[0001] COMPUTER VISION-BASED GUN CONTROLLER WITH INTEGRATED BALLISTICS AND FEEDBACK

[0002] FIELD OF THE INVENTION

[0003] The present invention relates to firearm control systems employing computer vision and inertial measurement, and more particularly, to integrated camera-based targeting controllers that perform real-time hit detection using calibrated ballistic models, Kalman filtering, and trigger-synchronized computations. Optional features include embedded microcontroller implementations, mobile device integration, and multimodal feedback mechanisms.

[0004] BACKGROUND OF THE INVENTION

[0005] Traditional firearm engagement systems rely on external tracking infrastructure, passive targeting displays, or disconnected ballistic calculators. Shooters must mentally integrate target observations, weapon orientation, and ballistic effects, introducing delays and human error. Simulation systems for training and sport shooting have evolved to include motion capture, drone tracking, and score feedback, but few integrate low-latency, on-device vision processing directly into the firearm controller itself. There exists a need for a self-contained gun controller that captures target imagery in real time, performs geometric and ballistic calculations on-device, determines hit outcomes with low latency, and provides immediate multimodal feedback all within a unified hardware-software platform.

[0006] SUMMARY OF THE INVENTION

[0007] The disclosed computer vision gun controller integrates a camera, inertial measurement unit (IMU), Kalman state estimator, ballistics engine, and feedback mechanisms into a unified system. The controller continuously captures video frames, detects targets via image processing, estimates target position and motion in the inertial frame via rotation matrix transforms, and maintains a high-fidelity state estimate of the target's position, velocity, and acceleration. When the trigger is pulled, the system reads the estimated target state, computes virtual projectile initial conditions accounting for barrel orientation and rotation-induced transverse velocity, solves for the time of closest approach to the target along the projectile's velocity vector, evaluates whether the miss distance at that moment is within the projectile's spread radius, and transmits a hit or miss result to feedback actuators. The system supports two primary embodiments: a mobile device variant using ARKit and CoreML for sensor fusion and neural network-based target detection, and an embedded microcontroller variant (ESP32-S3 + BNO085 IMU) supporting direct sensor access and lightweight on-device inference.

[0008] BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 is a top-level block diagram of the gun controller system showing camera, IMU, state estimator, ballistics engine, and feedback subsystems.
FIG. 2 is a sequence diagram showing the per-frame loop: camera capture, target detection, geometry computation, inertial transform, and Kalman update.
FIG. 3 is a flow diagram of the trigger-time computation phase: state extraction, projectile initial conditions, relative state, intercept time calculation, and hit test.
FIG. 4 is a schematic of the hit detection geometry, showing target position, projectile center, miss distance, and spread radius.
FIG. 5 is a block diagram of the iPhone/ARKit embodiment, showing ARKit session, Vision/CoreML detection layer, Kalman filter, ballistics engine, and feedback outputs.
FIG. 6 is a block diagram of the embedded ESP32-S3/BNO085 embodiment, showing I2C sensor interconnects, state estimation module, ballistics computation, and audio/haptic output.
FIG. 7 is a detail view of the inertial frame transform, illustrating barrel frame coordinates, rotation matrix M(t), and the mapping to world coordinates.
FIG. 8 is a detailed sequence diagram of an embodiment of the system illustrating the interaction between camera capture, detection, Kalman update, trigger event, and ballistics computation with feedback timing.

[0010] DETAILED DESCRIPTION OF THE INVENTION

[0011] Referring to FIG. 1, the gun controller system comprises a camera (100) rigidly mounted to the firearm barrel, an inertial measurement unit (IMU) (102) including a gyroscope and accelerometer, a central processor (104) executing vision and signal processing algorithms, a Kalman filter module (106) maintaining the target state estimate, a ballistics engine (108) computing projectile trajectories and hit outcomes, and a feedback subsystem (110) delivering haptic, audio, and optional visual cues. The camera continuously captures frames at a fixed rate synchronized with the IMU sampling.

