RBE-595 INS-GNSS Integration using UKF

Harshal Bhat
hbhat@wpi.edu

I. Objective

The objective of this report is to develop and evaluate nonlinear Kalman Filter implementations for trajectory analysis involving position and velocities. Specific goals include:

- Derive observation models for error correction and bias modeling.
- Implement and analyze a nonlinear error state Kalman filter with the state vector $\mathbf{x_{FB}} = [L, \lambda, h, \phi, \theta, \psi, v_N, v_E, v_D, e_L, e_\lambda, e_h]$.
- Implement and analyze a nonlinear full state Kalman filter with the state vector $\mathbf{x_{FF}} = [L, \lambda, h, \phi, \theta, \psi, v_N, v_E, v_D, b_{yx}, b_{yy}, b_{yz}, b_{gx}, b_{gy}, b_{gz}]$.
- Evaluate the accuracy and precision of each implementation, discuss the process noise tuning, and explore potential enhancements to the GNSS-based system.

II. Task 1: Observation Model

In the INS-GNSS integration we follow the propagation model of applying an attitude update, a velocity update, and finally a position update.

A. Attitude Update

The attitude update is determined by the Earth's rotation matrix Ω_e^i, the vehicle's rotational velocities ω_e^n derived from its navigational state, and the transformation of these velocities through R_i^e. This process integrates changes over time using previous attitudes and corrections for both Earth's rotation and the vehicle's own motion. It is modelled by the equations (1) to (9)

$$\Omega_e^i = \begin{bmatrix} 0 & -\omega_E & 0 \\ \omega_E & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1}$$

We then calculate ω_e^n as:

$$\omega_e^n = \begin{bmatrix} \frac{v_e}{R_e(L)+h} \\ -\frac{v_n}{R_n(L)+h} \\ -\frac{v_E tan(L)}{R_e(L)+h} \end{bmatrix} \tag{2}$$

where:

$$v_e = R_i^e(v_i - \Omega_i r_i) \tag{3}$$

$$R_i^e = \begin{bmatrix} cos\omega_t & sin\omega_t & 0 \\ -sin\omega_t & cos\omega_t & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$R_E(L) = \frac{R_0}{\sqrt{1 - e^2 sin^2(L)}} \tag{5}$$

$$L = atan(\frac{z_e(R_e(L)+h)}{(1-e^2)(R_e(L)+h) - \sqrt{x_e^2 + y_e^2}}) \tag{6}$$

$$\lambda = atan(\frac{y_e}{x_e}) \tag{7}$$

$$h = \frac{1}{cos(L)}\sqrt{x_e^2 + y_e^2} - R_e(L) \tag{8}$$

Now that we have ω_e^n, we create Ω_e^n and Ω_i^b following the screw form, but also find $R_{b,t}^n$ as:

$$R_{b,t}^n \approx R_{b,t-1}^n(I_3 + \Omega_i^b \delta t) - (\Omega_i^e + \Omega_e^n)R_{b,t-1}^n \delta t \tag{9}$$

B. Velocity Update

The velocity update is computed using the midpoint of rotational transformations between consecutive time steps, applying the average transformed forces $f_{n,t}$. It incorporates gravitational effects and corrects for the Earth's rotation, adjusting the north, east, and downward velocities accordingly.

$$f_{n,t} \approx \frac{1}{2}(R_{b,t-1}^n + R_{b,t}^n) f_{b,t} \tag{10}$$

$$v_{n,t} = v_{n,t-1} + \delta t(f_{n,t} + g(L_{t-1}, h_{t-1}) - (\Omega_{e,t-1}^n + 2\Omega_{i,t-1}^e)v_{n,t-1}) \tag{11}$$

C. Position Update

Finally, we can use the resultant velocities we previously obtained to calculate our position update.

$$h_t = h_{t-1} + \frac{\delta t}{2}(v_{D,t-1} + v_{D,t}) \tag{12}$$

$$L_t = L_{t-1} + \frac{\delta t}{2}(\frac{v_{N,t-1}}{R_e(L_{t-1}) + h_{t-1}} + \frac{v_{N,t}}{R_e(L_{t-1}) + h_t}) \tag{13}$$

$$\lambda_t = \lambda_{t-1} + \frac{\delta t}{2}(\frac{v_{E,t-1}}{(R_E(L_{t-1}) + h_{t-1})cosL_{t-1}} + \frac{v_{E,t}}{(R_E(L_{t-1}) + h_t)cosL_t}) \tag{14}$$

