EKF-Based Drone (9-State) and Rover (8-State) Simulation: Architecture, Workflows, and Implementation Status

Tuna Akyürek

28.08.2025

Abstract

This report consolidates the current state of two MATLAB simulations: (i) a 6-DOF multirotor using a 9-state EKF with IMU-driven prediction and GPS/Barometer/Magnetometer updates, and (ii) a 2D RoboMaster-style rover using an 8-state EKF with IMU prediction and GPS position, magnetometer yaw, and wheel-encoder velocity updates. We describe workflows, code structure, tuning, and expected results. **All mathematical definitions (EKF notation, 8-DOF/9-DOF models, Jacobians, and Joseph-form appendix) are centralized in `Total_Formulary.tex`; this report intentionally omits those formulas to avoid duplication.**

Contents

1 Purpose and Scope

This report consolidates the current state of two MATLAB simulations:

- A 6-DOF multirotor (drone) using a 9-state EKF with IMU-driven prediction and GPS/Barometer/Magnetometer updates.
- A 2D RoboMaster-style rover using an 8-state EKF with IMU-driven prediction and GPS position, magnetometer yaw, and wheel-encoder velocity updates.

All equations are provided in `Total_Formulary.tex`.

2 Systems Overview

Drone (9-state) — simulation intent

- IMU-driven prediction at IMU rate; GPS, barometer, magnetometer updates at their rates.
- Ground truth via `drone_dynamics.m`; EKF prediction via `drone_dynamics_imu.m`.
- Jacobians via `calculate_F_sensor_only.m`.

Rover (8-state) — simulation intent

- Planar body dynamics with forward/lateral accelerations and yaw rate.
- Wheel-encoder body-frame velocity replaces GPS velocity.
- Constraints (NHC/ZUPT/ZARU) applied per-step to stabilize yaw/velocity coupling.

3 Repository Structure (key files)

Drone (root):

- `main_random.m` (entry), `parameters.m`, `drone_dynamics.m`, `sensor_model.m`, `ekf_sensor_only.m`, `drone_dynamics_imu.m`, `calculate_F_sensor_only.m`, `rotation_matrix.m`, `animate_drone.m`

Rover (`RoboMaster_Sim/`):

- **Entry**: `main_rover_sim.m`
- **Parameters**: `parameters_rm.m`
- **Dynamics**: `rover_dynamics.m`
- **Sensors**: `sensor_model_wheel_encoders.m`
- **EKF**: `ekf8_init.m`, `ekf8_predict.m`, `ekf8_update_*.m`, `ekf8_apply_constraints.m`, `kalman_update.m`, `wrapToPi_local.m`
- **Visualization/processing**: `animate_rover_xy.m`, `post_smooth_estimates_8state.m`
- **Realistic variant**: `main_rover_sim_realistic.m`, `parameters_rm_realistic.m`, `sensor_model_realistic.m`
- **Tuning/analysis**: `fine_tuning_logger.m`

4 Simulation Workflows

Drone

1. Load parameters, set initial truth and EKF state/covariance.
2. Generate commanded trajectory (random-walk velocity) and torques.
3. Integrate truth with `drone_dynamics.m`.
4. Generate measurements with `sensor_model.m`.
5. EKF prediction at IMU rate via `drone_dynamics_imu.m`; update with GPS/Baro/Mag using `calculate_F_sensor_only.m`.
6. Log data, animate, and analyze.

Rover

1. Load `parameters_rm.m` (or `parameters_rm_realistic.m`).
2. Initialize EKF; optional brief alignment with first GPS/Mag.
3. Integrate truth with `rover_dynamics.m`.
4. Generate measurements with `sensor_model_wheel_encoders.m`.
5. EKF prediction each step; apply constraints; update with available sensors.
6. Log, optionally post-smooth, visualize, and analyze errors and noise.

5 Coordinate Frames and Conventions

- **Drone**: NED inertial frame; body-to-NED via ZYX rotation; barometer measures $-z$ in NED.
- **Rover**: XY world frame (planar); heading θ in radians; wheel velocities in body frame; wrap angles to $(-\pi, \pi]$.

6 Noise, Bias, and Realism

Drone

- Process noise $\boldsymbol{Q}$ scales with Δt; measurement noise $\boldsymbol{R}$ tuned per sensor.
- `parameters.m` sets timing, vehicle properties, and sensor noise.

