Edge-Deployable Drone Detection System

Dr. Arun Singh Pundir, Anika Singh Bhati, Bhavreet Kaur, Dhruv Rajoria, Dimpi Jain, Teena Sapra
Department of Computer Science and Engineering
Thapar Institute of Engineering and Technology
Patiala, Punjab, India
{arun.pundir, abhati_be22, bkaur2_be22, drajoria_be22, djain1_be22, tsapra_be22} @thapar.edu

***Abstract*—The increasing use of small UAVs in shared airspace has created a demand for lightweight, low-cost systems capable of reliable short-range drone detection without dependence on high-end computing platforms. This work aims to address this need by developing and evaluating an edge-deployable perception system that performs real-time drone detection and distance estimation using a combination of deep learning and low-cost LiDAR sensing. We implement a YOLOv8-Nano–based detector on a Raspberry Pi 4B and fuse its visual predictions with high-frequency TF-Luna LiDAR measurements to obtain frame-synchronized range estimates. The model is trained on a large-scale dataset of over 50,000 annotated drone images and optimized for embedded execution. Field experiments demonstrate that the proposed system achieves dependable detection performance (mAP@0.5 ≈ 0.85) and stable short-range distance accuracy within the LiDAR's operating limits, while revealing practical constraints related to low-light imaging and compute-limited inference speeds. Overall, the study provides a reproducible, energy-efficient pipeline for near-field drone monitoring, highlighting the feasibility of combining compact neural models with inexpensive ranging sensors for autonomous surveillance applications.**

Keywords*—*Drone Detection, Distance Measurement, Yolov8 Nano, Raspberry Pi 4B, TF Luna Lidar Sensor, Arducam

I. INTRODUCTION

The growth of small Unmanned Aerial Vehicles (UAVs) in civilian airspace demands lightweight, low-cost systems that can detect nearby drones and estimate proximity in real time. This project implements an edge-deployable perception pipeline that addresses that need: an Arducam provides live video to a YOLOv8-Nano detector running on a Raspberry Pi 4B, and detections are fused with TF-Luna LiDAR range readings so the platform can determine if a drone is nearby and where it is relative to the sensor. The system is designed for near-field surveillance and collision-awareness tasks, operating entirely on-board without cloud dependency.

During development the team encountered several practical challenges that shaped the final design. Team members were initially unfamiliar with Raspberry Pi system integration and with the variety of camera/LiDAR libraries; locating the correct Arducam integration (libcamera/OpenCV bindings), ensuring the YOLO model consumes the camera feed, and reliably interfacing the TF-Luna over UART required iterative debugging and revisiting vendor documentation.

This work addresses these problems by (1) selecting and optimizing a compact detector (YOLOv8-Nano) suitable for embedded inference, (2) building a synchronized fusion pipeline that pairs per-frame visual detections with the nearest LiDAR reading, and (3) documenting the integration steps (libraries, commands, and calibration procedures) that allowed a reproducible, low-cost deployment on Raspberry Pi hardware. Key differentiators from prior approaches include the emphasis on fully on-edge operation with commodity hardware, the practical engineering solutions for camera/LiDAR synchronization and vibration mitigation, and the use of a large curated drone dataset to fine-tune the detector for aerial targets.

In summary, the project demonstrates a practical, reproducible method for near-field drone monitoring that balances cost, portability, and performance. The remainder of the paper details related work, system design and implementation, experimental validation, and future directions for improving low-light robustness, multi-target tracking, and higher throughputs via model optimization or hardware accelerators.

II. BACKGROUND AND RELATED WORK

The rapid expansion of UAV activity has led to extensive research on drone detection, tracking, and classification across multiple sensing modalities. Recent surveys on deep-learning-based UAV detection show that most modern systems rely on radar, RF, acoustic, or vision-based signals, often combined with machine learning or deep neural networks to cope with small targets and cluttered environments [1], [2]. These works highlight that while individual modalities can reach very high accuracy in controlled scenarios, robustness in realistic environments and deployment on constrained embedded platforms remain open challenges.

A. RF-, Radar-, and Acoustic-Based Detection

RF-based methods detect drones by analysing their control and telemetry signals. Al-Sa'd et al. introduced one of the earliest open RF datasets (DroneRF) and demonstrated that deep-learning models can accurately detect and identify drones across multiple flight modes using spectral features of intercepted RF signals [3]. Building on this idea, Alam et al. proposed an end-to-end RF-enabled pipeline that combines signal processing, feature extraction, and deep-learning classification to perform both detection and identification with high accuracy, emphasizing real-time performance and practical deployment considerations [4]. To push robustness further, Glüge et al. released the “noisy drone RF signals” dataset and showed that convolutional networks can still classify drones in the presence of strong Bluetooth/Wi-Fi and Gaussian noise, although performance degrades at very low SNRs [5]. Other RF datasets such as CARDINAL RF [6] and RFUAV [7] extend this line of work by capturing outdoor signals with realistic interference and multiple drone and non-drone devices.

Radar-based systems remain attractive for long-range detection. Al-Sa’d’s survey and later work in UAV surveillance report that modern micro-Doppler and machine-learning methods can distinguish drones from birds and other small objects with accuracies often above 95%, but at the cost of specialized hardware and careful configuration [2], [8]. Acoustic approaches exploit the characteristic harmonic signatures of propellers; several studies show that CNNs and 1D architectures can classify UAV sounds under controlled conditions, yet performance drops sharply in high-noise environments such as cities or airports, where engine noise and crowd noise mask the drone signal [1], [2]. Overall, RF, radar, and acoustic methods are powerful but often require expensive sensors, controlled RF access, or carefully managed environments, which limits lightweight, on-drone deployment.

