AI & DEFENSE · BORDER SECURITY DRONE

Border Security Drone: Vision, Navigation & Embedded Hardware

The image-processing pipeline, state estimation (Kalman/INS) and control board of a computer-vision-based UAV for autonomous human detection.

Computer Vision | CLAHE | Canny | Kalman / INS | OpenCV | ESP-32 | PCB / Altium

Developer: Ömer Barış Berk

Scope: Image-processing pipeline · Navigation/state estimation · Embedded control board (ESP-32)

Category: AI & Defense — autonomous human detection in border zones

1. Project Summary & Concept

This project concerns an unmanned aerial vehicle (UAV) that flies autonomously over border zones and performs computer-vision-based human detection. The system has three core subsystems: an image-processing pipeline that makes the camera feed usable in low-light and noisy conditions; a navigation / state-estimation layer (Kalman/INS) that estimates the vehicle's position and velocity from noisy sensor data; and the embedded control board (ESP-32 + motor driver) that runs them. This document details the three subsystems separately, with the methods applied and the results obtained.

Three Subsystems

- **Image Processing:** low-light enhancement (CLAHE) → denoising (Median) → edge/silhouette detection (Canny)
- **Navigation:** position estimation from noisy measurements with a Kalman filter + INS navigation simulation
- **Embedded Hardware:** motor driver and power stage on a custom ESP-32-based printed circuit board (PCB)

2. System Architecture

Layer	Input → Process → Output
Sensing	Camera (RGB) → pre-processing → enhanced/edge image
Image Processing	CLAHE + Median + FFT + Canny → image ready for detection
Navigation	GPS/IMU (noisy) → Kalman/INS fusion → position, velocity, attitude estimate
Control/Hardware	ESP-32 → motor driver → drive; power: buck DC-DC

3. Image-Processing Pipeline

In border surveillance, images are often captured in twilight/backlit conditions, which makes it hard to separate a person (silhouette) from the background. The pipeline below was tested on such a real scene (a silhouette shot against sunset).

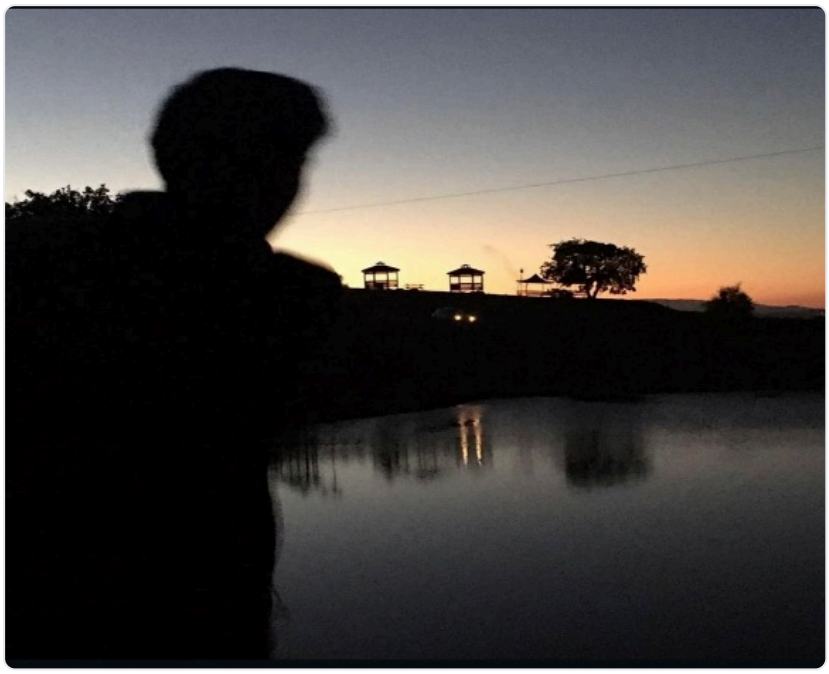

Figure 1. Input image: under-exposed against the backlight (sunset); the foreground silhouette and shore details are lost in shadow.

3.1. Problem Definition

- **Under-exposure / narrow dynamic range:** the histogram is compressed into pixel values 0–50, with detail loss in shadows.
- **Sensor noise (high ISO):** raising ISO in low light → high-frequency (Gaussian/shot) noise in sky and silhouette.
- **Blur / soft edges:** slow shutter → motion blur, edge transitions not sharp.
- **Colour artefacts:** grey/purple/green pixels in regions that should be black (low SNR — chrominance noise).

