IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 16, Issue 10, October 2026, ||Series -I|| PP 10-13

Ground-Test Characterization of a Barometer–Accelerometer Kalman Filter on a Custom Flight Computer

Abhigna Yerramreddikalva
Bengaluru, India
Received 27 October 2026; Accepted 07November 2026

Abstract
This paper characterizes the altitude Kalman filter running on, a custom flight computer built around an NXP i.MX RT1062. The filter fuses barometric altitude and accelerometer data in a three-state (altitude, velocity, acceleration) linear model at 50 Hz. A 52.6 s ground log was analyzed, consisting of a 16 s stationary window followed by handheld vertical motion. An offline replay of the filter reproduced the on-board estimates to within 3.1 mm (altitude) and 3.8 mm/s (velocity) RMS, so the firmware could be studied and modified offline. While stationary, the filter reduced altitude noise from 26 cm to 8.7 cm (1σ) and held velocity to 0.11 m/s (1σ). Over the same window, direct integration of the accelerometer drifted at 0.085 m/s per second. Code review and the log together exposed six implementation issues. The most significant were three accelerometer axes sharing one state vector, acceleration being counted twice in the prediction step, and the use of untilted body-frame acceleration as vertical acceleration. Correcting the first two issues and retuning noise covariances to measured sensor noise reduced stationary altitude and velocity noise by a further 37% and 59% in replay.

Keywords: *Kalman filter, sensor fusion, barometric altimeter, accelerometer, flight computer, state estimation*

I. Introduction

Small rockets and hoppers need altitude and vertical velocity for event detection and control. A barometer gives an absolute altitude that does not drift but is noisy. An accelerometer is responsive but must be integrated twice, so any bias grows into large velocity and position error within seconds. A Kalman filter combines the two, weighting each by its modeled uncertainty [1, 2].

TFAC is a custom flight computer (six PCB revisions) built around an NXP i.MX RT1062 microcontroller with a 10-DOF sensor suite, telemetry, and C++ firmware. Its altitude filter is implemented with the BasicLinearAlgebra library [3]. This paper (i) describes the filter as implemented, (ii) validates an offline replay against on-board outputs, (iii) quantifies performance on a ground log, and (iv) reports implementation issues found, with their measured effect.

II. Filter Design as Implemented

The state vector is x = [h, v, a]^T: altitude, vertical velocity and vertical acceleration. Prediction uses a constant-acceleration model with the measured acceleration u also applied as a control input:

$$\hat{x}_k^- = F\hat{x}_{k-1} + B\,u_k, \quad P_k^- = F P_{k-1} F^T + Q \tag{1}$$

with F = [[1, Δt, Δt²/2], [0, 1, Δt], [0, 0, 1]] and B = [Δt²/2, Δt, 0]^T. Δt is computed from millis(). The update uses z = [h_baro, 0, a_meas]^T with H = diag(1, 0, 1):

$$K = P^- H^T (H P^- H^T + R)^{-1}, \quad \hat{x} = \hat{x}^- + K(z - H\hat{x}^-), \quad P = (I - KH)P^- \tag{2}$$

The tuning is Q = diag(3.1×10^{-8}, 1.25×10^{-5}, 10^{-3}) and R = diag(0.01, 0.01, 0.01). The measured acceleration is a_meas = 9.80 − a_x,body, where x is the vehicle's long axis. Each loop calls the prediction once, then runs the update three times, once each with the x, y and z accelerometer axes.

III. Test Data and Replay Method

The log (FLT95) was recorded at 50 Hz on the bench. The first 543 rows had corrupted timestamps and were discarded, leaving 2632 samples (52.6 s). The first 4 s were treated as filter warm-up. From 4–20 s the board was stationary. From 20 s onward it was moved by hand over a ±1 m range, with substantial tilt (median 10°, 90th percentile 28°).

The filter was re-implemented in Python using the logged barometer and accelerometer values and logged time steps. After warm-up, the replay matched the on-board altitude and velocity to 3.1 mm and 3.8 mm/s RMS. The residual is consistent with the logs' two-decimal rounding. This confirms both the reading of the firmware in Section 2 and that modified versions can be evaluated on the same inputs.

