AVIAN: Autonomous Visual Indoor Aerial Navigation

Matei Gardea* **Armaan Goklani** **Ethan Bensimon**
University of California, Berkeley

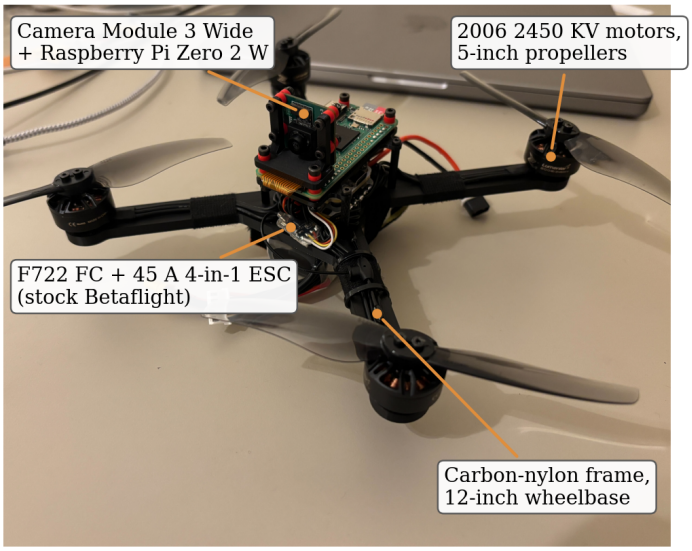

Figure 1: The AVIAN platform: a 250 g 3D-printed carbon-nylon quadrotor carrying a Raspberry Pi Zero 2 W and a Camera Module 3 Wide as its autonomy payload.

Abstract: Most autonomous indoor flight systems depend on sensing and compute that dominate the cost and weight of the vehicle: LiDAR, depth cameras, motion-capture infrastructure, or GPU-class companion computers. We study how much of the indoor position-estimation and waypoint-hold problem can be solved without them. AVIAN is a 250 g quadrotor whose autonomy stack consists of a Raspberry Pi Zero 2 W, a single rolling-shutter camera, and a printed chessboard fixed in the environment. Two earlier designs did not produce usable localization: streaming video to an offboard computer exceeded 250 ms glass-to-glass latency, and markerless frame-to-frame visual odometry failed under propeller vibration, motion blur, and scale ambiguity. The final system runs entirely onboard, recovering metric pose from the chessboard with a planar PnP solver at 10–30 Hz, filtering it with a Kalman position filter, and applying a PD position law through the flight controller's MSP RC-override interface. The marker is detected in 48–100 percent of logged frames with 0.24–0.29 px median reprojection error out to about 2.5 m. During an 80 s closed-loop hold, position error stays bounded at 36 cm XY RMS, with a 0.26 Hz oscillation in both horizontal axes: a limit cycle from running the control gains near the stability margin the loop's latency imposes. The platform design and flight software are available at https://mateig.github.io/avian-site/.

Keywords: Quadcopter, GPS-denied navigation, visual localization, open-source hardware

*Correspondence: matei.gardea@berkeley.edu

1 Introduction

Autonomous indoor flight is a mature capability on well-instrumented platforms. Motion-capture arenas provide millimeter-level ground truth, LiDAR and depth cameras measure geometry directly, and GPU-class companion computers run visual-inertial odometry at high rates. Each of these additions removes a constraint, and each adds cost, weight, power draw, and integration effort. Racing-quadrotor hardware occupies the other end of the spectrum: hobby-grade flight controllers, brushless motors, and 3D-printed frames are inexpensive and widely available, but the firmware assumes a human pilot and provides no autonomy.

This paper examines what lies between those two endpoints. We built AVIAN, a 250 g quadrotor whose perception and autonomy stack is a Raspberry Pi Zero 2 W, a single wide-angle camera, and a printed chessboard fiducial placed in the environment (Figure 1). The platform uses no LiDAR, no depth camera, no motion capture, no GPU, and no external compute, and the flight-controller firmware is unmodified. The question we address is how far indoor position estimation and waypoint hold can be pushed on this stack.

Reaching a working system required discarding two designs. The first streamed 720p video to a laptop for offboard processing. The stream itself worked, but measured glass-to-glass latency exceeded 250 ms against an initial budget of under 100 ms, which is too slow for inner-loop position control. The second design ran markerless ego-motion estimation onboard, using FAST corners, pyramidal Lucas-Kanade tracking, and essential-matrix decomposition. On a vibrating airframe over weakly textured indoor surfaces, a large fraction of frame-to-frame rotation estimates were physically impossible, recovered translation directions were close to uniformly distributed, and monocular scale was unobservable, so no usable pose estimate was produced. The third design accepts one piece of environmental infrastructure, a printed 3×5 chessboard with 70 mm squares, and recovers metric camera pose from it with a planar PnP solver. The vehicle flew autonomously on it.

The flight software runs three parallel threads on the Pi at a target rate of 30 Hz: a vision thread that detects the marker on a downsampled frame, refines corners at full resolution, and solves PnP; a Kalman position filter that fuses intermittent vision measurements under a constant-velocity motion model and coasts through detection gaps; and a flight-controller interface that reads the pilot's RC channels over MSP and writes back roll and pitch corrections from a PD position law, limited to about two percent of stick authority. Throttle and yaw remain under pilot control, and a transmitter switch engages or disengages autonomy at any time.