Rover

- Standard run uses `parameters_rm.m` with concise noises; velocity updates from wheel encoders.
- Realistic variant demonstrates GPS multipath, IMU bias drift, magnetometer interference, and quality indicators.

7 Post-Processing and Smoothing (Rover)

Optional post-smoothing applies median filtering and zero-phase Savitzky–Golay smoothing for cleaner visualizations while keeping the real-time EKF causal.

8 Schedules and Rates

- **Drone**: physics/IMU/GPS/Baro/Mag sample times are configurable.
- **Rover**: IMU $\sim 100\,\text{Hz}$; GPS $\sim 5\,\text{Hz}$; mag $\sim 10\,\text{Hz}$; wheel encoders aligned with GPS in logs/updates.

9 Initialization

Drone

Moderate initial covariance; convergence driven by GPS/Baro/Mag updates.

Rover

Soft prior with optional initial GPS/Mag updates; realistic run uses: position from GPS or origin, heading from magnetometer or default, zero velocity, zero-mean biases with large uncertainty.

10 Constraints and Stability

- Rover applies constraints each step to reflect non-holonomic behavior and maintain stability.
- Angle wrapping to $(-\pi, \pi]$ avoids discontinuities.
- Conditioning safeguards (e.g., SVD checks, small regularization) included.

11 Usage

Drone

1. Set MATLAB path to repo root; configure `parameters.m`.
2. Run `main_random.m`.

Rover

1. Run `RoboMaster_Sim/main_rover_sim.m` or `.../main_rover_sim_realistic.m`.
2. Optionally toggle post-smoothing in `main_rover_sim.m`.

12 Expected Results

Drone

EKF tracks position, altitude, and yaw with accuracy governed by $\boldsymbol{Q}/\boldsymbol{R}$ tuning and sampling intervals.

Rover

- Raw EKF shows sensor-correlated noise; post-smoothing improves visual smoothness.
- Realistic parameters produce errors consistent with consumer-grade sensors.

13 Tuning Guidance (Practical)

Process noise $\boldsymbol{Q}$: increase for agility/unmodeled dynamics; decrease to reduce jitter.
Measurement noise $\boldsymbol{R}$: match sensor specs; underestimation can destabilize, overestimation slows response.
Initial covariance $\boldsymbol{P}_0$: use higher values with poor prior; optionally apply conditional initial updates.

14 Known Limitations

Drone

Simplified couplings and linearization tune for efficiency; aggressive maneuvers may need retuned $\boldsymbol{Q}/\boldsymbol{R}$.

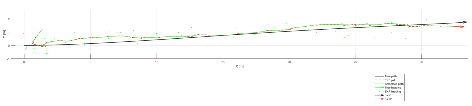
Rover

Encoder slip and mag interference only partially modeled in standard run; realistic variant shows more effects but remains simplified.

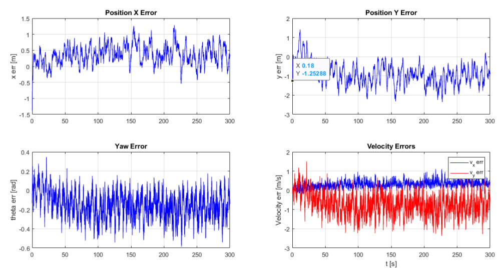
15 Verification and Analysis

Standard plots:

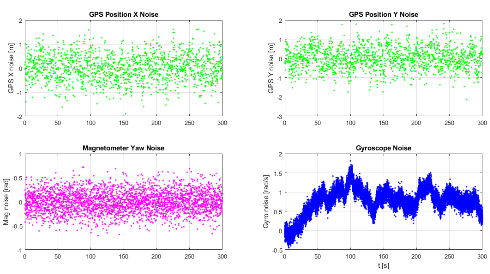
- **Drone**: trajectory, attitude, sensor vs estimate, error plots, animation.
- **Rover**: XY trajectory and states, raw sensors, sensor vs estimate, noise analysis, errors, animation.



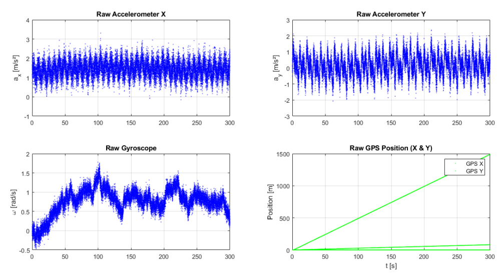
(a) Trajectory / headings (standard).



(b) EKF errors (standard).