As shown in FIG. 2, during each camera frame capture at time ti, the system performs six processing steps. Step 1: Camera Frame Capture. A frame buffer (112) containing the pixel values is delivered by the camera hardware. Step 2: Target Detection. Image processing, optionally using a trained neural network detector, locates the target in the image (114), yielding the centroid pixel coordinates (ui, vi) and apparent diameter si in pixels. Step 3: Geometry Computation (116). From the known physical target diameter S, camera focal length f, and apparent pixel diameter si, the distance is computed as di = Sf/si. The pixel offset from the image center (Δui = ui − u0, Δvi = vi − v0) defines a unit ray direction êi in the barrel (camera) frame via ê_i = (Δu_i, Δv_i, f)^T / ‖·‖. The target position in barrel frame coordinates is r^b_i = di · êi. Step 4: Inertial Frame Transform (118). The barrel frame position is rotated to the inertial frame using the rotation matrix M(ti) derived from integrating the IMU angular velocity history. The

inertial-frame observation is ri = M(ti) r^b_i.

Step 5: Kalman Filter Update (120). The observation ri is fed into a Kalman filter whose internal state is the target's position, velocity, and acceleration ($\hat{r}$, $\hat{v}$, $\hat{a}$) in the inertial frame. The filter uses a constant-acceleration process model; its state transition matrix is block-tridiagonal with identity blocks and timestep-dependent off-diagonal blocks. With each new observation, the filter refines the state estimate, smoothing sensor noise and interpolating over missed detections. Step 6: Trigger Check (122). If the trigger has not been pulled, control returns to Step 1 for the next frame. If the trigger is pulled at time t0, execution proceeds to Phase 2 (trigger-time computation).

Referring to FIG. 3, when the trigger is pulled, the system executes the trigger-time computation in Phase 2. Step 7: Read Kalman State. The current best estimate ($\hat{r}0$, $\hat{v}$, $\hat{a}$) is extracted from the Kalman filter. Step 8: Projectile Initial Conditions (124). The projectile departs from the muzzle position rm (a fixed offset from the pivot point along the barrel axis, determined by calibration). Its initial position is p0 = rm. Its initial velocity accounts for both muzzle velocity vp along the barrel direction $\hat{b}$ and transverse velocity induced by the barrel's rotation: $V = vp\,\hat{b} + \omega \times rm$, where ω is the barrel angular velocity at t0. Step 9: Relative State (126). The problem is reformulated in relative (target-minus-projectile) coordinates: $\delta 0 = \hat{r}0 - p0$, $\delta v = \hat{v} - V$, $\delta a = \hat{a} - g$, where g is gravitational acceleration.

Step 10: Intercept Time (128). The system computes the time τ^* when the projectile's expanding sphere passes through the target's along-track position. By requiring that $\Delta(\tau^*) \cdot V = 0$ (the separation vector is perpendicular to the projectile velocity), a quadratic equation is derived: $(1/2)\,\alpha\,\tau^{*2} + \beta\,\tau^* + \gamma = 0$, where $\alpha = \delta a \cdot V$, $\beta = \delta v \cdot V$, $\gamma = \delta 0 \cdot V$. The smallest positive root is $\tau^* = (-\beta + \sqrt{(\beta^2 - 2\alpha\gamma)}) / \alpha$, or $\tau^* = \gamma / (-\beta)$ if $\alpha \approx 0$. Step 11: Miss Distance (130). At time τ^*, the perpendicular miss distance is $\rho = \|\Delta(\tau^*)\|$ and the projectile travel distance is $\ell = vp\,\tau^*$. Step 12: Hit Test (132). The hit condition is $\rho \le R(\ell)$, where $R(\cdot)$ is the projectile's spread radius as a function of distance traveled. If ℓ exceeds the projectile's maximum effective range, the result is a miss regardless of ρ. Otherwise, HIT is returned if $\rho \le R(\ell)$, and MISS is returned if $\rho > R(\ell)$ or no positive root exists.