IV. Results

4.1 Stationary performance

In the stationary window, the barometer altitude had a standard deviation of 26.0 cm. The on-board filter output had 8.7 cm, a 3.0× reduction (Fig. 1). Velocity had a standard deviation of 0.11 m/s and a mean of −0.03 m/s. The accelerometer read 9.90 m/s² at rest, a 0.10 m/s² bias against the 9.80 m/s² constant used in the firmware. Integrating it directly produced a velocity drift of −0.085 m/s per second, or 1.4 m/s over 16 s. Over a 60 s flight this would grow to about 5 m/s. The filter bounded this error because the barometer continuously corrects velocity through the state coupling.

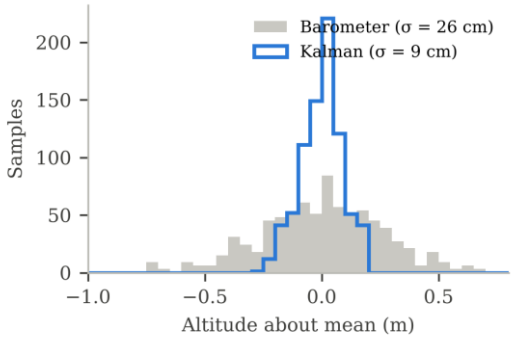


Figure 1. Distribution of altitude about its mean in the stationary window (4–20 s, 800 samples). Raw barometer versus on-board Kalman estimate.

4.2 Handheld motion

During motion, the filter tracked the barometer trend with visibly less noise (Fig. 2, top). Its velocity stayed within ±1 m/s until the final 0.5 s. Over the same period, accelerometer-only integration diverged to over 8 m/s (Fig. 2, middle). Rapid tilting and an end-of-test flip (tilt > 100°) coincided with the velocity spike at the end. Without an independent altitude reference during motion, tracking accuracy could not be measured. Only noise and drift are reported quantitatively.

V. Implementation Issues Found

Reviewing the code against the log identified six issues, summarized in Table 1.

Table 1. Issues found, evidence and recommended fix

Issue	Evidence	Recommended fix
1. The x, y and z updates share one state vector and covariance	Logged x, y and z velocities correlate at r = 0.9997; the barometer is applied three times per step	Use one vertical filter; estimate y and z separately if needed
2. Acceleration is counted twice in prediction (via F and via B·u)	Prediction adds both the state acceleration and the measured acceleration	Keep a as a state with B = 0, or drop a from the state and use u as an input
3. Body-axis acceleration is used as vertical acceleration	Tilt reached 28° (90th pct.); a 30° tilt creates 1.3 m/s² of false vertical acceleration	Rotate the accelerometer into the world frame using the attitude quaternion
4. a_meas = 9.80 − a_x has the opposite sign to an up-positive altitude	Upward acceleration gives a negative a_meas	Use a_x − g; verify with a vertical drop or lift test
5. R does not match measured noise	Barometer σ = 26 cm (R = 0.067 m²) versus R = 0.01 m²; accelerometer σ = 0.18 m/s²	Set R from stationary logs; remove accelerometer bias at startup
6. Timing and indexing details	Δt from millis() has 1 ms resolution at 20 ms steps; startTime is not set on the first call; X(0,2) reads X(2,0) only because of storage order	Use micros(); initialize startTime; use X(2,0); log P(0,0) with more decimals