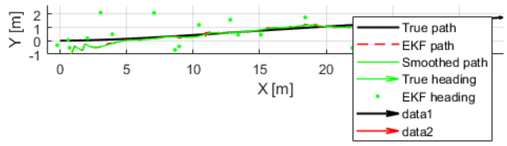


(c) Sensor noise (standard).

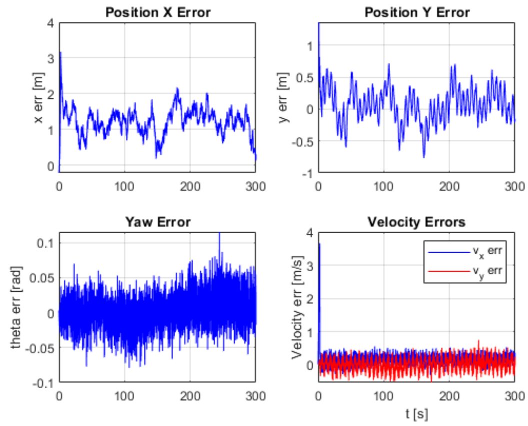


(d) Raw sensors (standard).

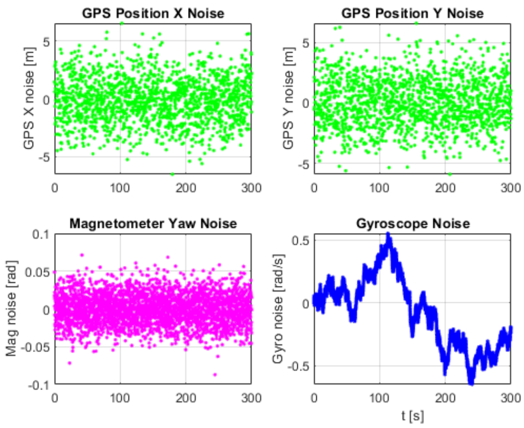
Figure 1: Standard/baseline RoboMaster S1 run with simulated iPhone sensors.



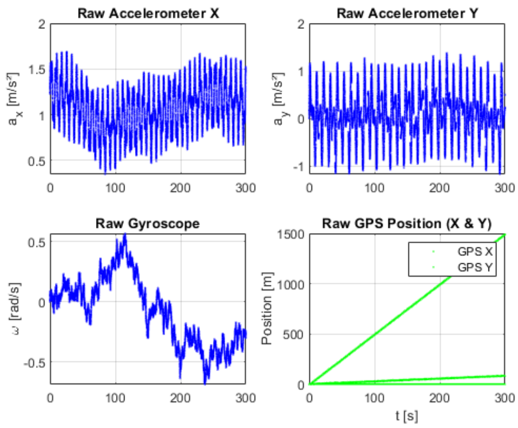
(a) Trajectory / headings (realistic).



(b) EKF errors (realistic).



(c) Sensor noise (realistic).



(d) Raw sensors (realistic).

Figure 2: Realistic-parameters run (multipath, IMU drift, mag interference).

16 Summary of Key Design Choices

- Drone EKF: IMU prediction; GPS/Baro/Mag updates; NED with barometer as $-z$.
- Rover EKF: wheel-encoder velocity rather than GPS velocity; immediate updates and constraints.
- Realistic error modeling available; post-smoothing is offline only for visualization.

17 Reproducibility Checklist

- MATLAB path includes repository root and `RoboMaster_Sim/`.

Drone:

- `parameters.m` configured; run `main_random.m`.

Rover:

- Run `RoboMaster_Sim/main_rover_sim.m` or `.../main_rover_sim_realistic.m`.
- Optional: toggle post-smoothing in `main_rover_sim.m`.

References (MATLAB-based report)

[1] S. Särkkä and L. Svensson, *Bayesian Filtering and Smoothing*, 2nd ed. Cambridge University Press, 2023.

[2] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[3] A. Barrau and S. Bonnabel, "The invariant extended kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017.

[4] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed. Artech House, 2013.

[5] D. Choukroun, I. Y. Bar-Itzhack, and Y. Oshman, "Novel quaternion kalman filter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 174–190, 2006.

[7] P. Kaniewski, R. Gil, and S. Konatowski, "Estimation of uav position with use of smoothing algorithms," *Metrology and Measurement Systems*, vol. 24, no. 1, pp. 127–142, 2017.

[10] M. S. H. Motayed, *Sensor fusion for drone position and attitude estimation using ekf*, ResearchSquare Preprint, 2025.