Our Observation model for Feedback type system is given by,

$$z = I_n mean(x_{FF}) + N(0, R) \tag{15}$$

$$z = I_{12\times12}\begin{bmatrix} L \\ \lambda \\ h \\ \phi \\ \theta \\ \psi \\ V_N \\ V_E \\ V_D \\ e_L \\ e_\lambda \\ e_h \end{bmatrix} + N(0,R) = \begin{bmatrix} L \\ \lambda \\ h \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + N(0,R) \quad (16)$$

Errors are defined by

$$distmean(x_{FF})_{9:12} = mean(x_{FF0:3}) - gnss_{0:3} \quad (17)$$

Similarly, our observation model for Full State is given by,

$$z = I_{15\times15}\begin{bmatrix} L \\ \lambda \\ h \\ \phi \\ \theta \\ \psi \\ V_N \\ V_E \\ V_D \\ b_{a_x} \\ b_{a_y} \\ b_{a_z} \\ b_{g_x} \\ b_{g_y} \\ b_{g_z} \end{bmatrix} + N(0,R) = \begin{bmatrix} L \\ \lambda \\ h \\ 0 \\ 0 \\ 0 \\ V_N \\ V_E \\ V_D \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + N(0,R) \quad (18)$$

Feedback system is 12 states vector and Fullstate system is 15 states vector.

III. Task 2 & 3: Nonlinear Error State and Full State Implementaion

Task 2 and 3 were implemented in the file `inss_gns.py` as a class for modularity. To observe plots and rmse values run `integrator_main.py` file. Some issues while implementation were the covariance matrix shot up to extremely high values and corresponding latitude, longitude estimates were highly iregular. To overcome this we had to device a try, except conditions. Another issue with covariance matrix not being positive definite. A small jitter addition and squrare shape conditions fixed this issue. Tuning the noise was done by comparing values of covariance matrix for different iterations of the filter. Figures 1, 2, 3, 4 depict results for Feedback type model. Figures 5, 6 , 7, 8 depict results for Full state implementation.

IV. Task 4: Performance Analysis

This image presents a top-down perspective of the aircraft's latitude and longitude along its trajectory, subtly highlighting the actual path in the background. Additionally, it includes a

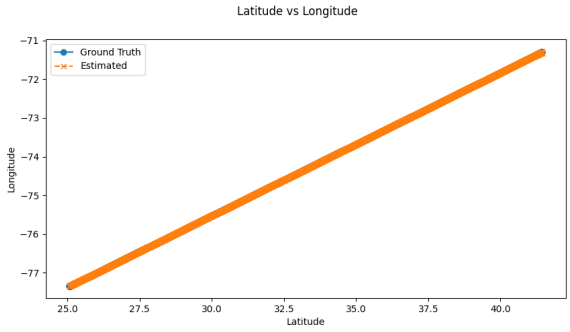

Fig. 1. Latitude vs Longitude for Feedback model

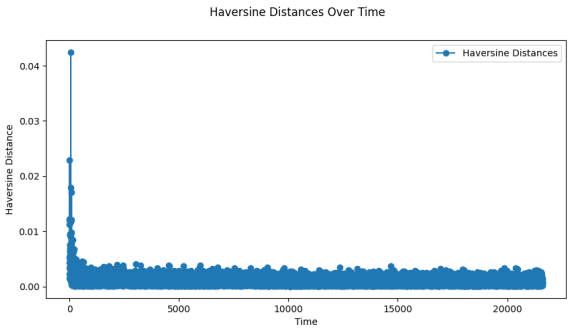

Fig. 2. Haversine distance between the estimated position and the true (GNSS measured) position for Feeback Model

graph comparing the haversine distance between our anticipated position and our calculated position over time.

The average RMSE value for Feeback Model is around $\mathbf{2.221} \times 10^{-3}$

The average RMSE value for Feedforward method is $\mathbf{3.3} \times 10^{-3}$, however multiple spikes suggest that the performance is unstable. Another observation was the time takes for feedforward is higher than feedback model.

V. Discussion and Conclusion

The estimations of Feedback type system are better than Feedforward system model eventough the R. Initially, it seemed logical that a system which accounts for errors and integrates them into its predictions would yield a more accurate model. There might be an issue with the error integration with the plant model, which drifts the estimations to unlikely values and flaw in the implementation.