3.2. Method Applied

The image is converted to HSV space and CLAHE (Contrast-Limited Adaptive Histogram Equalization) is applied to the V (brightness) channel; noise is then suppressed with a 5×5 Median filter. Noise components are examined in the frequency domain with an FFT, and finally Canny edge detection extracts the silhouette/structure boundaries.

```
import cv2, numpy as np

img = cv2.imread("frame.jpg")
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
h, s, v = cv2.split(hsv)

# 1) Low-light enhancement — CLAHE on the V channel
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
v = clahe.apply(v)
enhanced = cv2.cvtColor(cv2.merge([h, s, v]), cv2.COLOR_HSV2BGR)

# 2) Denoising — 5x5 Median filter
denoised = cv2.medianBlur(enhanced, 5)

# 3) Structure/silhouette — Canny edge detection
edges = cv2.Canny(denoised, 50, 150)
```

CLAHE operates adaptively per tile, revealing hidden detail in dark regions; the median filter then removes granular noise without blurring the silhouette's sharp edges. Together they yield a cleaner image ready for edge detection.

3.3. Enhancement Stages & Visual Result

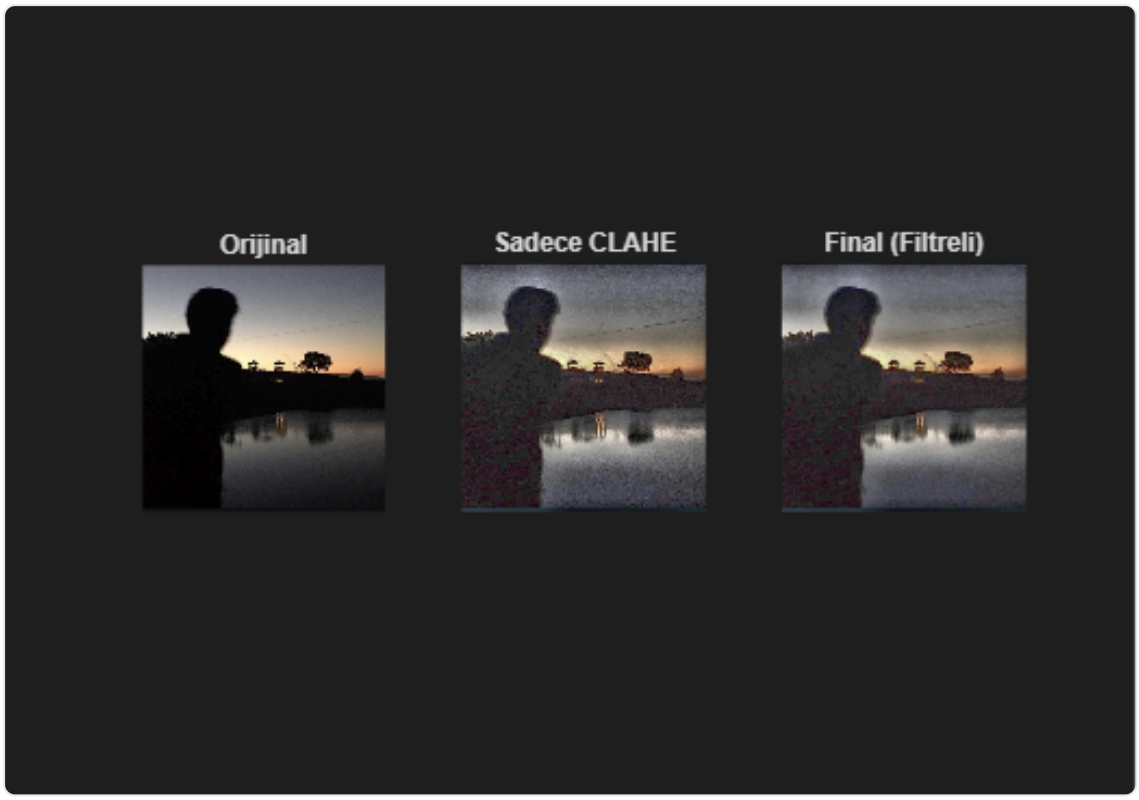

Figure 2. Enhancement stages: (left) original low exposure, (middle) after CLAHE visibility increases but noise rises, (right) after the median filter noise drops while edges are preserved.