The quantitative results are mixed. The vision pipeline is accurate: camera calibration reaches 0.246 px mean reprojection error, and accepted PnP solutions have 0.24–0.29 px median reprojection error with detection rates of 48–100 percent across logged flights. The closed loop is stable but imprecise: over an 80 s autonomous hold, the vehicle stays bounded with 36 cm XY RMS error, and the error spectrum has a dominant peak near 0.26 Hz in both horizontal axes. The oscillation holds near-constant amplitude over the flight: a limit cycle from operating the control gains near the stability margin that the loop's latency imposes.

This paper makes three contributions. First, it documents a complete, reproducible autonomy stack for hobby-grade hardware, including the MSP override mechanism that lets a companion computer share control authority with an unmodified Betaflight flight controller. Second, it reports measurements of two designs that failed, offboard streaming and markerless monocular odometry, and identifies the causes. Third, it characterizes the closed-loop behavior of the system, including the limit cycle and the latency measurements that explain it.

2 Related Work

Autonomy on resource-constrained aerial platforms. A sustained line of work moves autonomy onto progressively smaller vehicles. PULP-Dronet demonstrated CNN-based navigation on a 27 g Crazyflie using a parallel ultra-low-power GAP8 processor [1], and later work in the same ecosys-

tem has distilled smaller networks for navigation and combined local and global perception onboard nano-UAVs [2, 3]. Our platform is heavier (250 g) but less specialized: it uses an unmodified consumer single-board computer and a hobby flight controller rather than a purpose-built research deck, and we measure what that commodity stack supports.

Fiducial-based UAV localization. Visual fiducials such as AprilTag [4], ArUco [5], and chessboard targets are a standard low-cost source of indoor metric pose, and recent work continues to adapt them for flight: marker fields with multiple size classes for indoor localization [6], fiducial-corrected visual-inertial localization for GPS-denied inspection [7], and learned detectors such as YoloTag that improve runtime robustness [8]. We use a single printed chessboard because its corner detector and sub-pixel refinement are mature in OpenCV [9] and its planar geometry admits the efficient IPPE PnP solver [10]. Relative to this literature, our contribution is an end-to-end characterization on minimal compute: detection range, dropout statistics, solve time, and closed-loop precision for a single-marker system.

Markerless visual odometry under vibration. Monocular VO and visual-inertial systems such as ORB-SLAM3 [11] and VINS-Mono [12] avoid environmental infrastructure, and event-camera approaches target the latency problem directly [13]. Our experience matches the documented failure modes on small airframes: vibration and rolling-shutter blur corrupt feature tracks, indoor surfaces provide little texture, and monocular scale is unobservable without inertial fusion at rates our serial link does not support. Section 4.2 reports these failures quantitatively, since they motivated the fiducial design.

Latency in vision-based flight control. The effect of perception latency on closed-loop flight is well studied for agile platforms: work on high-speed sense-and-avoid quantifies how latency shrinks the feasible flight envelope [14], event cameras are motivated by their low sensor latency [13], and a recent study of vision-based teleoperation reports a sharp collapse in stability as one-way perception latency grows from 150 to 225 ms [15]. Our measurements place a Pi-class commodity stack in that regime, with over 250 ms glass-to-glass for streaming and 40–100 ms for onboard detection alone, and our closed-loop spectra show the corresponding oscillation.

Open aerial research platforms. Agilicious [16] and the Crazyflie ecosystem [17] provide open quadrotor stacks for research, typically with custom autopilot firmware or dedicated compute decks. AVIAN instead treats an unmodified Betaflight flight controller [18] as a black box and injects control through the MSP RC-override channel that the firmware already exposes. This interface is limited, since the companion computer can only move virtual sticks, but it transfers to any MSP-speaking vehicle without reflashing, and the pilot retains a hardware-level override at all times.

3 System

3.1 Design constraints

Four specifications drove the design: roughly eight minutes of flight with the compute and camera payload; 720p grayscale capture at 30 Hz with approximately 100 degrees field of view; onboard processing at 20 Hz or better (or, in the abandoned offboard design, streaming under 50 ms); and a real-time interface between the companion computer and the flight controller. These targets balance endurance against payload, visual coverage against per-frame processing cost, and control rate against the available compute.

3.2 Airframe and avionics

The airframe is a custom 3D-printed continuous-carbon-reinforced nylon frame with a 12-inch wheelbase and 5-inch propellers, sized to fit the printer bed while carrying the compute payload with margin; all-up weight is approximately 250 g (Figure 1). Propulsion is four 2006-size 2450 KV

brushless motors on a 4S 900 mAh LiPo, driven by an integrated F7 flight-controller and 45 A four-in-one ESC stack running stock Betaflight. A 2.4 GHz ELRS receiver links a handheld transmitter for piloted flight and emergency override.

