_{xk} + V_{By}\left(\frac{Q}{T}\right) \\ 0 & 1 & 0 & 0 & 0 & \omega_{yk} + V_{Bx}\left(\frac{Q}{T}\right) \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{42}$$

$$r_k = z_k - h(\bar{x}_k, X) \tag{43}$$

$$S_k = H_k \bar{P}_k H_k' + M_k \tag{44}$$

$$K_k = \bar{P}_k H_k' S_k^{-1} \tag{45}$$

$$\hat{P}_k = \bar{P}_k + K_k r_k \tag{46}$$

$$\hat{P}_k = (I - K_k H_k)\bar{P}_k \tag{47}$$

$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + K_k r_k \tag{48}$$

Note that the observation update can occur at a lower sample rate than the predict step sample rate, which is equal to that of the IMU sample rate. The measurement update can occur irregularly, whenever a new measurement is available. At the output of the measurement update function, the current state entering the next predict step process will be set equal to the state estimate after the measurement update:

$$\bar{\mathbf{x}}_k = \hat{\mathbf{x}}_k \tag{49}$$

V. Simulations

Simulation tests were performed to assess the effectiveness of a horizontal velocity estimation system for a drone, integrating the Optical Flow Sensor and an IMU. In the evaluation process, we conducted simulations involving a drone moving through various motion patterns: circular and square. Our method estimated the states (position and velocity) of the drone and the accelerometers biases. We are using an AHRS which provides platform orientation matrix C for use in the navigator. For the IMU section of the simulation, we set the IMU measurement noise standard deviation σ_{gyro} to 610 μrad/sec, σ_{acc} to 0.014 m/s^2, and the optical flow sensor measurement noise standard deviation σ_{OF} to 0.02 m/s^2. The IMU functions at a frequency of 50 Hz, whereas the optical flow sensor operates at a frequency of 10 Hz. The results consistently demonstrated the efficacy of the Vision-Nav method in reducing velocity and position errors compared to dead reckoning alone.

A. Description of the Scenarios

1) Circular Motion Scenario. In the circular motion scenario the drone is flying in a circular motion for 100 seconds. The comparison between true North-East (N-E) locations and the estimated N-E locations using the EKF is shown in Figure 8. This comparison highlights the accuracy of the EKF in estimating the drone's trajectory.

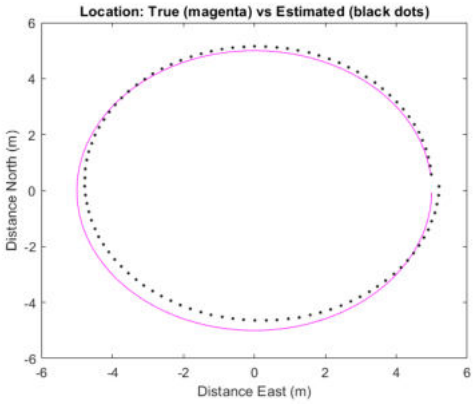

Fig. 3: Comparison of True N-E Locations and Estimated N-E Locations.

In Figure 4 and Figure 5, the position and velocity errors of the drone are illustrated using the EKF. The EKF significantly reduces these errors, demonstrating its effectiveness in managing the nonlinearities and uncertainties associated with the drone's motion.

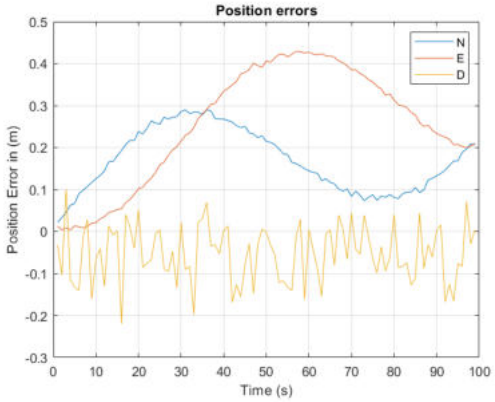

Fig. 4: Circular Motion EKF Position Estimation Errors.

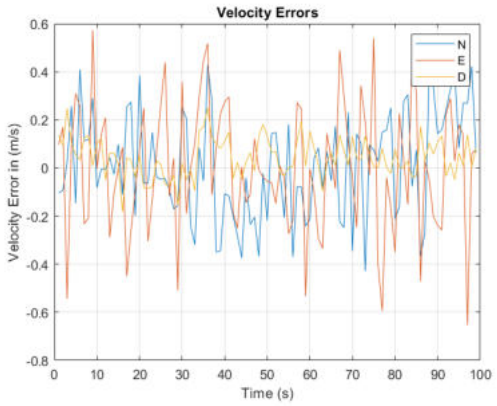

Fig. 5: Circular Motion EKF Velocity Estimation Errors.

Figures 6 and 7 illustrate the position and velocity errors associated with the Dead-reckoning method. The errors observed in Dead-reckoning are significantly higher compared to those seen with the EKF method. This highlights the substantial improvement in navigation accuracy achieved by using the EKF.