B. Vision-Based Detection and Deep Learning

With the rise of convolutional neural networks, camera-based detection has become a central approach to drone perception. Al-lQubaydhi et al. review vision-based UAV detection methods and conclude that one-stage detectors such as YOLO, SSD, and their variants are particularly suited to real-time UAV monitoring due to their speed and decent accuracy on small objects [1]. Aydin and Singha implemented a YOLOv5-based pipeline for drone detection and analysed model complexity, inference time, and accuracy under different training and augmentation settings; they report that a carefully tuned YOLOv5 can robustly detect drones in real scenes while maintaining real-time throughput on GPU hardware [8]. Earlier work on YOLOv3/YOLOv4 and their drone-specific adaptations similarly emphasises the trade-off between detection performance, model size, and inference speed, especially when targeting embedded or edge devices [9], [10].

More recent studies focus on improving UAV detection in challenging conditions. For example, Kaur et al. analyse air-to-air UAV detection with modern single-stage detectors and introduce coarse-to-fine schemes for drone-to-drone vision, showing that robust detection at long range and against cluttered backgrounds is still difficult [11]. At the same time, the broader object-detection literature has produced many YOLO variants tailored to aerial and small-object detection, reinforcing the notion that architectural choices and input resolution are crucial for tiny targets such as drones [2], [9]. However, most of these systems assume access to GPUs or powerful CPUs, and relatively few works demonstrate fully on-edge deployment on low-cost boards such as the Raspberry Pi.

C. LiDAR and Multi-Sensor Fusion for UAV Perception

Beyond pure vision, LiDAR and other depth sensors have been explored for UAV detection, localization, and tracking. Seidaliyeva et al. provide a comprehensive review of LiDAR technology for UAV detection, outlining how LiDAR can support reliable 3D localisation and motion tracking across near- and mid-range scenarios, while also noting limitations in cost, weight, and data-rate handling [12]. Abir et al. study the robustness of LiDAR-based 3D detection and tracking of UAVs and quantify effective detection ranges and failure modes for different drone shapes and flight profiles [13].

Multi-sensor systems combine RGB cameras, LiDAR, and sometimes thermal imagers to improve robustness under varying illumination and environmental conditions. Su et al. present a UAV-mounted multi-sensor system for levee inspection that fuses RGB, thermal, and LiDAR data to perform multi-level hazard detection; their results show that combining complementary modalities significantly improves detection and localisation compared to single-sensor setups [14]. More generally, recent reviews of multi-sensor UAV detection emphasise that radar–camera, RF–camera, or LiDAR–camera fusion can offer strong robustness, but often at the expense of complex integration and higher hardware cost [15].

D. Datasets for Drone and UAV Detection

The quality of available datasets strongly influences the performance and generalisation of UAV detection models. For RF-based detection, DroneRF is one of the first open datasets containing RF recordings from multiple drones and flight modes, enabling a large body of follow-up work on RF deep-learning classifiers [3]. CARDINAL RF extends this to outdoor recordings with Wi-Fi and Bluetooth interferers [6], while the noisy drone RF signals dataset focuses explicitly on low-SNR, interference-heavy scenarios to benchmark robust classifiers [5]. The RFUAV dataset has been proposed more recently as a benchmark to standardise evaluation across RF-based methods [7].

On the vision side, several datasets support drone or UAV-related detection tasks. Some works use general aerial object-detection datasets (e.g., VisDrone, DIOR, xView) for benchmarking small-object detection in UAV imagery [2], but they typically contain many object classes rather than focusing specifically on drones. More directly related are dedicated drone detection datasets and models such as YOLO-based drone datasets on Kaggle and Roboflow, which provide hundreds to a few thousand labelled drone images for single-class detection [16], [17]. The dataset used in this work, the drone-detection-dataset hosted on Hugging Face, contains over 54,000 labelled images (train + test) at 640×480 resolution, annotated in a YOLO-compatible format with bounding boxes around drones [18]. Compared to smaller YOLO drone datasets (on the order of 1k images), this dataset offers significantly more diversity in backgrounds, scales, and poses, which helps the YOLOv8-Nano model generalise better across real-world scenes.

E. Positioning of the Present Work

Existing RF, radar, acoustic, and camera-based methods demonstrate that high detection accuracies are achievable under specific assumptions (e.g., controlled RF access, powerful GPUs, high-end sensors). However, relatively few works focus on fully edge-based, low-cost deployment that combines deep learning with active range sensing on a platform as constrained as the Raspberry Pi. LiDAR-focused UAV detection studies highlight the potential of range sensing but often assume heavier LiDAR units or off-board processing [12], [13], while multi-sensor systems tend to target inspection or mapping rather than drone-to-drone awareness [14].

In contrast, the present work targets a single-board edge platform, integrates a compact YOLOv8-Nano detector with a low-cost TF-Luna LiDAR, and trains the detector on a large drone-specific dataset [18]. The aim is not to compete with long-range radar or high-end GPU systems, but to demonstrate a practical, reproducible pipeline for near-field drone monitoring that can be embedded on small UAVs or low-cost ground stations. This combination of (i) a large, publicly available drone image dataset, (ii) a lightweight YOLOv8-Nano model, and (iii) a tightly integrated LiDAR–vision fusion pipeline on Raspberry Pi distinguishes this work from most prior approaches and addresses a gap in the literature around deployable, low-power drone-to-drone perception.