The autonomy hardware is a Raspberry Pi Zero 2 W and a Raspberry Pi Camera Module 3 Wide (approximately 100 degrees field of view), capturing 1280×720 grayscale frames at 30 Hz. The camera feeds the Pi over CSI; the Pi and the flight controller communicate over a single UART carrying MSP, the MultiWii Serial Protocol that Betaflight exposes for telemetry and RC override; and the Pi is powered from the flight controller's 5 V rail (Figure 2). The autonomy stack uses no other sensors: no rangefinder, no optical-flow board, and no access to the flight controller's IMU beyond MSP telemetry.

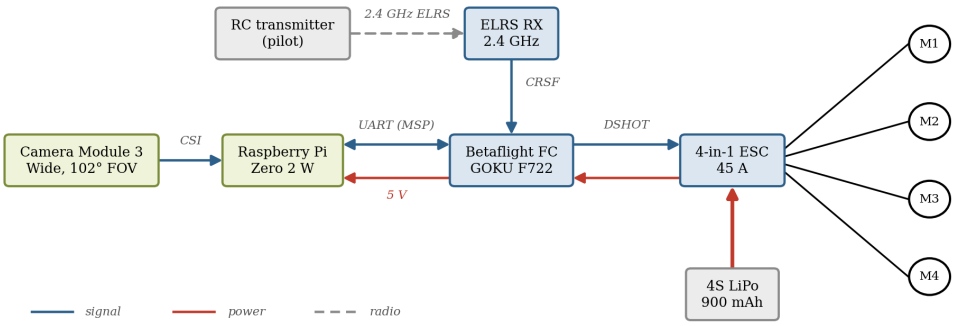

Figure 2: Avionics. Signal wires are blue, power wires are red, and the radio link is dashed. The battery feeds the four-in-one ESC, which powers the flight controller; the flight controller drives the motors over DSHOT, powers the ELRS receiver and the Pi, and exchanges MSP traffic with the Pi over a single UART.

One practical constraint affected testing throughout. The Pi Zero 2 W thermally throttles within about one minute of sustained vision processing when the propellers are idle; in flight, prop wash keeps the package cool, but on the bench it overheats. Timing measurements therefore had to be taken quickly, with forced airflow, or in actual flight.

4 Localization

We describe the three localization designs in the order we attempted them. The first two failed for measurable reasons, and the third is the basis of the closed-loop results in Section 6.

4.1 Attempt 1: offboard processing

The original architecture streamed H.264-encoded 720p video from the Pi to a laptop, ran all perception there, and returned position estimates over Wi-Fi. This design is attractive because the laptop has roughly an order of magnitude more compute and the Pi only encodes. Streaming was eventually made reliable, but measured glass-to-glass latency, from scene change to decoded pixel, exceeded 250 ms against an initial assumption of under 100 ms. Encoding, packetization, Wi-Fi transport, decoding, and buffering each contribute, and none is easily removed in commodity software. Inconsistent clock synchronization between the Pi and the base station further complicated timestamping for filtering. A position loop closed through this path would act on quarter-second-old state, so we abandoned offboard processing rather than attempt to tune around it.

4.2 Attempt 2: markerless visual odometry

The second design estimated ego-motion onboard without environmental infrastructure: FAST corner detection, pyramidal Lucas-Kanade optical flow with forward-backward consistency checking, and frame-to-frame essential-matrix estimation with RANSAC followed by pose recovery. For normalized correspondences $\hat{x} \leftrightarrow \hat{x}'$ between consecutive frames, the essential matrix satisfies the epipolar constraint

$$\hat{x}'^{\top} E \hat{x} = 0, \qquad E = [t]_{\times} R, \tag{1}$$

and decomposes into a rotation $R \in SO(3)$ and a translation direction $t/\|t\|$; the translation magnitude is unobservable from two views, so monocular scale is lost from the start. We evaluate the recovered motion through the rotation magnitude

$$\theta = \arccos\left(\frac{\mathrm{tr}(R) - 1}{2}\right) \tag{2}$$

and through the angle ψ between consecutive unit translation directions, whose density for isotropically random directions is $p(\psi) = \frac{1}{2}\sin\psi$, with median 90 degrees.

The pipeline ran but did not produce usable pose. On a representative handheld sequence, recorded under conditions gentler than flight, the recovered rotations are bimodal: 39 percent of frame pairs return rotations near 180 degrees, reflection-like flips with no physical counterpart, since even a fast handheld pan at 30 Hz rotates the camera by about 3 degrees per frame. The recovered translation directions change by a median of 79 degrees between consecutive frames, close to the 90 degrees expected of uniformly random directions, so successive estimates are nearly uncorrelated. Monocular translation is in any case recovered only up to scale, which nothing in this stack can resolve.

Several factors contribute to the failure. Propeller vibration couples into the rolling-shutter camera and blurs or distorts features within single frames. Indoor walls and floors offer little texture, so the tracked feature set is small, unevenly distributed, and dominated by a few high-contrast edges. At 30 Hz with small inter-frame baselines, the essential-matrix problem is also poorly conditioned, since rotation and translation are nearly indistinguishable for distant, low-parallax features. Together, these effects made the approach unworkable on this platform.