Referring to FIG. 4, the hit detection geometry is illustrated. The target (134) at estimated position $\hat{r}0$ is modeled as a spherical volume of radius Rtarget (or treated as a point for simplified embodiments). The projectile center follows a ballistic trajectory $p(\tau) = p0 + V\tau + (1/2)g\tau^2$. At intercept time τ^*, the projectile has expanded to a cone or cylinder of radius $R(\ell)$, and the perpendicular distance from the target's track to the projectile's flight line is computed (136). This geometry naturally accounts for moving targets, barrel swings, and projectile spread.

FIG. 8 shows a detailed sequence diagram of an embodiment of the system illustrating the interaction between camera capture, detection, Kalman update, trigger event, and ballistics computation with feedback timing.

[0012] CLAIMS

[0013] 1. A computer vision gun controller comprising:

a camera configured to capture frames at a fixed frame rate, the camera rigidly mounted relative to a firearm barrel;

an inertial measurement unit comprising a gyroscope and accelerometer, configured to measure barrel angular velocity and acceleration;

a target detection module configured to receive each frame and output a target centroid in pixel coordinates and an apparent pixel diameter;

a geometry computation module configured to compute a distance estimate from the apparent diameter, a unit ray direction from pixel offset coordinates, and a position in the barrel frame;

an inertial frame transformer configured to rotate barrel-frame positions into an inertial coordinate system using a rotation matrix derived from the angular velocity history;

a Kalman filter configured to maintain an estimate of the target's position, velocity, and acceleration in the inertial frame, with a constant-acceleration process model and per-frame update steps;

a trigger input interface;

a ballistics engine configured to, upon trigger activation, read the target state estimate, compute projectile initial conditions including muzzle velocity and rotation-induced transverse velocity, compute a relative state, solve for an intercept time at which the projectile's expanding sphere passes through the target's along-track position, compute a perpendicular miss distance at the intercept time, and compare the miss distance to a spread radius function to determine a hit or miss outcome;

a feedback subsystem configured to deliver the hit or miss outcome to one or more output actuators.

2. The gun controller of claim 1, wherein the target detection module comprises a neural network detector trained on a dataset of target images.

3. The gun controller of claim 1, wherein the target detection module comprises color segmentation followed by contour fitting.

4. The gun controller of claim 1, wherein the Kalman filter uses a 9×9 state covariance matrix and implements per-frame prediction and update steps with detection confidence weighting.

5. The gun controller of claim 1, wherein the inertial frame transformer integrates the angular velocity from the gyroscope over time to maintain the rotation matrix M(t) in SO(3).

6. The gun controller of claim 1, wherein the ballistics engine computes projectile initial conditions as $p_0 = r_m$ and $V = v_p \hat{b} + \omega \times r_m$, where r_m is the muzzle offset, v_p is the muzzle speed, $\hat{b}$ is the barrel direction unit vector, and ω is the barrel angular velocity.

7. The gun controller of claim 1, wherein the ballistics engine solves the intercept time quadratic equation $(1/2)\,\alpha\,\tau^{*2} + \beta\,\tau^{*} + \gamma = 0$ where $\alpha = \delta a \cdot V$, $\beta = \delta v \cdot V$, $\gamma = \delta 0 \cdot V$, using the smallest positive real root.

8. The gun controller of claim 1, wherein the feedback subsystem comprises a haptic actuator configured to produce a recoil pulse on trigger activation.

9. The gun controller of claim 1, wherein the feedback subsystem further comprises an audio output configured to play a distinct sound for hit versus miss outcomes.

10. The gun controller of claim 1, wherein the feedback subsystem further comprises a visual overlay on the camera feed indicating the computed impact point and spread circle.