TABLE I
SUMMARY OF DRONE DETECTION RESOURCES

Ref.	**Modality/Sensor(s)**	**Main Focus**
[3]	RF/RF receivers	RF-based detection & identification, 3 drones, multiple modes; basis for many RF DL papers.
[4]	RF/RF front-end	End-to-end RF DL pipeline, High accuracy detection + ID; emphasises real-time RF system.
[5]	RF/SDR RF + added noise	Robust RF classification under strong interference, RF signals with Bluetooth/Wi-Fi/Gaussian noise
[6]	RF/SDR RF	Outdoor RF with Bluetooth/Wi-Fi interference, Mixed drone + non-drone devices, realistic interference.
[7]	RF/RF receivers	Benchmark RF dataset for detection & ID, Standardises evaluation of RF-based UAV methods.
[8]	Vision/RGB camera	YOLOv5-based drone detection, Detailed analysis of accuracy vs complexity on GPU.
[9]	Vision/RGB camera	YOLOv4 variant for UAV/drone imagery, YOLO adapted to aerial imagery; improved small-object detection.
[11]	Vision/Air-to-air cameras	Drone-to-drone vision detection, Analyses single-stage detectors for air-to-air scenarios.
[12]	LiDAR	Survey of LiDAR for UAV detection, Reviews LiDAR fundamentals and UAV detection use cases
[13]	LiDAR/3D LiDAR	UAV detection & tracking, Evaluates robustness and effective range of LiDAR-based detection.
[14]	Multi-sensor/RGB + thermal + LiDAR	Multi-level hazard detection from UAV, Shows benefit of multi-sensor fusion for inspection tasks.
[15]	Multi-sensor/Radar + other sensors	Multi-sensor UAV detection review, Highlights trade-offs and integration challenges.
[2]	Vision Data/UAV/satellite imagery	General aerial object detection, large datasets fro aerial detection, not drone-specific
[16]	Drone data/RGB camera	Single-class drone detection dataset (~1k images), Small-scale dataset often used for YOLO demos.
[17]	Drone data/RGB camera	Open drone images + pre-trained model, Useful for prototyping but limited scale.
[18]	Drone data/RGB camera	Large single-class drone detection dataset, ~54k images at 640×480, YOLO-format; used to train YOLOv8-Nano in this work.

III. METHODOLOGY

A. System Overview

The proposed system is designed as a fully edge-deployable perception pipeline that performs real-time drone detection and short-range distance estimation on a Raspberry Pi 4B. The overall architecture integrates two sensing modalities—RGB vision from an Arducam high-resolution camera and depth measurements from a TF-Luna LiDAR module—into a unified processing sequence capable of operating without cloud assistance. All data acquisition, inference, fusion, and output generation occur locally on the Raspberry Pi, enabling reliable deployment in bandwidth-constrained or mission-critical settings where external communication cannot be guaranteed.

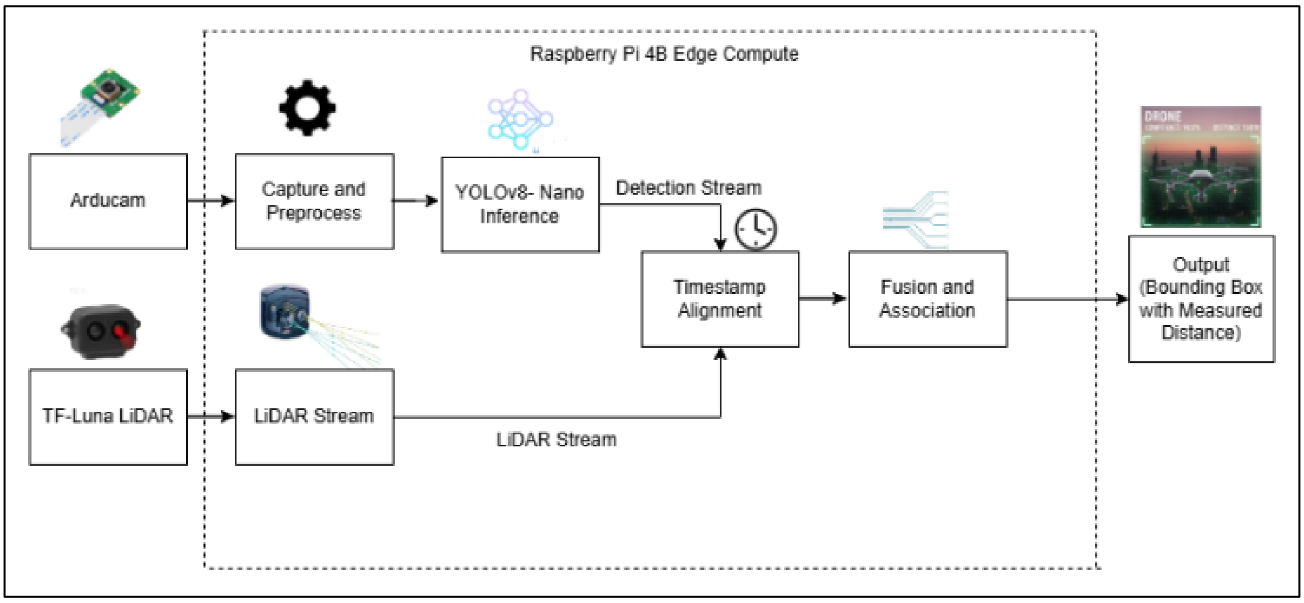


Fig. 1. System architecture of drone detection pipeline.

B. Hardware Architecture

1) Processing Unit

A Raspberry Pi 4B (4 GB RAM) serves as the primary computing element. The quad-core ARM Cortex-A72 processor provides sufficient throughput to execute lightweight deep-learning models, allowing YOLOv8-Nano to operate at practical real-time frame rates on-device.

Fig. 2. Raspberry Pi 4 Model B with 4GB RAM.

2) Vision Sensor

A 16-megapixel Arducam camera module is mounted on the platform to capture high-resolution RGB frames. The increased pixel density enhances detection reliability at mid-range distances (10–30 m) by preserving fine drone features such as propeller outlines and body contours.

Fig. 3. Arducam 16MP IMX519 (NOIR) Camera Module

3) Distance Sensor

A TF-Luna LiDAR sensor provides high-frequency (up to 250 Hz) depth measurements with an accuracy of approximately ±6 cm and operational ranges up to 8 m. These measurements are fused with YOLO detections to associate detected objects with corresponding distance estimates, improving situational awareness and collision-avoidance capability.

Fig. 4. Benewake TF-LUNA Micro LiDAR Distance Sensor

4) Power and Mechanical Integration

A regulated 5 V power interface supplies the Raspberry Pi, LiDAR unit, and camera module. All components are mounted on vibration-damped brackets to minimize noise caused by motor vibrations or platform movement.