To improve the INS-GNSS integration techniques like Differential GNSS (DGNSS) or Real-Time Kinematic (RTK) positioning can be employed to correct GNSS errors by using reference stations. Applying machine learning algorithms to predict errors or to directly estimate trajectories can potentially offer improvements by learning complex patterns from large datasets of sensor readings and their associated errors.

Using the Haversine distance to compare errors between GNSS and INS estimates might not be the most accurate method because it only considers the latitude and longitude, ignoring the altitude, which is crucial in 3D navigation.

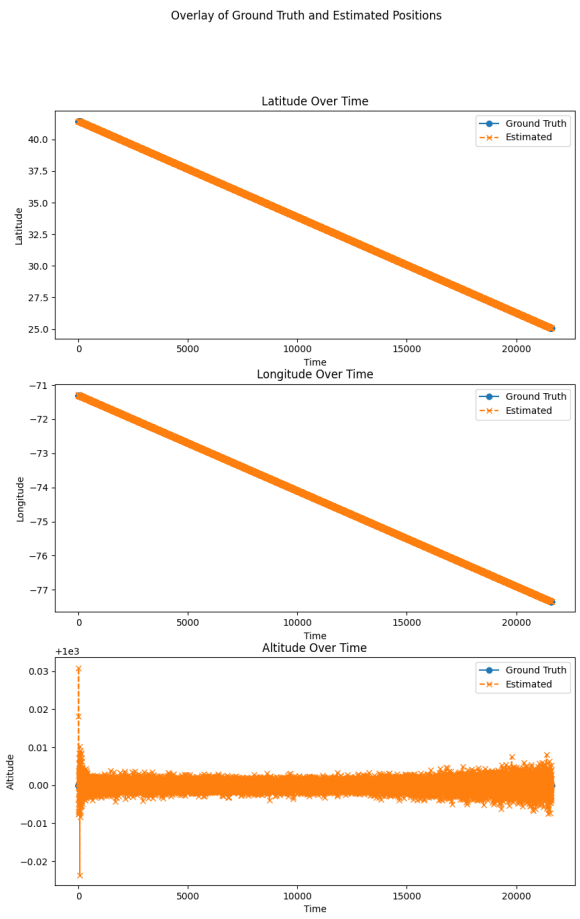

Fig. 3. Latitude, Longitude and Altitude change over time for Feedback Model

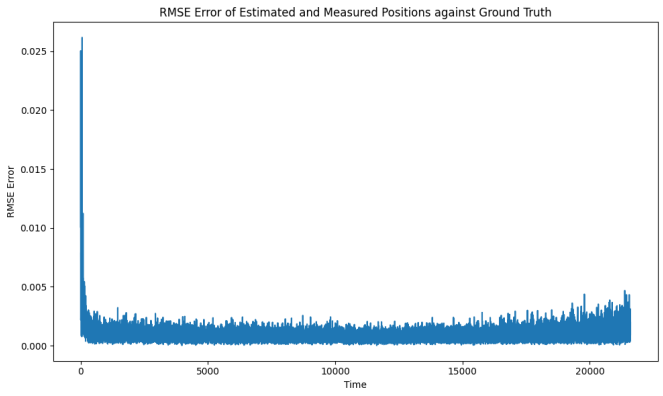

Fig. 4. RMSE values of Estimated and Measured Positions for Feedback Model

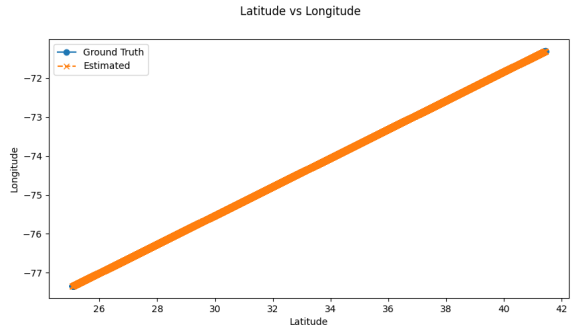

Fig. 5. Latitude vs Longitude for Feedforward model

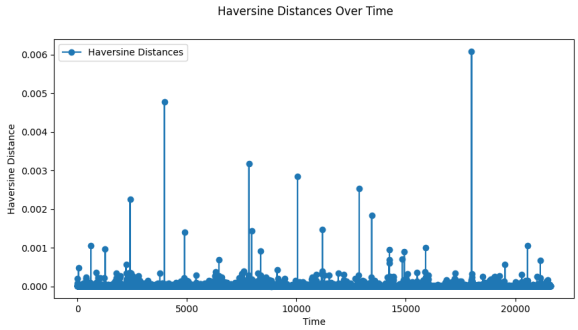

Fig. 6. Haversine distance between the estimated position and the true (GNSS measured) position for Feedforward Model