CLAHE successfully surfaces hidden detail in the dark regions (gazebos, water reflection); as expected, boosting gain in dark areas also raises sensor noise. The subsequent 5×5 median filter cleans the granular noise without blurring the silhouette's sharp edges, giving a cleaner output.

3.4. Histogram Analysis

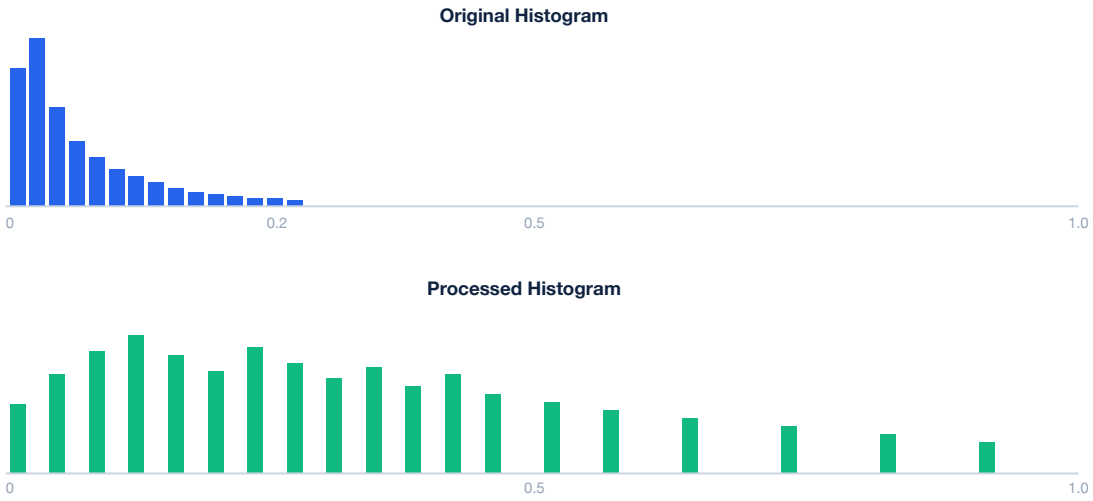

Figure 3. Histogram analysis: (top) the original V-channel pixel values are compressed into [0, 0.2]; (bottom) after processing the values are spread evenly across the full [0, 1] dynamic range.

The histogram is the mathematical proof of the visual improvement: pixel intensities are stretched from the narrow, dark [0, 0.2] range across the full normalized range [0, 1], markedly raising contrast and visibility.