11. A method for real-time hit detection in a gun controller, comprising:

continuously capturing camera frames synchronized with inertial measurement unit samples;

detecting a target in each frame and extracting centroid and apparent diameter in pixel coordinates;

computing a distance estimate, ray direction, and barrel-frame position for the target;

applying an inertial frame rotation derived from integrated gyroscope data to transform the target position to world coordinates;

updating a Kalman filter with the transformed observation to refine an estimate of target position, velocity, and acceleration;

upon trigger activation, reading the current target state estimate and barrel angular velocity;

computing virtual projectile initial conditions from the muzzle position, muzzle velocity, and barrel rotation;

formulating a relative state between target and projectile and solving for an intercept time at which the separation vector is perpendicular to the projectile velocity;

evaluating the perpendicular miss distance at the intercept time and comparing it to a spread radius function;

determining a hit or miss outcome and providing feedback to the user.

12. The method of claim 11, wherein the target detection step uses a trained neural network classifier.

13. The method of claim 11, wherein the Kalman filter update incorporates a detection

confidence score to weight the observation contribution.

14. The method of claim 11, wherein the feedback subsystem transmits hit/miss outcomes to haptic and audio output devices with a latency below 100 milliseconds from trigger activation.

[0014] EMBODIMENT OF THE INVENTION

[0015] Embodiment A: iPhone/ARKit Implementation. In this embodiment, the gun controller is integrated into an iPhone running a custom shooter application. The camera hardware provides 60 fps video capture and Exif metadata with precise exposure timestamps. ARKit (ARSession with ARWorldTrackingConfiguration) delivers sensor fusion, producing a 4×4 device pose matrix in a stable world coordinate frame. The device's 6-DOF pose is updated at 60 Hz and includes both position and orientation (attitude) derived from visual-inertial SLAM; the gyroscope and accelerometer are accessed via direct CMMotionManager queries at up to 100 Hz when higher temporal resolution is needed.

Extraction of barrel state [160] proceeds as follows. From ARKit's pose matrix, the barrel direction $\hat{b}$ is read as the negative z-axis column of the 3×3 rotation submatrix (by Apple's convention, −z is the camera's forward direction). The barrel angular velocity ω [162] is computed by numerically differentiating the rotation matrix between consecutive frames at 100 Hz, or by supplementing ARKit's quaternion-based orientation with direct CMMotionManager gyroscope readings. The muzzle position rm [164] is the translation column of the pose matrix plus a fixed calibration offset (mm in phone-local frame). Camera intrinsics f, u0, v0 [166] are provided automatically by ARKit and require no manual calibration.

Target detection [168] is handled by Apple's Vision framework (VNCoreMLRequest or VNDetectContoursRequest). For high-contrast targets (e.g. orange disc against sky), a CIFilter-based color threshold followed by VNDetectContoursRequest extracts contours; a circle-fitting algorithm yields centroid (ui, vi) and diameter si. For complex or varied scenes, a lightweight neural network (MobileNet-SSD or YOLOv8-nano, exported to CoreML) runs on the Neural Engine at ~3–8 ms per frame. The detector outputs a bounding box; the centroid and diameter are computed from the box dimensions.

The Kalman filter [170] is implemented in Swift using SIMD types (simd_float3x3, simd_float3) for efficient matrix operations. The state is a 9×1 vector: [r^T, v^T, a^T]^T. The process model assumes constant acceleration with a timestep $\Delta t = 1/60$ s. The predict step applies the standard kinematic matrix; the update step incorporates observations with a measurement noise covariance weighted by detection confidence. The implementation is approximately 50 lines of Swift.

Upon trigger activation (via a paired Bluetooth Low Energy (BLE) button [172]), the ballistics

engine [174] executes. It reads the Kalman state, computes projectile initial conditions using the current muzzle position and velocity, formulates the relative state, solves the quadratic for τ^*, evaluates the miss distance, and tests against the spread radius. The computation (dot products, a square root, and a comparison) completes in < 1 ms. Feedback [176] is delivered via CoreHaptics (haptic engine) for a recoil pulse, AVFoundation for audio (a distinct WAV file for hit vs miss), and optionally a SpriteKit or RealityKit overlay on the camera feed showing the predicted impact point and spread circle.