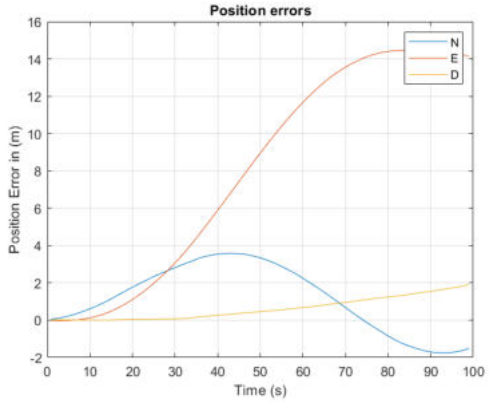

Fig. 6: Circular Motion Dead-Reckoning Position Errors.

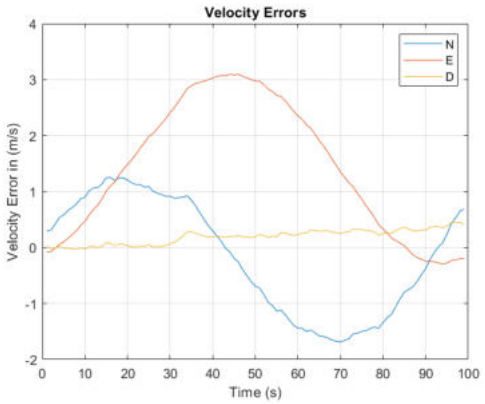

Fig. 7: Circular Motion Dead-Reckoning Velocity Errors.

2) Square Motion Scenario. In the square motion simulation scenario, the simulation consisted of a drone flying in a square motion for 100 seconds. Figure 8 compares the true North-East (N-E) locations with the estimated N-E locations obtained using the EKF, demonstrating the EKF's accuracy in estimating the drone's trajectory.

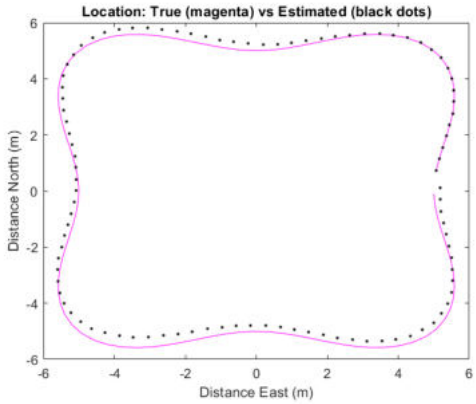

Fig. 8: Comparison of True N-E Locations and Estimated N-E Locations.

Figures 9 and 10 illustrate the position and velocity errors of the drone using the EKF method. The EKF effectively minimizes these errors, showcasing its robustness in handling the nonlinearities and uncertainties in the drone's motion.

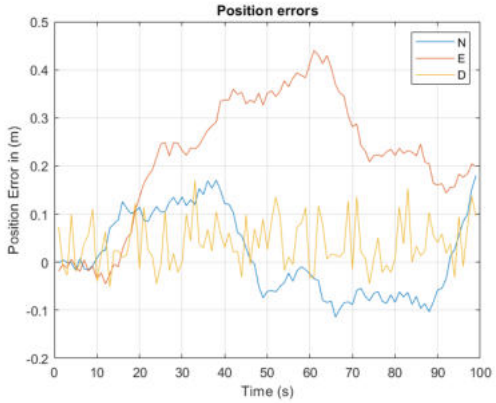

Fig. 9: Square Motion EKF Position Estimation Errors.

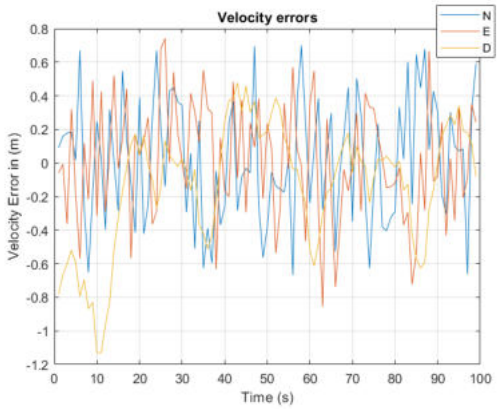

Fig. 10: Square Motion EKF Velocity Estimation Errors.

Figures 11 and 12 illustrate the position and velocity errors associated with the Dead-reckoning method. The errors observed in Dead-reckoning are significantly higher compared to those seen with the EKF method. This highlights the substantial improvement in navigation accuracy achieved by using the EKF.

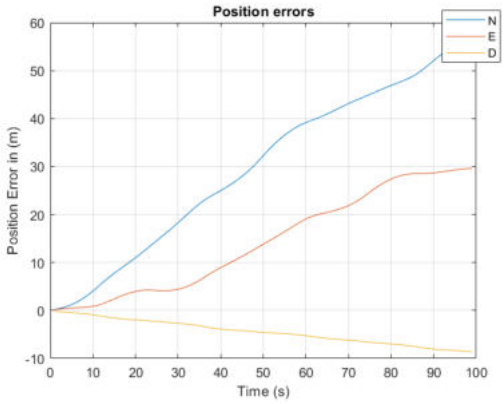

Fig. 11: Square Motion Dead-Reckoning Position Errors.