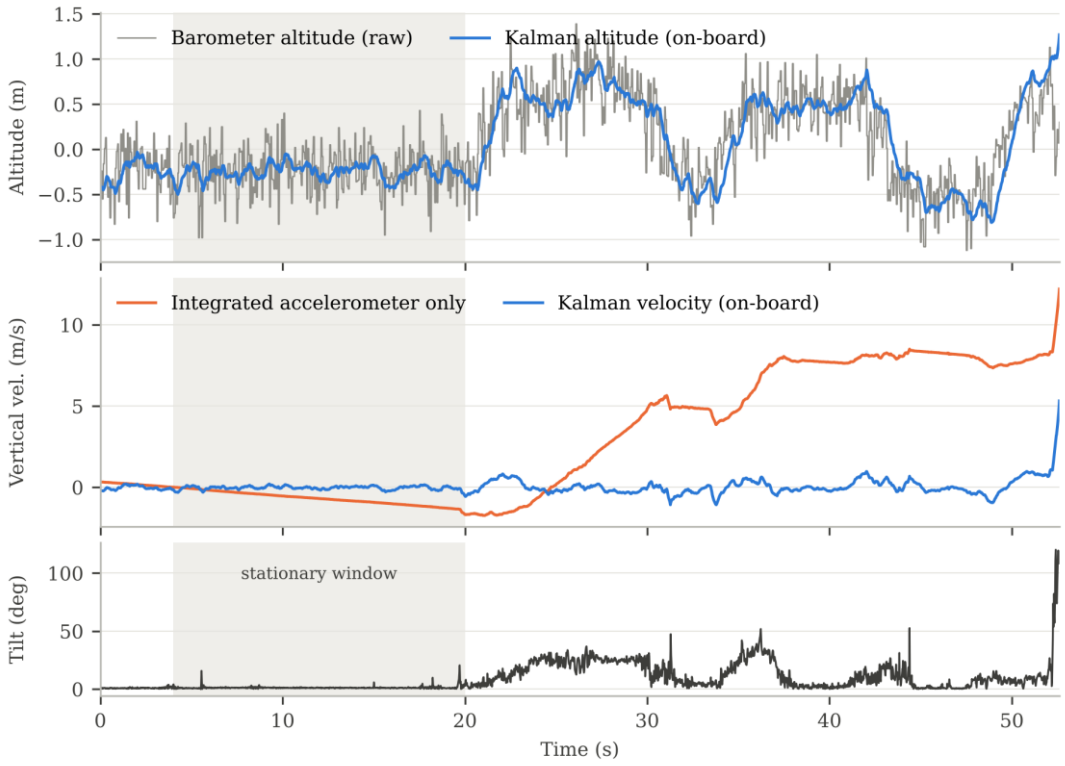


Figure 2. Ground log FLT95. Top: barometer altitude and on-board Kalman altitude. Middle: on-board Kalman velocity versus direct accelerometer integration (zeroed at t = 4 s). Bottom: tilt of the vehicle x-axis from vertical, computed from the accelerometer. The shaded region is the stationary window.

VI. Effect of Corrections

Issues 1, 2 and 5 were corrected cumulatively in the replay and evaluated on the stationary window, where the true velocity is zero (Table 2). Issues 3 and 4 cannot be evaluated on this window, since the board was level and not accelerating. They require an attitude estimate and a controlled vertical-motion test.

Table 2. Stationary-window performance of replayed filter variants

Variant	Altitude σ (cm)	Velocity σ (m/s)	Velocity mean (m/s)
A — As implemented	8.7	0.106	−0.032
B — Single vertical update (fix 1)	7.4	0.082	−0.175
C — B + no double-counting (fix 2)	7.3	0.071	−0.087
D — C + measured R, bias removed (fix 5)	5.5	0.043	+0.022

Each correction lowered noise. The full set (D) reduced altitude noise from 8.7 cm to 5.5 cm (37%) and velocity noise from 0.106 m/s to 0.043 m/s (59%). Bias removal also brought mean velocity close to zero. Variants B and C show a larger mean velocity offset than A. Without the extra barometer updates, the uncorrected 0.10 m/s² accelerometer bias has more influence. This shows why the bias correction belongs with the structural fixes.

VII. Limitations and Future Work

This is a single ground test with no independent altitude truth during motion, so accuracy under dynamics is not established. The stationary window favors filters that trust the barometer less, since there is no motion to lag behind. The next steps are: rotate acceleration into the world frame using TFAC's quaternion attitude estimate; repeat the test with a known vertical displacement (for example, raising the board through a measured height, or an elevator ride); and evaluate on flight data. Adding the barometer innovation and P(0,0) to the log would allow a filter-consistency check (normalized innovation squared).

VIII. Conclusion

TFAC's barometer–accelerometer filter reduced stationary altitude noise threefold and prevented the velocity drift of direct integration. A replay that reproduces the firmware to millimeter level made it possible to find and measure implementation issues. Fixing the shared-state, double-counting and noise-tuning issues cut residual noise by a further 37–59%. Tilt compensation is the remaining correction, and the most important one for flight.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.
[2] G. Welch and G. Bishop, "An introduction to the Kalman filter," Tech. Rep. TR 95-041, Univ. of North Carolina at Chapel Hill.
[3] T. Stewart, BasicLinearAlgebra Arduino library, GitHub.
[4] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed. Artech House, 2013.