Embodiment B: Embedded ESP32-S3 / BNO085 Implementation. In this embodiment, the gun controller is a standalone microcontroller unit (MCU) mounted directly on the firearm. The Seeed XIAO ESP32-S3 [180], a 240 MHz dual-core processor with 8 MB PSRAM, serves as the main controller. It connects via I2C [182] to a CEVA BNO085 9-axis IMU [184] (3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer integrated in a single package with an onboard Cortex-M0+ running sensor fusion). The IMU provides calibrated quaternion-based orientation estimates at up to 100 Hz, eliminating the need for manual gyroscope bias correction, gravity subtraction, or quaternion integration.

A camera module [186] (OV2640 or OV5640, 2MP, 320×240 QVGA or higher) connects to the ESP32-S3 via I2S/parallel bus, capturing frames at 30–60 fps. Target detection [188] is performed using a lightweight embedded model: either a color-threshold approach for simple targets, or a quantized MobileNet or YOLO variant (TensorFlow Lite) running on the MCU. Inference latency is 10–50 ms depending on image resolution and model complexity.

The Kalman filter [190] is implemented in C/C++ using fixed-point or single-precision floating-point SIMD operations. State and covariance matrices are stored in PSRAM; the predict/update cycle runs in ~5 ms per frame. The inertial frame rotation is computed from the BNO085's quaternion output, avoiding the need for manual integration.

Trigger input [192] is read via a GPIO pin polling a tactile button or an analog trigger-force sensor (e.g., Interlink FSR 402). Upon trigger detection, the ballistics engine [194] reads the Kalman state and current IMU quaternion, computes projectile conditions, performs the intercept solve, and outputs a HIT or MISS result. Feedback [196] is delivered to a piezo speaker [198] playing audio samples (WAV files stored in ESP32 flash or external PSRAM) and to a haptic linear resonant actuator [200] driven by a PWM signal, providing tactile recoil feedback. Optional serial logging [202] transmits ballistic details to a ground control station via USB or Bluetooth for analytics.

Both embodiments share the same mathematical framework for hit detection (Kalman filtering, intercept geometry, spread radius comparison) but differ in sensor integration and processing substrate. The iPhone variant leverages high-performance sensor fusion and neural network

inference via dedicated hardware (Neural Engine), while the embedded variant provides portability and direct firearm integration with minimal external infrastructure.

[0016] DRAWING SHEETS

FIGURE 1 - System Block Diagram

FIG. 1 is a top-level block diagram of the gun controller system showing camera, IMU, state estimator, ballistics engine, and feedback subsystems.

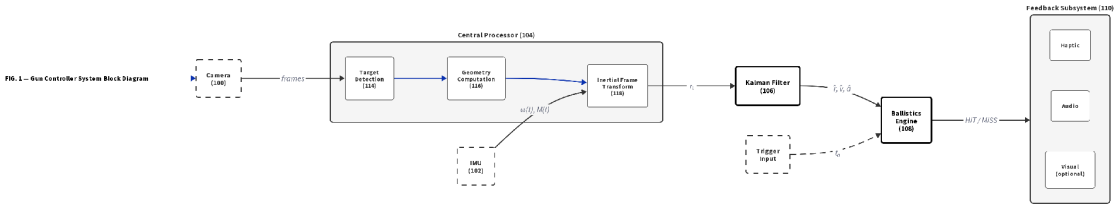

FIGURE 2 - Per-Frame Processing Sequence

FIG. 2 is a sequence diagram showing the per-frame loop: camera capture, target detection, geometry computation, inertial transform, and Kalman update.