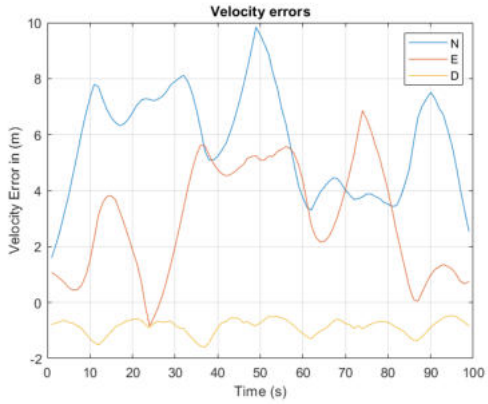

Fig. 12: Square Motion Dead-Reckoning Velocity Errors.

VI. Conclusions

This paper discussed the integration of IMUs and optical flow sensors for enhance velocity estimation and control in UAV navigation systems. IMUs are essential for providing velocity, orientation, and acceleration data, but drift over time. Optical flow sensors measure relative UAV-ground motion, complementing IMUs and reducing drift, improving dead reckoning, especially when GPS is unavailable. A model for optical flow sensors was introduced, relating sensor outputs to the UAV's motion. Notably, these sensors measure angular displacement changes directly, rather than angular velocity, impacting velocity determination. However, optical flow sensors have limitations like a narrow operating range and sensitivity to platform angles. The presented discrete-time model emphasizes the integration of displacement, attitude and altitude data for optimal performance. This research aims to improve UAV navigation system robustness and accuracy in real-world scenarios. Future research will involve empirically evaluating the fusion of IMUs, altimeters, magnetometers, optical flow and ranging sensors in UAVs under diverse conditions. Leveraging optical flow displacement is expected to improve navigation accuracy significantly over that provided by IMU dead reckoning.

References

[1] S. Huang and G. Dissanayake, "Robot localization: An introduction," *Wiley Encyclopedia of Electrical and Electronics Engineering*, p. 1–3, 2016.

[2] T. D. Barfoot, *State Estimation for Robotics: Second edition, Introduction*. Cambridge University Press, 2024, p. 1–4.

[3] Q. Yuan and I.-M. Chen, "Localization and velocity tracking of human via 3 imu sensors," *Sensors and Actuators A: Physical*, vol. 212, p. 25–33, 2014.

[4] S. M. Joshi, "Adaptive control in the presence of simultaneous sensor bias and actuator failures," National Aeronautics and Space Administration, Langley Research Center, Hampton, VA, NASA Technical Report NASA/TM-2012-217231, 2012. [Online]. Available: https://ntrs.nasa.gov/citations/20120016200

[5] B. Wang, "Reduced integration time improves accuracy in dead reckoning navigation systems," Mar 2023. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/reduced-integration-time-improves-accuracy.html

[6] M. L. Abell and J. P. Braselton, "Chapter 2 - first order ordinary differential equations," Tech. Rep., 2018. [Online]. Available: https://doi.org/10.1016/B978-0-12-814948-5.00002-1

[7] H. B. Gilbert, O. Celik, and M. K. O'Malley, "Long-term double integration of acceleration for position sensing and frequency domain system identification," in *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2010, pp. 453–458.

[8] P. Tong, X. Yang, Y. Yang, W. Liu, and P. Wu, "Multi-uav collaborative absolute vision positioning and navigation: A survey and discussion," *Drones*, vol. 7, no. 4, p. 261, 2023.

[9] D. Belfadel, D. Haessig, and C. Chibane, "Relative navigation of uav swarm in a gps-denied environment," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXXII*, vol. 12547. SPIE, 2023, pp. 152–161.

[10] F. Santoso, M. A. Garratt, and S. G. Anavatti, "Visual–inertial navigation systems for aerial robotics: Sensor fusion and technology," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 260–275, 2017.

[11] Mathworks, "Simultaneous localization and mapping – matlab," *What Is SLAM (Simultaneous Localization and Mapping) – MATLAB*, Accessed 2023. [Online]. Available: https://www.mathworks.com/discovery/slam.html

[12] T. Braber, C. De Wagter, G. de Croon, and R. Babuška, "Optical-flow-based stabilization of micro air vehicles without scaling sensors," in *10th International Micro-Air Vehicles Conference*, 2018, pp. 289–297.

[13] S. Lee and J.-B. Song, "Robust mobile robot localization using optical flow sensors and encoders," vol. 1, pp. 1039–1044 Vol.1, 2004.

[14] F. Kendoul, I. Fantoni, and K. Nonami, "Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles," *Robotics and Autonomous Systems*, vol. 57, no. 6–7, pp. 591–602, 2009.

[15] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. London, United Kingdom: Institute of Electrical Engineers, 2004, doi: 10.1049/PBRA017E.