4.3 Adopted design: marker-based planar PnP

The adopted design accepts one piece of infrastructure, a printed 3×5 chessboard with 70 mm squares fixed in the environment. The marker defines the world frame, and localization reduces to planar pose estimation from known correspondences.

A camera-frame point $X = (X_1, X_2, X_3)$ projects to the pixel

$$x = \pi(X) = K\, d(X_1/X_3,\ X_2/X_3)\,, \qquad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

where $d(\cdot)$ is the standard radial-tangential distortion map. Camera intrinsics (K, d) are calibrated once from a chessboard video using standard Zhang-style calibration [19], which minimizes the total reprojection error over all detected corners and per-frame board poses; the fit gives 0.246 px mean reprojection error over the 53 of 87 frames with successful detections. To characterize calibration stability, we bootstrap the calibration over resampled subsets of frames: the focal lengths vary with a standard deviation of about 5.9 px and the principal point with 1.8–2.3 px, so calibration is not the limiting error source downstream.

At runtime, each 1280×720 frame is downsampled by $0.75\times$ and passed to OpenCV's chessboard detector with the fast-check option. Detection at full resolution costs roughly 200 ms per frame on the Pi Zero 2 W, which is unusable at 30 Hz; detection on the downsampled frame costs 40–100 ms, which made onboard processing viable. Detected corners are scaled back to full-frame coordinates and refined with sub-pixel corner localization on the original image (Figure 3), so downsampling reduces detection range but not measurement precision.

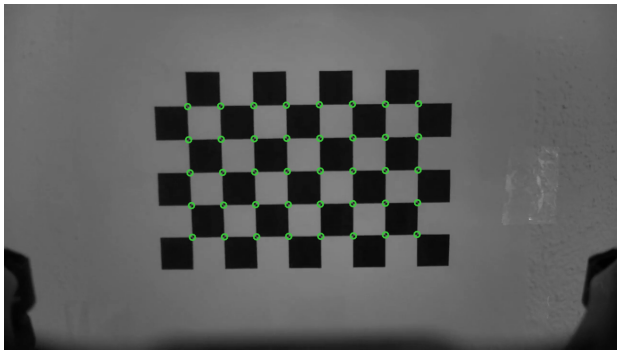

Figure 3: Example detection: the onboard camera viewing a wall-mounted chessboard, with the detected sub-pixel corners overlaid.

Two implementation details had a large effect on reliability. First, OpenCV returns chessboard corners in an order that is ambiguous under 180-degree rotation of the board, and the returned order occasionally flips between frames. We disambiguate by comparing the image x-coordinates of the first and last corners and reversing the array when needed; without this check, the recovered pose intermittently reflects through the marker plane. Second, motion blur breaks sub-pixel refinement well before it breaks detection, so we reduced exposure time and added scene lighting to keep corner localization sharp during flight.

The marker defines the world frame: with square size $s = 70\,\text{mm}$, the inner corners sit at $X_{ij} = (js, is, 0)$ for $i \in \{0, 1, 2\}$, $j \in \{0, \ldots, 4\}$. Given the refined image corners x_{ij}, PnP estimates the camera pose by

$$(R, t) = \arg \min_{R \in SO(3),\, t \in \mathbb{R}^3} \sum_{i,j} \| \pi(R X_{ij} + t) - x_{ij} \|_2^2 . \tag{4}$$

Because the X_{ij} are coplanar, the projection reduces to a homography, $x \simeq K\,[\,r_1 \;\; r_2 \;\; t\,](X_1, X_2, 1)^\top$ with r_k the columns of R, which the IPPE solver [10] exploits to recover the pose in closed form rather than by iterative optimization. The vehicle position in the marker frame is then $p = -R^\top t$, rotated into the drone body frame through the fixed camera mounting rotation. We report the per-frame reprojection error of the accepted solution, the residual of Eq. (4) averaged over the 15 corners; accepted solutions across logged flights show 0.24–0.29 px median reprojection error, and the recovered position is metric, since the square size fixes scale without drift. The cost of this design is locality: the vehicle is localized only while the marker is in view, and detection reliability degrades beyond about 2.5 m of working depth (Section 6.2).

5 Onboard Software Stack

Three threads run in parallel on the Pi, all targeting the 30 Hz camera rate (Figure 4). The vision thread captures a frame, runs the detection and PnP pipeline of Section 4.3, and pushes timestamped positions into a shared buffer. The estimator thread maintains the state estimate at a fixed 33 ms cadence regardless of vision availability. The flight-controller interface thread exchanges RC channel data with the Betaflight FC over MSP.