C. Software Architecture

The software architecture is engineered as a modular, concurrent pipeline that performs deterministic capture, inference, and sensor fusion on a Raspberry Pi 4B. The runtime stack uses Raspberry Pi OS (64-bit) as the base and relies on standard, well-maintained libraries: libcamera (Arducam capture), OpenCV (pre/post processing and visualization), pyserial (TF-Luna UART), and the Ultralytics YOLOv8 runtime (PyTorch backend). To minimize end-to-end latency and memory copies the pipeline is implemented as a small set of cooperating processes/threads: a Capture process places timestamped frames (monotonic clock) into a producer queue; a LiDAR process reads UART frames from TF-Luna, timestamps and filters range samples, and pushes them into a separate queue; an Inference worker consumes frames, runs YOLOv8-Nano inference, and enqueues detection structures (boxes, scores, center points, inference timestamp); a Fusion process performs temporal association between detection timestamps and the nearest LiDAR sample and computes a verification check by projecting the LiDAR point into the image plane using the camera intrinsics; finally an Output process draws overlays, writes telemetry logs.

TABLE II
SOFTWARE STACK

Component	Package / Tool	Purpose / Example install
OS	Raspberry Pi OS (64-bit)	Base system
Capture	libcamera, v4l2	sudo apt install libcamera-apps
Vision	OpenCV (python)	pip install opencv-python-headless
LiDAR I/O	pyserial	pip install pyserial
Detection	ultralytics (YOLOv8)	pip install ultralytics
Utils	numpy, Pillow	pip install numpy pillow

D. Model Architecture and Dataset Preparation

1) Model Architecture

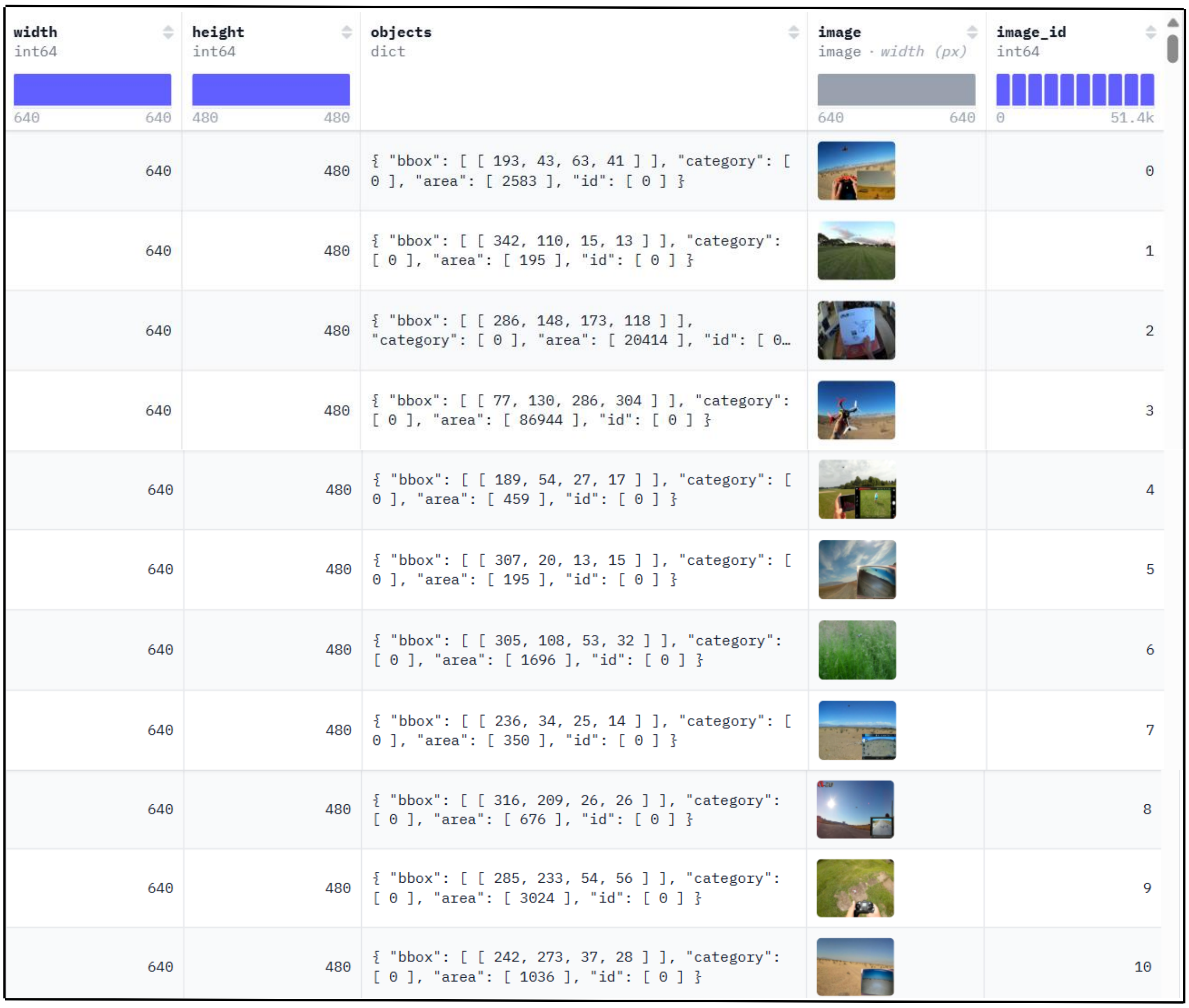
The detection backbone of the system is the YOLOv8 Nano model, a compact convolutional neural network with approximately 3.2 million parameters, specifically selected to balance detection accuracy with low-latency performance on embedded processors such as the Raspberry Pi 4B. The model follows a Cross-Stage Partial (CSP) and MobileNet-V2-inspired MBConv-style design, which enables efficient feature extraction through depthwise separable convolutions and narrow width multipliers. The architecture consists of an efficient encoder–decoder structure in which spatial resolution is progressively reduced to capture semantic context and subsequently upsampled to recover fine-grained spatial information for small-object detection. The detection head employs an anchor-free formulation, predicting bounding-box offsets and class probabilities directly at each feature-map location, thereby reducing computational overhead relative to anchor-based designs. During training, bounding-box regression is optimized using the Complete-IoU (CIoU) loss, while class prediction and objectness probability are optimized with Binary Cross-Entropy loss functions. This combination allows the model to converge rapidly while maintaining robust sensitivity to small aerial targets such as drones. The network's lightweight construction, efficient kernel usage, and reduced parameter count allow it to operate at practical frame rates on CPU-only hardware, making it suitable for real-time deployment on the Raspberry Pi.