3.5. Frequency Domain (FFT) & Edge Detection

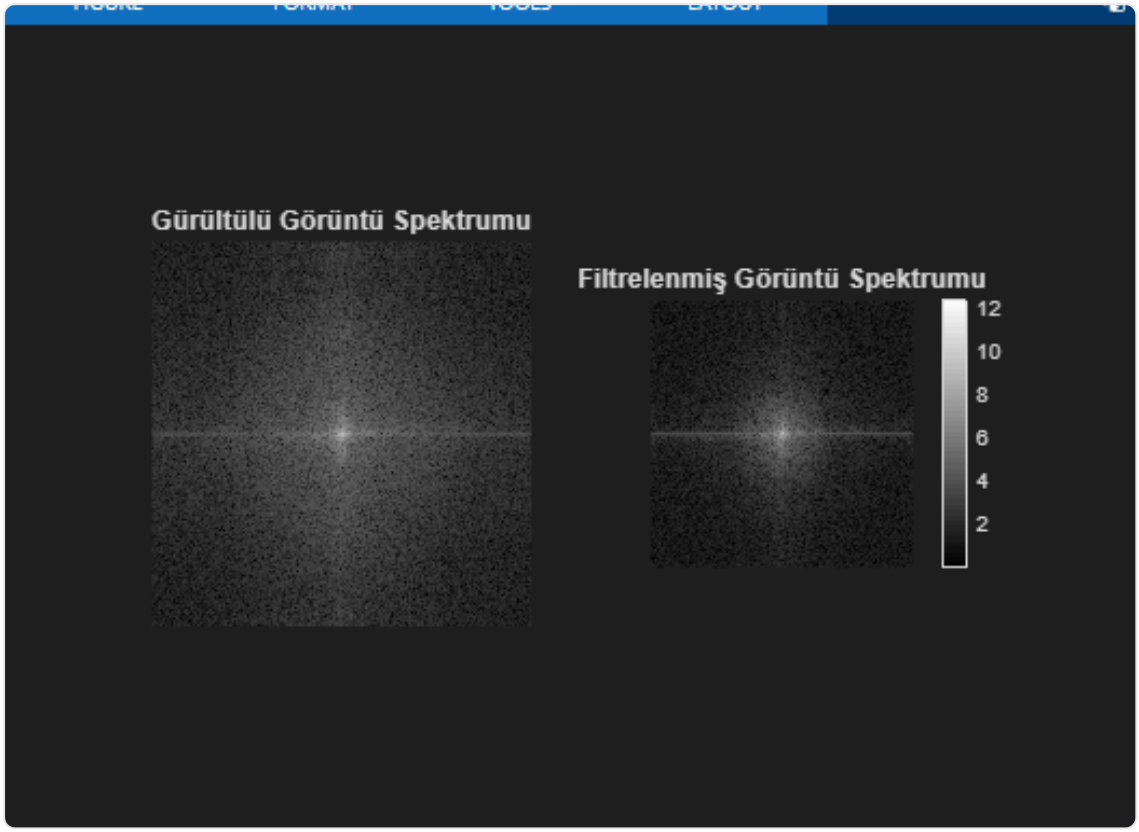

Figure 4. FFT spectrum comparison: (left) high-frequency energy is scattered in the noisy spectrum (the sensor noise CLAHE raised); (right) after the median filter the high-frequency components are reduced.

Noise is concentrated mainly in the high-frequency components. By preserving low-frequency structural components while suppressing high-frequency noise, the median filter reduces the scatter in the FFT spectrum. After denoising, Canny edge detection clearly reveals the boundary between the silhouette and the background, providing enough structural integrity for segmentation/detection.

Image Processing — Numerical Result	
Colour space	BGR → HSV (processing on the V channel)
Enhancement	CLAHE (Contrast-Limited Adaptive Histogram Equalization)
Denoising	5×5 Median filter
Frequency analysis	FFT (verification of high-frequency noise)
Edge detection	Canny (noise-robust)
PSNR (original ↔ enhanced)	16.05 dB (reflects the intense pixel change needed for low-light enhancement)

4. Navigation & State Estimation (Kalman / INS)

Autonomous flight requires reliably estimating the vehicle's position and velocity from noisy sensor measurements. This was done in two stages: first, noise reduction and state estimation were shown with a basic Kalman filter; then, position/velocity/attitude estimation performance was examined with a full INS navigation simulation.

4.1. Kalman Filter — Basic State Estimation

A one-dimensional Kalman filter applies the predict and update steps cyclically with a constant-motion model (A=1), direct measurement (H=1), process-noise covariance (Q) and measurement-noise covariance (R). The red line shows the noisy measurements; the blue line shows the filter's estimate.

```
% Kalman filter parameters
dt = 1; A = 1; H = 1;           % time step, state/measurement matrix
Q = 0.01; R = 0.1;              % process and measurement noise covariances
x = 0; P = 1;                   % initial state and error covariance
measurements = [1.1 2.0 2.9 4.2 5.1 6.3 6.9 8.1 9.2 10.3];

for k = 1:length(measurements)
    x_pred = A*x;                          % 1) Predict
    P_pred = A*P*A' + Q;
    K = P_pred*H' / (H*P_pred*H' + R);     % Kalman gain
    x = x_pred + K*(measurements(k) - H*x_pred);   % 2) Update
    P = (1 - K*H)*P_pred;
    x_estimated(k) = x;
end
```