Moreover, the Haversine formula might not accurately reflect small distance errors due to the curvature of the Earth. A better approach would be to use 3D Euclidean distance, which includes altitude and gives a more comprehensive measure of spatial error. Additionally, assessing the error components in three dimensions or using statistical measures like RMSE or MAE can provide deeper insights into error characteristics and their impact.

REFERENCES

[1] Alex Becker(2023) Kalman Filter from the Ground up. https://www.kalmanfilter.net/book.html

[2] Sun, Ke Mohta, Kartik Pfrommer, Bernd Watterson, Michael Liu, Sikang Mulgaonkar, Yash Taylor, Camillo Kumar, Vijay. (2017). Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight. IEEE Robotics and Automation Letters. PP. 10.1109/LRA.2018.2793349.

[3] https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/10-Unscented-Kalman-Filter.ipynb

[4] https://www.cs.cmu.edu/ 16831-f14/notes/F11/16831 lecture04 tianyul.pdf

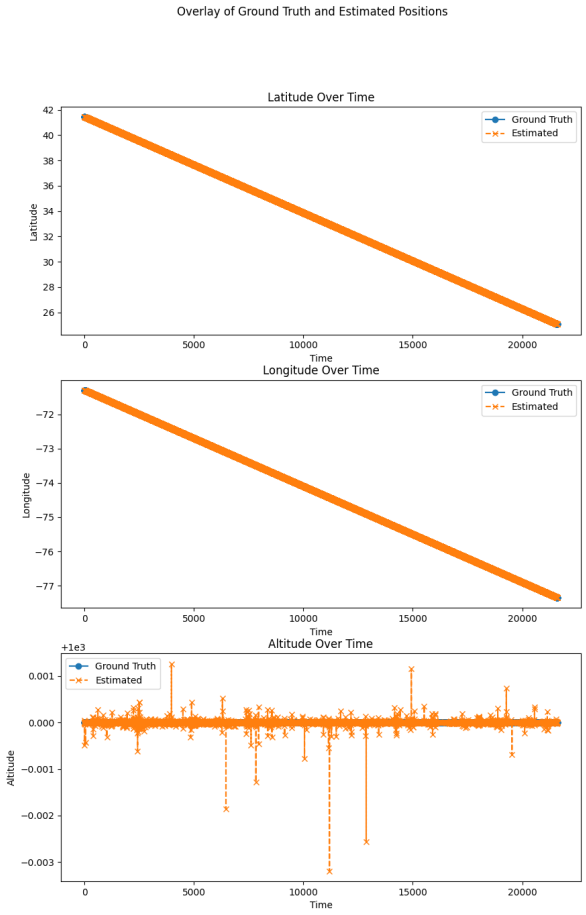

Fig. 7. Latitude, Longitude and Altitude change over time for Feedforward Model

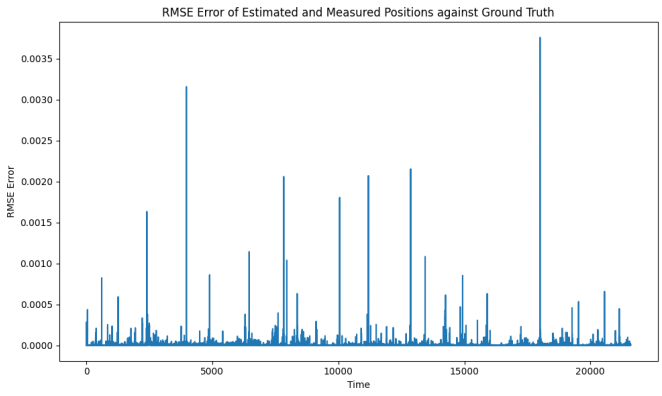

Fig. 8. RMSE values of Estimated and Measured Postions for Feedforward Model