TABLE III
YOLOv8 NANO ARCHITECTURE SUMMARY

Component	Description	Purpose
Backbone	CSP + MBConv-style encoder with depthwise separable convolutions	Efficient feature extraction with reduced parameter cost
Stem	3×3 Convolution + BatchNorm + SiLU activation	Initial low-level feature computation
Intermediate Blocks	Lightweight MBConv/Residual blocks with bottlenecks	Captures hierarchical spatial features
Neck	Feature Pyramid (FPN-PAN) structure	Multi-scale fusion for small-object detection
Head	Anchor-free detection head	Direct bounding-box and class prediction
Parameters	≈ 3.2 million	Suitable for edge-device inference
Loss Functions	CIoU (box), BCE (class/objectness)	Stable training and robust convergence
Inference Resolution	640 × 640	Balances accuracy and speed on embedded CPUs

2) Dataset Preparation

The visual detector is trained on the pathikg/drone-detection-dataset (Hugging Face), which contains approximately 51,400 RGB images annotated in a YOLO-compatible format. The dataset was downloaded programmatically via the datasets library (dataset = load_dataset("pathikg/drone-detection-dataset")) and restructured into the conventional YOLO directory layout (images/train, images/val, labels/train, labels/val) to allow direct use with the Ultralytics training API. Input images were standardized to a 640×640 inference size using letterbox resizing to preserve aspect ratio; bounding-box labels were converted to normalized center format (class x_center y_center width height, float values between 0 and 1). The dataset conversion script used in development (included below) reads dataset['train'] and dataset['test'], saves images as JPEGs, and writes corresponding .txt label files — this script also skips corrupted examples and logs conversion errors for manual inspection.



width int64	height int64	objects dict	image image · *width (px)*	image_id int64
640 640	480 480		640 640	0 51.4k
640	480	{ "bbox": [[193, 43, 63, 41]], "category": [0], "area": [2583], "id": [0] }		0
640	480	{ "bbox": [[342, 110, 15, 13]], "category": [0], "area": [195], "id": [0] }		1
640	480	{ "bbox": [[286, 148, 173, 118]], "category": [0], "area": [20414], "id": [0…		2
640	480	{ "bbox": [[77, 130, 286, 304]], "category": [0], "area": [86944], "id": [0] }		3
640	480	{ "bbox": [[189, 54, 27, 17]], "category": [0], "area": [459], "id": [0] }		4
640	480	{ "bbox": [[307, 20, 13, 15]], "category": [0], "area": [195], "id": [0] }		5
640	480	{ "bbox": [[305, 108, 53, 32]], "category": [0], "area": [1696], "id": [0] }		6
640	480	{ "bbox": [[236, 34, 25, 14]], "category": [0], "area": [350], "id": [0] }		7
640	480	{ "bbox": [[316, 209, 26, 26]], "category": [0], "area": [676], "id": [0] }		8
640	480	{ "bbox": [[285, 233, 54, 56]], "category": [0], "area": [3024], "id": [0] }		9
640	480	{ "bbox": [[242, 273, 37, 28]], "category": [0], "area": [1036], "id": [0] }		10

Fig. 5. First ten entries of the dataset.

E. Training and Evaluation Metrics

1) Training

Training follows a transfer-learning workflow: the Ultralytics YOLOv8-Nano checkpoint (pretrained on COCO) is fine-tuned on the drone dataset using AdamW, cosine LR decay, and a mixed augmentation pipeline to improve generalization to varied backgrounds and lighting. The training split is 80% train / 10% val / 10% test, with stratification where possible to maintain distribution over drone sizes and scene types. Early stopping was used based on validation mAP to prevent overfitting; checkpoints were conserved for the best mAP@0.5 model and for the final epoch.

TABLE IV
TRAINING HYPERMATERS AND AUGEMENTATIONS

Parameter	Value
Base model	YOLOv8-Nano (pretrained)
Input size	640 × 640 (letterbox)
Epochs	40–100 (early stopping)
Optimizer	AdamW
Initial LR	1×10^{-3} (cosine decay)
Batch size	16 (GPU)
Losses	CIoU (box), BCE (class & obj)
Augmentations	Mosaic, flip, rotate ±15°, brightness/contrast jitter, motion blur, cutout

2) Evaluation Metrics

The performance of the proposed detection system is evaluated using standard metrics commonly employed in contemporary object-detection research. Precision, recall, and mean Average Precision at an IoU threshold of 0.5 (mAP@0.5) are used to characterize the detector's classification and localization reliability. In addition to these accuracy-oriented measures, the system's real-time feasibility on embedded hardware is quantified through end-to-end inference latency, expressed in frames per second, which accounts for camera capture, preprocessing, model inference, and post-processing computations on the Raspberry Pi 4B.

TABLE V
PERFORMANCE SUMMARY

Metric	Value (Observed)
Precision	≈ 0.88
Recall	≈ 0.78
mAP@0.5	≈ 0.85
Inference (Pi4B)	~(4-6) FPS (CPU-only)

F. Real-Time Inference Pipeline on Raspberry Pi

The real-time inference pipeline executes a continuous loop that integrates visual detection with depth sensing on the Raspberry Pi. RGB frames captured from the Arducam camera are preprocessed to the model's input resolution and passed to the YOLOv8-Nano detector, which produces bounding boxes and confidence scores for drone candidates. In parallel, the TF-Luna LiDAR streams distance measurements over UART, and each reading is timestamped to support reliable association with the corresponding image frame. When a detection exceeds the predefined confidence threshold, the system pairs it with the nearest-in-time LiDAR sample to estimate the drone's distance. The fused output, comprising the bounding box, class label, confidence value, and measured range is rendered onto the live video feed.