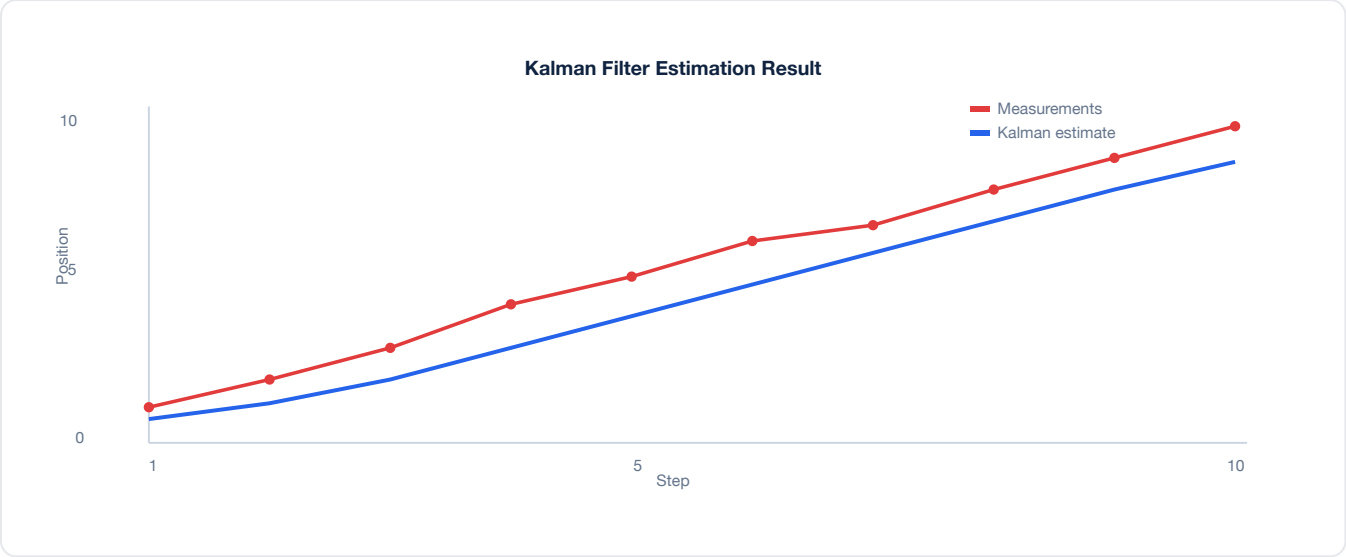

Figure 6. Kalman filter result: red = noisy measurements, blue = the filter's estimate. Despite high measurement noise, the state is tracked successfully.

4.2. INS Navigation Simulation

In a more realistic scenario, a 60-second flight profile was simulated for an inertial navigation system (INS). Position, velocity and attitude on the North/East/Down axes were tracked together with their estimation errors.

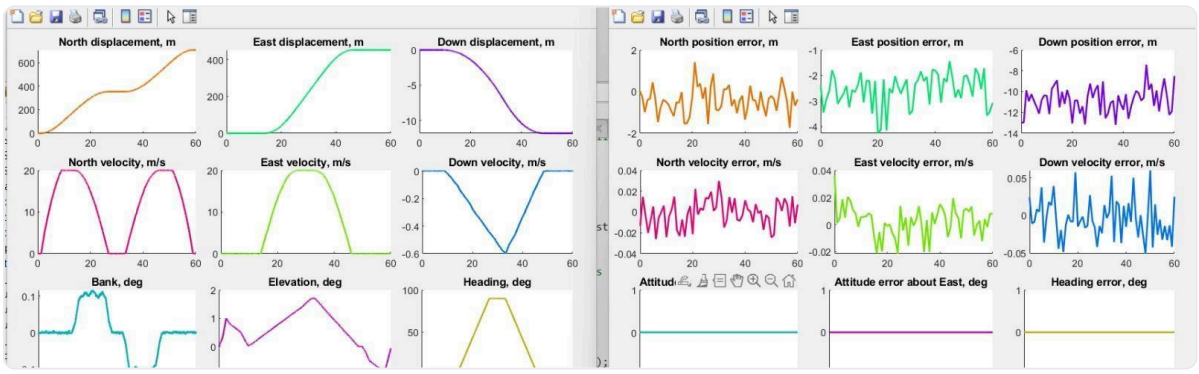

Figure 7. INS simulation. (Left) North/East/Down position, velocity and Bank/Elevation/Heading angles. (Right) position, velocity and attitude estimation errors — attitude errors near zero, position/velocity errors within a limited band.

The results show the attitude estimation error stays very close to zero while position and velocity errors remain within a narrow band, confirming that a stable position/velocity estimate is achievable in autonomous flight with GPS/IMU fusion.