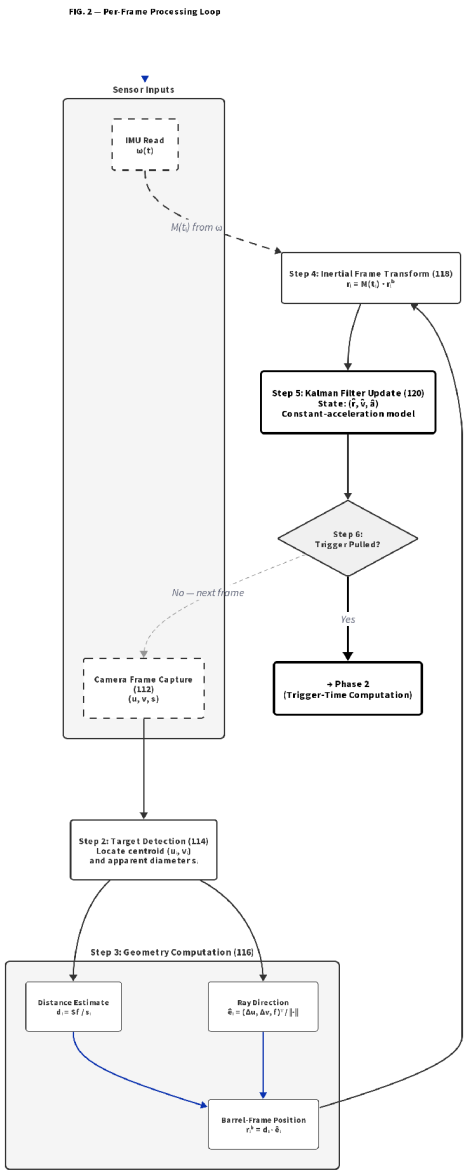

FIGURE 3 - Trigger-Time Computation Flow

FIG. 3 is a flow diagram of the trigger-time computation phase: state extraction, projectile initial conditions, relative state, intercept time calculation, and hit test.

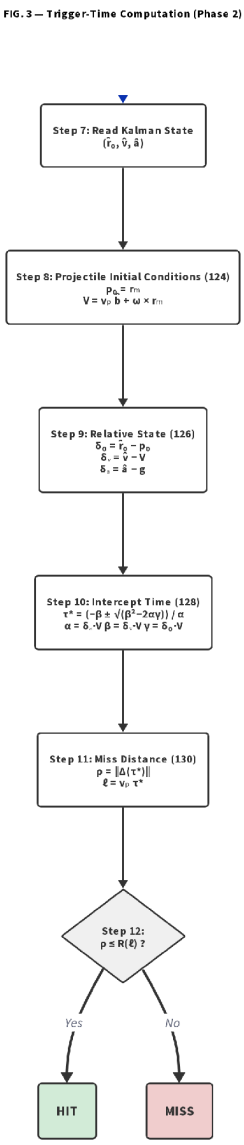

FIGURE 4 - Hit Detection Geometry

FIG. 4 is a schematic of the hit detection geometry, showing target position, projectile center, miss distance, and spread radius.

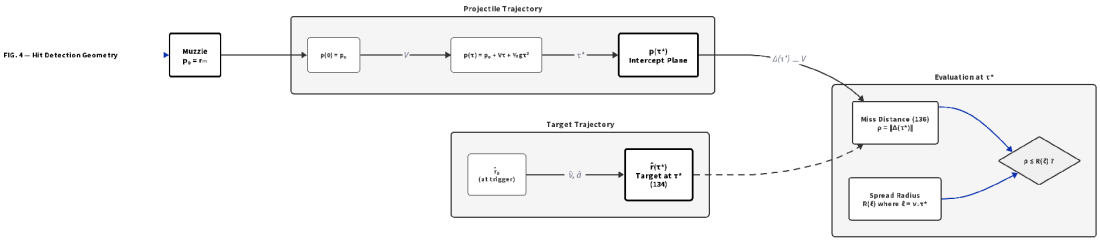

FIGURE 5 - iPhone/ARKit Embodiment Architecture

FIG. 5 is a block diagram of the iPhone/ARKit embodiment, showing ARKit session, Vision/CoreML detection layer, Kalman filter, ballistics engine, and feedback outputs.