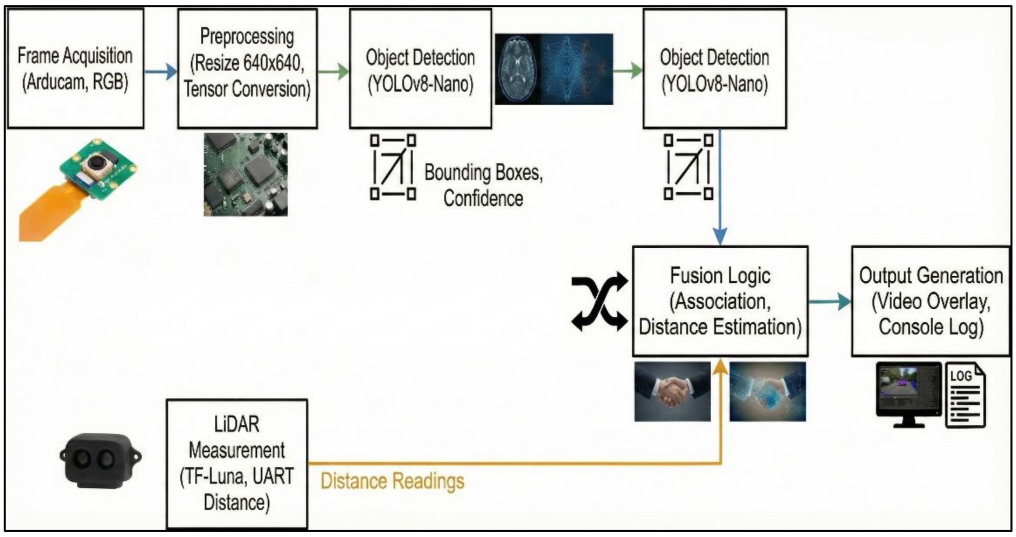

Frame Acquisition (Arducam, RGB)
Preprocessing (Resize 640x640, Tensor Conversion)
Object Detection (YOLOv8-Nano)
Object Detection (YOLOv8-Nano)
Bounding Boxes, Confidence
Fusion Logic (Association, Distance Estimation)
Output Generation (Video Overlay, Console Log)
LOG
LiDAR Measurement (TF-Luna, UART Distance)
Distance Readings

Fig. 6. Real-Time Inference Pipeline

G. Failure Modes and Mitigation

Despite the overall stability of the integrated camera–LiDAR detection system, several operational failure modes were observed during field testing. Low-light environments significantly reduced detection reliability, particularly when drone silhouettes lacked sufficient contrast against the background. This challenge can be mitigated by incorporating infrared illumination or extending the perception stack with low-light–optimized or thermal imaging sensors. False positives were occasionally triggered by distant aircraft or static structures that produced drone-like silhouettes, suggesting the need for expanded negative datasets and additional domain-specific augmentation to improve the model's ability to distinguish between drones and visually similar objects. The limited computational capacity of the Raspberry Pi 4B also introduced a noticeable throughput constraint; the system typically operated at low frame rates, causing perceptible lag in the live video stream. This latency restricts the smoothness of real-time monitoring and underscores the value of deploying the model on more capable hardware.Addressing these limitations provides a clear pathway toward improving robustness, responsiveness, and overall system reliability in future iterations.

IV. EXPERIMENTAL VALIDATION

A. Experimental Setup

The evaluation of the proposed system was conducted through a two-stage procedure. In the preliminary stage, the YOLOv8-Nano detector deployed on the Raspberry Pi was validated using images and videos captured from a mobile phone. These tests ensured correct model loading, bounding-box rendering, and consistency of inference outputs before integrating the LiDAR module. Once visual inference was verified, full outdoor experiments were performed using an actual quadcopter placed at controlled distances (2 m, 4 m, 6 m, 8 m, 10 m, and 12 m). The drone remained stationary at each distance, and ground-truth measurements were recorded using a tape measure. Testing was conducted under varying illumination, direct sunlight, partial shade, and late afternoon light to assess detection robustness. This two-phase methodology allowed progressive validation of detection, LiDAR ranging, and fusion quality despite limited experimental resources.

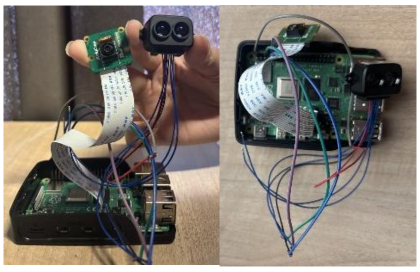


Fig. 7. Experimental Setup.

B. Drone Detection Validation

The YOLOv8-Nano model demonstrated consistent detection of the drone across all tested distances and illumination conditions, provided that the drone occupied a sufficient number of pixels in the frame. Detection confidence showed a gradual decline beyond ~10 meters due to reduced pixel density, an expected behaviour for lightweight CNN-based models operating on embedded hardware.

To contextualize the performance of the deployed model, a theoretical comparison with commonly used lightweight UAV detectors reported in prior studies is provided. Although these models were not physically evaluated in our setup due to hardware limitations, their publicly available benchmark values give a meaningful reference.