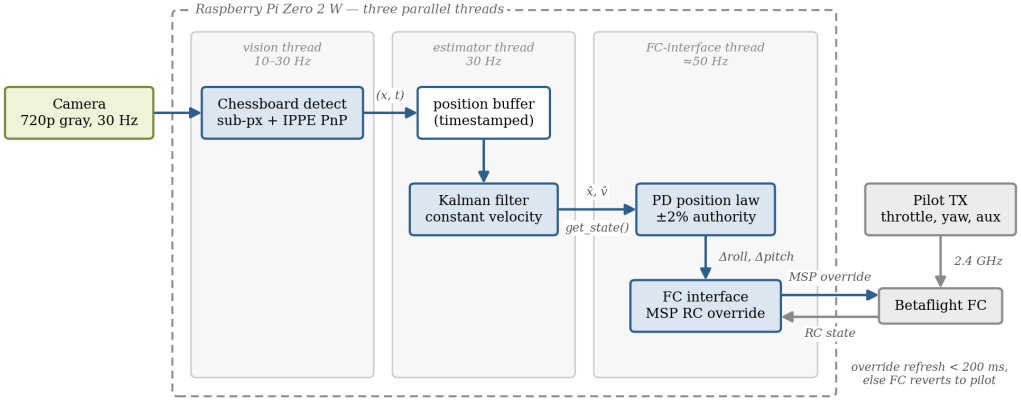

Figure 4: Onboard software. Three parallel threads run on the Pi: the vision thread detects the marker and solves PnP at 10–30 Hz, writing timestamped positions into a shared buffer; the estimator thread runs a Kalman position filter with constant-velocity prediction at a fixed 30 Hz; the FC-interface thread applies the PD law and exchanges RC state and overrides with the flight controller over MSP at about 50 Hz. The override buffer must be refreshed within 200 ms or the flight controller reverts to the pilot's receiver.

5.1 State estimation

The estimator maintains a position estimate $\hat{p} \in \mathbb{R}^3$ with covariance $P \in \mathbb{R}^{3\times3}$ and a velocity estimate $\hat{v} \in \mathbb{R}^3$, updated at a fixed $\Delta t = 33$ ms [20]. Each tick predicts with a constant-velocity model,

$$\bar{p}_k = \hat{p}_{k-1} + \hat{v}_{k-1}\Delta t, \qquad \bar{P}_k = P_{k-1} + Q, \tag{5}$$

with $Q = 0.01\, I$. When a vision measurement z_k is available, the position receives a Kalman update with measurement noise $R_m = 0.5\, I$,

$$K_k = \bar{P}_k \left(\bar{P}_k + R_m\right)^{-1}, \qquad \hat{p}_k = \bar{p}_k + K_k \left(z_k - \bar{p}_k\right), \qquad P_k = \left(I - K_k\right) \bar{P}_k, \tag{6}$$

and the velocity estimate is driven by the scaled innovation,

$$\hat{v}_k = \hat{v}_{k-1} + \frac{\alpha}{\Delta t} K_k \left(z_k - \bar{p}_k\right), \qquad \alpha = 0.1. \tag{7}$$

This is a position-state Kalman filter with a first-order velocity estimate rather than a joint position-velocity filter: the small α heavily smooths $\hat{v}$, which suppresses the depth jitter of individual PnP solutions at the cost of phase lag in the velocity signal, a cost that reappears in the closed-loop analysis of Section 6.4. When no measurement is available the filter coasts on the prediction alone. Vision measurements arrive at an irregular 10–30 Hz depending on detection success. The relatively large R_m reflects that individual PnP solutions, while sub-pixel in reprojection, still jitter in 3D through the depth-sensitive geometry of a small planar target. At startup, the filter averages the first ten vision measurements to define the origin, so the hold target is the pose at which autonomy was engaged.

5.2 Control and flight-controller interface

The position controller is a PD law per horizontal axis. With position error $e = p_{\text{ref}} - \hat{p}$ (zeroed inside a 1 cm deadband) and the filtered velocity $\hat{v}$, the raw command for each axis is

$$\tilde{u}_k = R_u \left(K_P\, e_k - K_D\, \hat{v}_k\right), \qquad K_P = 3.0, \quad K_D = 20.0, \quad R_u = 20\ \mu\text{s}, \tag{8}$$

an offset in RC pulse width about the trimmed stick center. The applied command passes through a slew-rate limit of $\delta = 10$ μs per control step and a hard authority limit of $a = 20$ μs:

$$u_k = \mathrm{sat}_{\pm a}(u_{k-1} + \mathrm{sat}_{\pm\delta}(\tilde{u}_k - u_{k-1}))\,, \tag{9}$$

where $\mathrm{sat}_{\pm c}(x) = \max(-c, \min(c, x))$. The authority limit is roughly two percent of full stick range, and the command saturates once $|K_P e - K_D \hat{v}|$ exceeds 1, that is, at 33 cm of position error with zero estimated velocity. A full PID law was attempted first and abandoned: on this platform the integral term mostly integrated the oscillation described in Section 6, and the simpler PD law was easier to tune.

The interface thread continuously reads the pilot's RC channels from the FC over MSP, mixes in the autopilot roll and pitch offsets when a transmitter auxiliary switch engages autonomy, and writes the resulting channel set back via the MSP RC-override command. Two properties of this mechanism shaped the design. First, Betaflight reverts to the hardware receiver if the override buffer is not refreshed within roughly 200 ms, so the loop rewrites the buffer continuously, with unmodified pilot inputs when autonomy is off. The staleness timeout also acts as a watchdog: if the companion computer hangs, the pilot's transmitter regains authority automatically. Second, a read-modify-write cycle costs about 20 ms over the UART, which bounds the interface near 50 Hz; this is adequate for the 30 Hz estimate stream but adds to the total loop delay. Throttle and yaw are always passed through from the pilot; only roll and pitch are overridden.