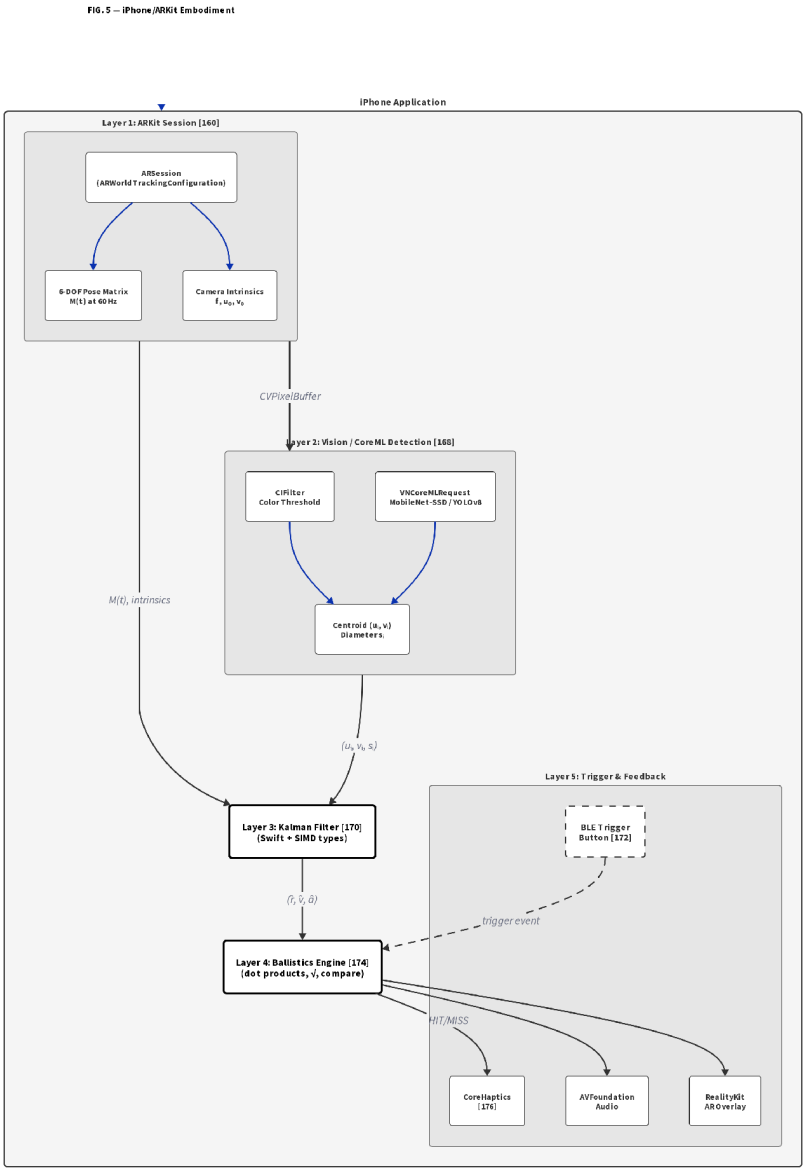

FIGURE 6 - Embedded ESP32-S3/BNO085 Embodiment

FIG. 6 is a block diagram of the embedded ESP32-S3/BNO085 embodiment, showing I2C sensor interconnects, state estimation module, ballistics computation, and audio/haptic output.