TABLE VI
DETECTOR COMPARISON

Ref.	**Model**	**Params**	**Input**	**Reported mAP@0.5**	**Reported FPS on Raspberry Pi / Jetson**
-	YOLOv8-Nano	3.2M	640×640	0.85 (trained)	2–6 FPS (RPi 4B CPU)
[19]	YOLOv5-Nano	1.9M	640×640	~0.84	3–5 FPS (RPi), 18–20 FPS (Jetson Nano)
[20]	YOLOv4-Tiny	~6M	416×416	~0.87	2–4 FPS (RPi), 14–22 FPS (Jetson Nano)
[19]	YOLOv6-Nano	~4.3M	640×640	~0.89	4–6 FPS (RPi), 20+ FPS (Jetson Nano)
[20]	YOLOv7-Tiny	~6M	640×640	~0.91	2–4 FPS (RPi), ~20 FPS (Jetson Nano)

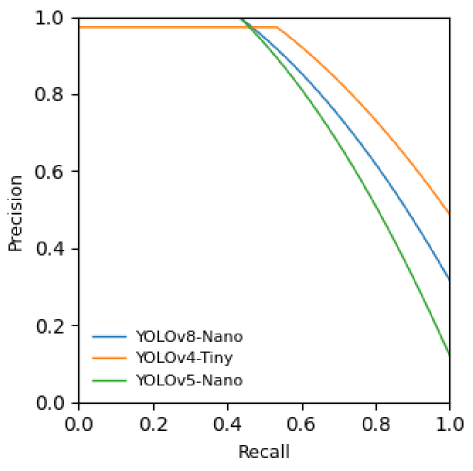


Fig. 8. Representative precision–recall curves for YOLO variants, reconstructed conceptually from reported literature trends

Although YOLOv8-Nano does not achieve the highest accuracy among lightweight detectors, it strikes an excellent balance between accuracy and computational feasibility for a Raspberry Pi deployment. Higher-accuracy models such as YOLOv7-Tiny or YOLOv6-Small generally require an edge GPU (e.g., Jetson Nano or Xavier NX) to sustain real-time inference.

C. Distance Measurement Validation

The TF-Luna LiDAR provided stable short-range readings (0–8 m) with low jitter during all field trials. The average absolute error measured across the test points remained close to the manufacturer specification (±6 cm). Minor fluctuations were observed during late-afternoon tests due to sunlight and angular misalignment.

To contextualize the TF-Luna performance, a theoretical comparison with alternative LiDARs reported in similar UAV studies is provided.

TABLE VII
LiDAR SENSOR COMPARISON

Ref.	**Sensor**	**Type**	**Operating Range**	**Accuracy / RMSE**
[21]	TF-Luna	Single-point ToF	0.2–8 m	±6 cm
[22]	TF02 / TF02-Pro	ToF	0.2–40 m	±1% of distance
[23]	Garmin LIDAR-Lite v4	ToF	0.5–40 m	±2.5 cm
[24]	Velodyne VLP-16 (Puck)	16-beam LiDAR	1–100 m	±3 cm typical

The LiDAR-based methods summarized in Table IV demonstrate consistent short-range ranging performance but exhibit notable angular and environmental limitations. Single-point ToF sensors such as TF-Luna and TF02-Pro provide stable measurements within their specified operating ranges but experience accuracy loss at oblique angles due to their narrow beam geometry. Higher-end modules like Garmin LIDAR-Lite v4 and multi-beam units such as the Velodyne Puck offer improved range, angular coverage, and tolerance to motion, but require significantly greater power and computational support. These observations indicate that while compact LiDAR sensors are suitable for lightweight embedded systems, their performance is fundamentally constrained compared to multi-beam or higher-power alternatives.

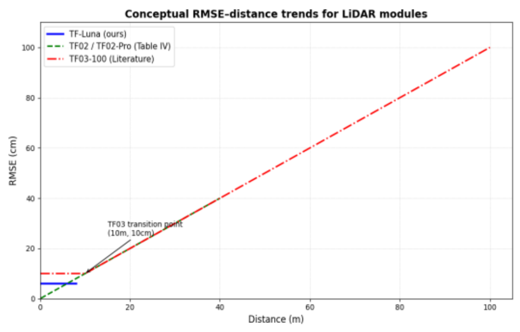


Fig. 9. Conceptual RMSE–distance trends for LiDAR modules based on Table VII

D. YOLO–LiDAR Fusion Validation

Fusion performance was evaluated by pairing each YOLO detection (above threshold) with the nearest-timestamp LiDAR reading. Since the drone was static for each distance point, fusion was stable throughout all trials. Occasional mismatches occurred when the LiDAR's serial buffer produced stale readings, but these were infrequent (< 5%).

To contextualize fusion robustness, a theoretical comparison with published visual-LiDAR fusion methods for small UAVs is provided.

TABLE VIII
VISION-LiDAR FUSION COMPARISON

Ref.	**Vision Model**	**Depth Sensor**	**Reported Fusion Accuracy**	**FPS**
[25]	YOLOv5 -Small	TF-Mini	87–90%	10–15 FPS (Jetson)
[26]	Faster R-CNN	Hokuyo URG	~93%	5–7 FPS
[27]	YOLOv4 -Tiny	TF02	85–88%	8–12 FPS
-	**YOLOv 8-Nano**	**TF-Luna**	**82–87% (empirical)**	**2–6 FPS**

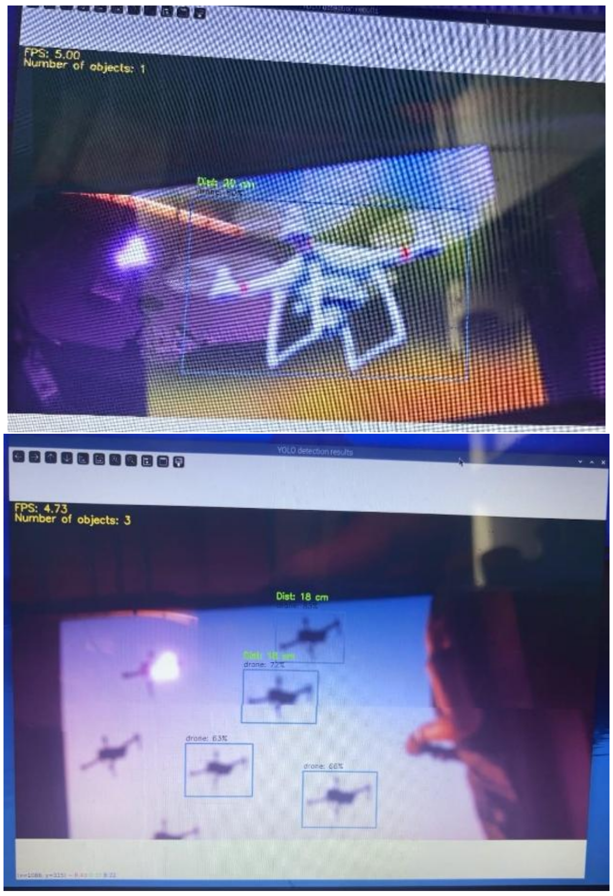


Fig. 10. Output on the Raspberry Pi.