6 Experimental Results

6.1 Setup and metrics

All results come from logged flights of the physical vehicle in an indoor space with the chessboard marker fixed to a wall, plus bench recordings for calibration. We log raw video onboard and re-run the exact onboard pipeline on the logs for analysis, so the reported detection and accuracy statistics reflect the flight code path rather than a separate reimplementation. We evaluate: calibration quality (reprojection error, bootstrapped intrinsic spread); PnP performance (detection rate versus distance to the marker, dropout durations, reprojection error of accepted solutions); timing (per-frame detection cost at full and reduced resolution, MSP cycle time, glass-to-glass latency of the abandoned streaming path); and closed-loop precision (XY RMS error about the held position, per-axis standard deviation, and error spectra). No external ground truth was available, so closed-loop error is measured relative to the mean of the held position; this measures precision rather than absolute accuracy.

6.2 Marker detection and PnP accuracy

Detection is reliable close in and collapses with range. Pooled across the four marker-visible logged flights, the per-frame detection rate stays above 95 percent out to about 2 m, drops past 2.5 m, and falls below 10 percent by 2.9 m (Figure 5a), where the 70 mm squares approach the detector's resolution limit in the downsampled image. This curve bounds the usable operating volume to roughly a 2.5 m radius around the marker; the one flight that operated near and beyond that boundary detected the marker in only 48 percent of frames, against 92–100 percent for the others. Dropouts within range are mostly under 200 ms and the filter coasts through them.

When the marker is detected, the measurement is accurate. Median reprojection error of accepted solutions is 0.24–0.29 px in every flight (Figure 5b). For a planar target the depth estimate is the most sensitive coordinate: a corner error of σ_{px} on a target of extent L at distance z propagates to a depth error of approximately

$$\sigma_z \approx \frac{z^2}{fL}\,\sigma_{\mathrm{px}}, \tag{10}$$

with lateral errors smaller by a factor of about z/L. At $z = 2$ m with $f \approx 556$ px, corner span $L \approx 0.28$ m, and $\sigma_{\mathrm{px}} = 0.27$ px, this gives $\sigma_z \approx 7$ mm: the position measurement is accurate to

about a centimeter through the working volume, well below the closed-loop error we observe in Section 6.3.

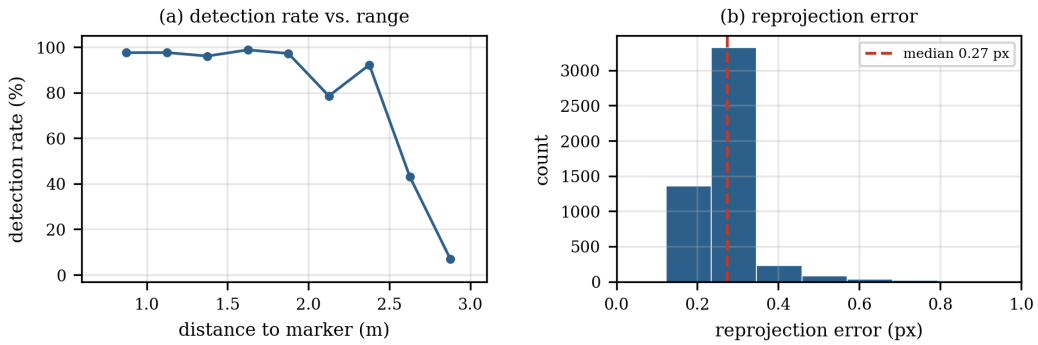

Figure 5: Marker detection and PnP accuracy, pooled across the four marker-visible flights. (a) Per-frame detection rate against distance to the marker (distance taken from the filtered estimate): reliable to about 2 m, collapsing beyond 2.5 m, which bounds the operating volume. (b) Reprojection error of accepted solutions; the 0.27 px median corresponds to centimeter-level position noise.

6.3 Closed-loop position hold

Figure 6 shows frames from a position-hold flight: the vehicle holds station near the wall-mounted marker with roll and pitch under autonomous control.

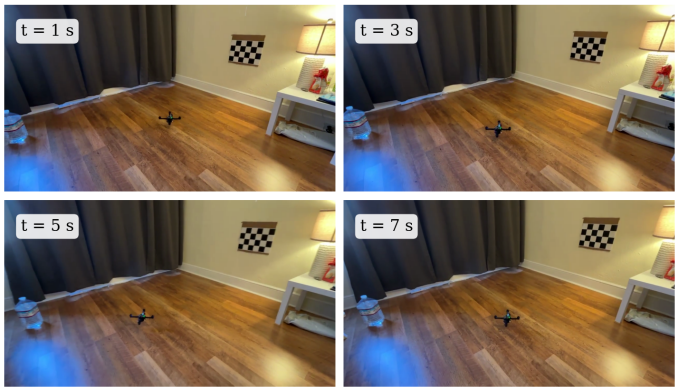

Figure 6: Frames spanning six seconds of an autonomous position hold. The vehicle holds station near the wall-mounted marker with roll and pitch under autonomous control.