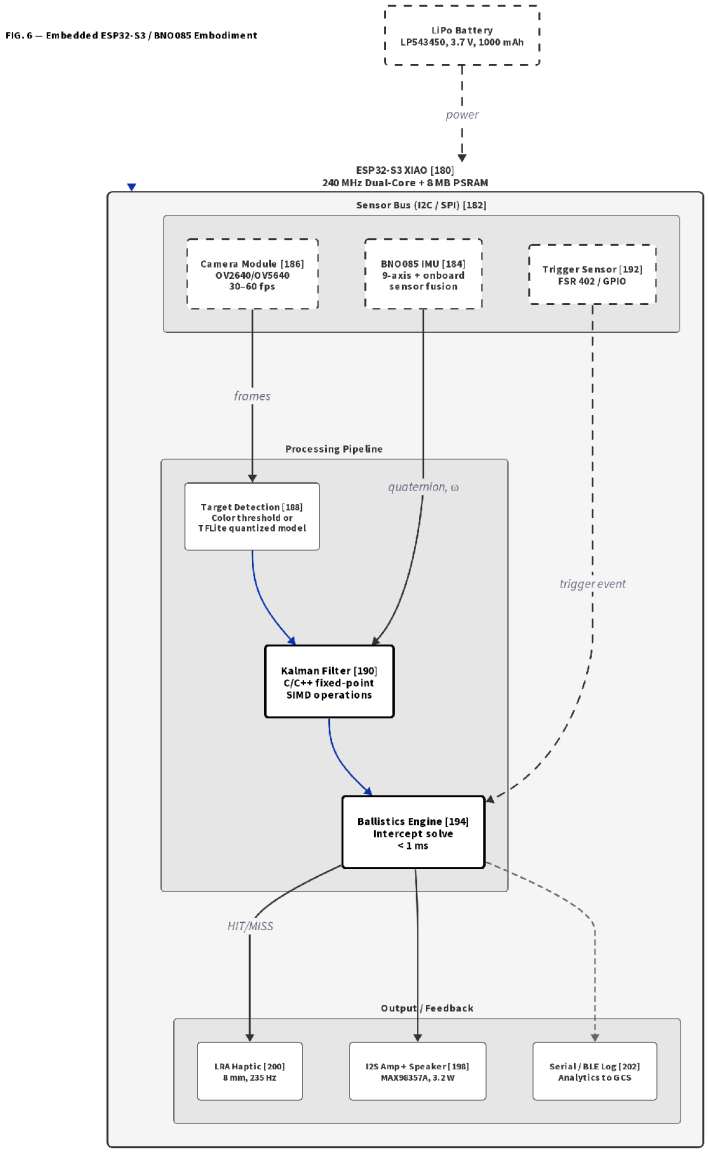

FIGURE 7 - Inertial Frame Transform

FIG. 7 is a detail view of the inertial frame transform, illustrating barrel frame coordinates, rotation matrix M(t), and the mapping to world coordinates.

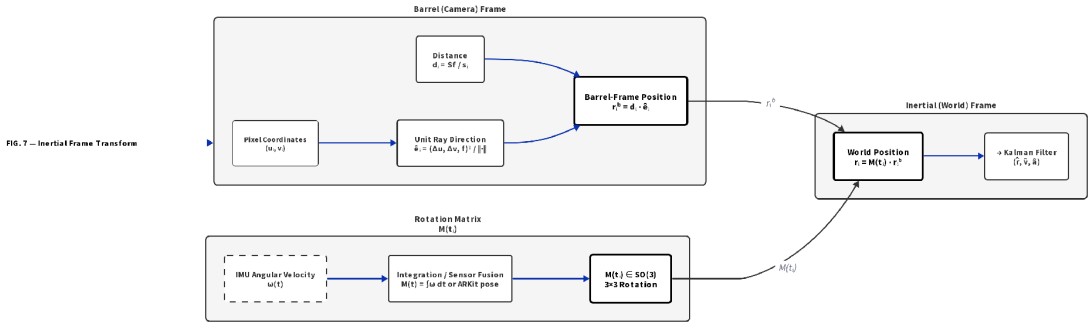

FIGURE 8 - Detailed Trigger-Time Sequence Diagram

FIG. 8 is a detailed sequence diagram of an embodiment of the system illustrating the interaction between camera capture, detection, Kalman update, trigger event, and ballistics computation with feedback timing.