5. Embedded Hardware & PCB

5.1. Control Board

The vehicle's embedded control is consolidated on our own manufactured PCB. Main components on the board:

- **ESP-32 (ESP32-WROOM-32)** dev board: dual-core main controller with Wi-Fi/Bluetooth; communication and control.
- **Motor-driver module:** drives the (stepper/BLDC) motor from the step/direction signals from the ESP-32.
- **Buck DC-DC converter:** steps the input voltage down to a stable 5 V/3.3 V for the ESP-32/logic.
- **4-pin JST connector + resistors:** sensor/peripheral connection and signal conditioning.
- **Power input:** supply line to the buck converter's input.

5.2. Designed PCB — Layer Drawings

A two-layer design was prepared for the board: component placement (silkscreen) on top and copper routing on the bottom. The design is a single-sided board with one 16-pin IC (U1), 12 resistors (R1–R12) and 8 capacitors (C1–C8).

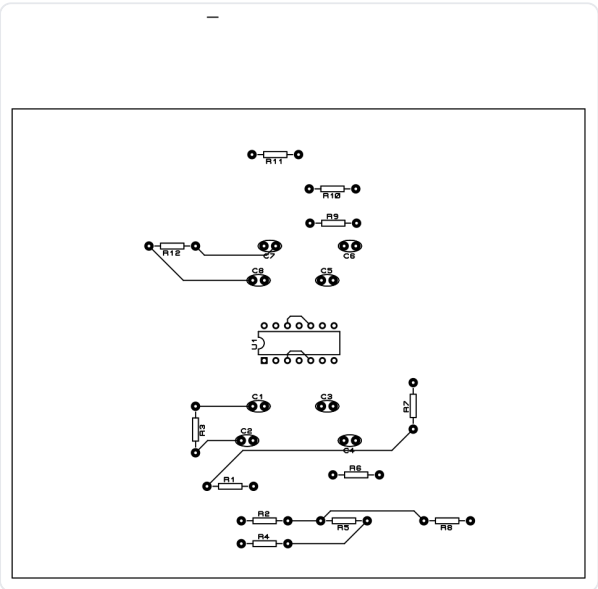

Figure 8. PCB top layer — component placement (U1 16-pin DIP, R1–R12, C1–C8).

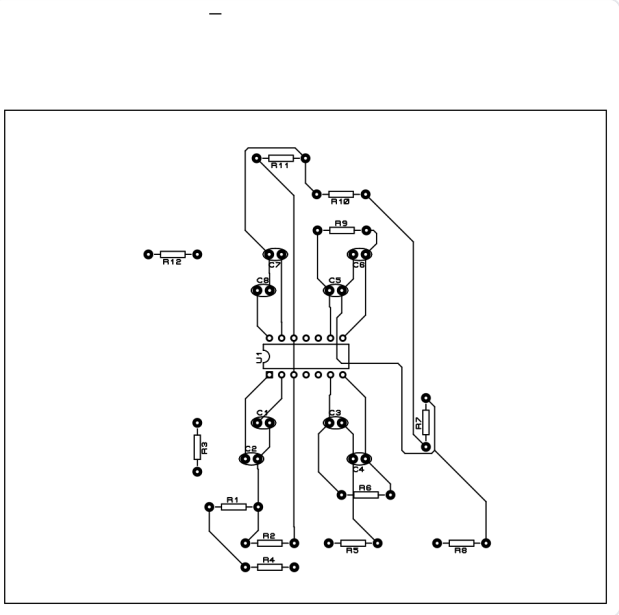

Figure 9. PCB bottom layer — copper routing. The connections between components are solved on a single copper side.

6. Conclusion

The border security drone brings three engineering subsystems together. The image-processing pipeline makes low-light/backlit images (CLAHE + Median + FFT + Canny) ready for human/silhouette detection (PSNR 16.05 dB). The navigation layer provides stable position, velocity and attitude estimates from noisy sensor data via Kalman/INS. The embedded hardware hosts the motor driver and power management on a custom ESP-32-based PCB. Together they form the basis of an autonomous, vision-based border-surveillance platform.

> *This document is compiled from the image-processing term project (CLAHE/Median/FFT/Canny), the Kalman/INS navigation study and the manufactured control board/PCB design.*