Figure 7a plots the logged horizontal error of an 80 s autonomous hold at approximately 30 Hz. The vehicle remains bounded throughout: XY RMS error is approximately 36 cm, with per-axis standard deviations of 23 cm in x, 28 cm in y, and 10 cm in z (z is pilot-held through throttle passthrough, so its smaller spread reflects the pilot rather than the autopilot).

The error also has a regular temporal structure. The horizontal error oscillates at near-constant amplitude rather than wandering, and its spectrum concentrates in a single peak at 0.26 Hz in both axes (Figure 7b): the residual is one coherent oscillation, not broadband noise. The loop runs close to its stability margin, and the oscillation could not be tuned away: raising the gains increased its frequency, while lowering them slowed it without usefully shrinking it, since the heavily filtered velocity estimate already weakens the damping term. The oscillation is a limit cycle set jointly by

the control gains and the loop's accumulated delay — the gains needed for adequate authority are about as high as the latency allows before the loop oscillates.

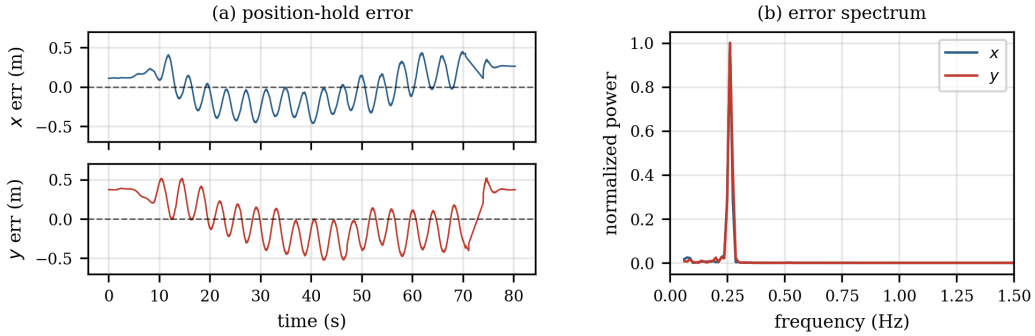

Figure 7: Closed-loop position hold over an 80 s flight. (a) Horizontal position error about the held position: bounded (36 cm XY RMS) but oscillating at near-constant amplitude. (b) The position-error spectrum concentrates in a single peak at 0.26 Hz in both axes; the residual is one coherent oscillation rather than broadband noise, a limit cycle set by the controller gains and the loop's latency.

6.4 Latency budget

The latency budget explains why the loop oscillates at the gains it needs. Exposure and readout, the $0.75\times$ downsample, detection (40–100 ms), sub-pixel refinement and PnP, the filter's 33 ms cadence, the roughly 20 ms MSP read-modify-write, and the flight controller's own response each contribute, and the sum exceeds 100 ms before the airframe begins to move. For comparison, the abandoned streaming path alone measured over 250 ms glass-to-glass, and full-resolution onboard detection alone cost about 200 ms per frame.

A linearized model makes the mechanism explicit. Near hover, a small commanded tilt ϕ produces horizontal acceleration $\ddot{x} \approx g\phi$, and the inner attitude loop is fast relative to the position loop, so the plant from RC offset to position is approximately a double integrator $G(s) = gk_u/s^2$, where k_u is the RC-to-tilt gain of the flight controller's angle mode. With the PD law $C(s) = R_u(K_P + K_D s)$ and an aggregate loop delay τ, the loop transfer function is

$$L(s) = R_u\,(K_P + K_D s)\,\frac{gk_u}{s^2}\,e^{-s\tau}, \tag{11}$$

and a sustained oscillation sits at the phase-crossover frequency ω_c where the delay cancels the PD phase lead:

$$\arctan\!\left(\frac{K_D\,\omega_c}{K_P}\right) = \omega_c\,\tau. \tag{12}$$

At the measured $\omega_c = 2\pi(0.26\text{ Hz}) = 1.6$ rad/s and $K_D/K_P = 6.7$ s, Eq. (12) implies an effective delay of roughly 0.9 s, several times the measured transport delay. The difference is consistent with the phase lag that the model omits: the heavily smoothed velocity estimate of Section 5.1 ($\alpha = 0.1$) delays the damping signal well beyond the raw pipeline latency, and the slew-rate limit adds amplitude-dependent lag of its own. The amplitude of the oscillation is set by the authority limit: as the orbit grows, the describing function of the saturation in Eq. (9) reduces the effective loop gain until the oscillation neither grows nor decays, and the saturation threshold of 33 cm matches the observed 36 cm RMS orbit. The conservative authority limits are therefore what keep the oscillation bounded, and most of our tuning effort went into bounding authority rather than shaping gains.

7 Discussion

The system shows that a complete vision-based position-hold loop fits on a commodity stack: a printed target, a calibrated consumer camera, and well-established planar PnP deliver sub-pixel measurements and drift-free metric position on a low-cost processor, and an unmodified hobby flight controller accepts the resulting corrections over an interface it already exposes. The closed-loop precision achieved within that envelope is limited by the loop's delay, which caps how aggressively the controller can be tuned before it oscillates: the system localizes itself to centimeters but acts on estimates that are more than 100 ms old.

In retrospect, the three most consequential engineering decisions were moving computation onboard to escape streaming latency, bounding actuator authority so that the oscillation stayed bounded, and handling vision-side robustness details (corner-order disambiguation, exposure management) that determine whether measurements exist at all. None of these was a planned focus at the proposal stage.

The two failed designs carry lessons for this hardware class. Offboard streaming can be made to work as a video link, but the resulting latency makes it unsuitable as the sensing path for inner-loop control, and the failure only becomes visible once glass-to-glass latency is measured rather than assumed. Markerless monocular odometry on a vibrating airframe in texture-poor rooms fails through the combination of vibration, blur, weak texture, and scale ambiguity. Each of these failure modes is documented in prior work; on this platform they occurred together, and a single fiducial removed all of them at the cost of a bounded operating volume. Both designs appeared viable at the proposal stage, and both failures became clear within days once we measured latency and pose scatter directly.

8 Limitations and Future Work

The system localizes only within sight of a single fixed marker, and reliably only within about 2.5 m of it, which limits the operating volume to a small region around the marker. Closed-loop precision is at the tens-of-centimeters level, and the analysis in Section 6 attributes that imprecision to loop delay and the gains it forces rather than to a component that could be upgraded in isolation. Altitude and yaw are pilot-held, so the demonstrated autonomy is a two-axis position hold rather than full waypoint flight. Without external ground truth, our error metrics measure precision about the held mean rather than accuracy against an absolute reference.

Future work follows from the latency analysis. The first priority is reducing loop delay: a faster camera path, tracking small regions of interest between full detections, and delay-compensating control such as a Smith predictor or forward-predicting the error by the measured pipeline delay. Closing throttle and yaw would complete the control authority. Replacing the single chessboard with a field of AprilTag or ArUco markers would extend the operating volume at little hardware cost. Removing fiducials entirely would require onboard mapping, and our markerless results suggest that this step needs either inertial fusion at high rate or a global-shutter camera, both of which raise the hardware floor.

9 Conclusion

We presented AVIAN, an autonomous indoor position-hold system built from a 250 g hobby-grade quadrotor, a Raspberry Pi Zero 2 W, one camera, and one printed chessboard. Two earlier designs failed for measurable reasons: offboard streaming exceeded the latency budget, and markerless odometry broke down under vibration, weak texture, and scale ambiguity. The marker-based planar-PnP design ran fully onboard at 10–30 Hz and flew the vehicle autonomously. The vision stack delivers sub-pixel, metrically scaled pose, and the vehicle holds position within a bounded 36 cm RMS region, oscillating in a 0.26 Hz limit cycle: the gains needed for adequate authority sit near the

stability margin imposed by the loop's end-to-end delay. Reducing that delay, which would allow higher gains before the loop oscillates, is the most direct path to tighter precision on this platform.

References

[1] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini. A 64-mW DNN-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal*, 6(5):8357–8371, 2019.

[2] L. Lamberti, L. Bellone, L. Macan, E. Natalizio, F. Conti, D. Palossi, and L. Benini. Distilling tiny and ultra-fast deep neural networks for autonomous navigation on nano-UAVs. *IEEE Internet of Things Journal*, 2024.

[3] L. Lamberti, G. Rutishauser, F. Conti, and L. Benini. Combining local and global perception for autonomous navigation on nano-UAVs. *arXiv preprint arXiv:2403.11661*, 2024.

[4] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407, 2011.

[5] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[6] M. Bertoni, S. Michieletto, R. Oboe, and G. Michieletto. Indoor visual-based localization system for multi-rotor UAVs. *Sensors*, 22(15):5798, 2022.

[7] F. Wang, Y. Zou, C. Zhang, J. Buzzatto, M. Liarokapis, E. Del Rey Castillo, and J. B. P. Lim. UAV navigation in large-scale GPS-denied bridge environments using fiducial marker-corrected stereo visual-inertial localisation. *Automation in Construction*, 156, 2023.

[8] S. Raxit, S. B. Singh, and A. A. R. Newaz. YoloTag: Vision-based robust UAV navigation with fiducial markers. In *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2024.

[9] G. Bradski. The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 2000.

[10] T. Collins and A. Bartoli. Infinitesimal plane-based pose estimation. *International Journal of Computer Vision*, 109(3):252–286, 2014.

[11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[12] T. Qin, P. Li, and S. Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[13] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022.

[14] D. Falanga, S. Kim, and D. Scaramuzza. How fast is too fast? The role of perception latency in high-speed sense and avoid. *IEEE Robotics and Automation Letters*, 4(2):1884–1891, 2019.

[15] A. Khalil and J. Kwon. Nonlinear performance degradation of vision-based teleoperation under network latency. *arXiv preprint arXiv:2603.06850*, 2026.

[16] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 7(67):eabl6259, 2022.

[17] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 37–42, 2017.

[18] Betaflight Project. Betaflight flight controller firmware, 2025. URL https://betaflight.com. Accessed 2026-06-11.

[19] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[20] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.