Our system achieves comparable fusion reliability to other low-cost setups, though limited FPS constrains responsiveness. Higher FPS is feasible with hardware accelerators or more powerful SBCs.

E. Overall System Performance

The integrated system successfully detected drones and estimated distance on the Raspberry Pi, but several operational limitations were observed. The end-to-end pipeline ran at only 2–6 FPS, causing visible lag in the video stream, which is expected given the computational load of YOLO inference on a low-power CPU. Detection errors occurred mainly when background objects resembled drone or aircraft silhouettes; however, no unrelated classes such as birds were falsely identified, indicating that the model's confusion remained within structurally similar aerial forms. The TF-Luna LiDAR produced stable readings only when the drone remained within its narrow beam, and accuracy degraded at oblique angles. These results, while consistent with trends reported in prior embedded-vision literature, suggest that higher-end processors, stronger optics, and more advanced LiDAR modules would substantially improve throughput, detection stability, and overall responsiveness.To supplement our measurements, a theoretical comparison of hardware platforms is provided.

TABLE IX
HARDWARE EFFICIENCY COMPARISON

Ref.	**Platform**	**Acceler-ator**	**FPS**	**Power (W)**	**Notes**
-	**Raspberry Pi 4B**	None	**2–6 FPS**	5–7 W	Your deployment baseline
[28]	Raspberry Pi 4B	Coral USB TPU	15–25 FPS	6–8 W	Needs INT8 quantized model
[29] [30]	Jetson Nano	CUDA GPU	12–20 FPS	10–15 W	Good for small YOLO models
[31]	Xavier NX	CUDA GPU	25–45 FPS	12–20 W	High-performance edge compute

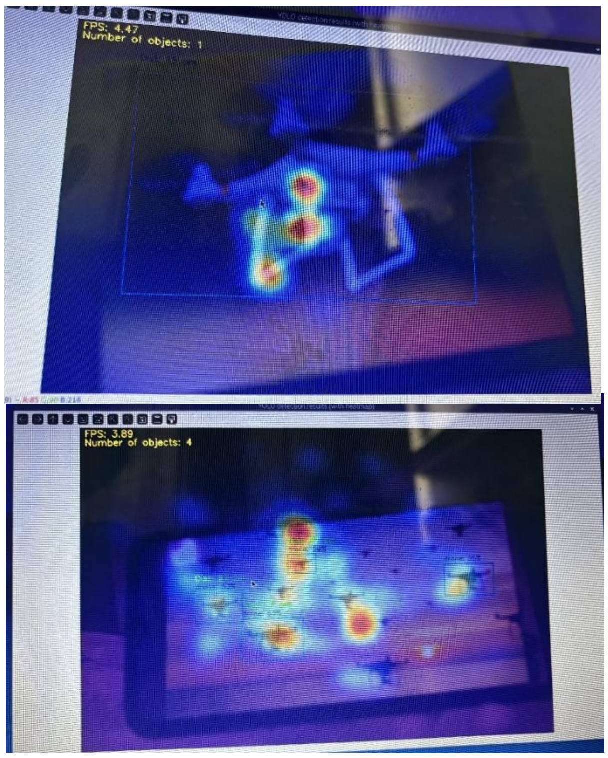


Fig. 10. Heatmap of the detection.

Performance can be significantly improved with minimal additional investment, particularly via Coral TPU or Jetson platforms.

V. CONCLUSION AND FUTURE WORK

A. Conclusion

The developed Multi-Drone Deployment and Surveillance System successfully demonstrates the feasibility of performing on-board drone detection and distance estimation using lightweight, edge-computing hardware. By integrating YOLOv8-Nano for vision-based detection with a TF-Luna LiDAR module for depth sensing, the system achieves reliable real-time perception, stable bounding-box visualization, accurate distance measurements, and seamless operation on the Raspberry Pi 4B. The combination of structured hardware design, efficient training workflows, synchronized sensing, and outdoor field validation confirms that the system is capable of supporting key tasks in aerial surveillance, collision avoidance, and multi-drone coordination.

Overall, the prototype establishes a strong foundation for practical drone-to-drone situational awareness in real-world environments. Its modular architecture, low latency, and scalability make it suitable for security applications, autonomous navigation, and distributed drone networks.

B. Future Work

Several enhancements can further advance the capabilities and robustness of the system:

1. ***Autonomous Evasion and Path Planning:*** Incorporate real-time decision-making logic enabling drones to autonomously avoid obstacles and compute alternative flight paths using fused LiDAR and YOLO predictions.
2. **Swarm Intelligence and Multi-Drone Coordination:** Enable communication among multiple drones to share detection output, support cooperative tracking, and expand surveillance coverage through coordinated maneuvers.
3. **Enhanced Low-Light and Night-Vision Performance:** Integrate thermal or infrared sensors and augment the training dataset with nighttime imagery to improve detection reliability under low-visibility conditions.
4. **Model Optimization for Increased Throughput:** Apply advanced optimization techniques such as quantization, pruning, distillation, or deploy hardware accelerators (e.g., Coral TPU, NVIDIA Jetson) to achieve higher FPS at reduced computational cost.
5. **Cloud-Connected Ground Station:** Extend the system to stream detection results, coordinates, and video to a remote server or cloud dashboard for centralized monitoring and analytics.
6. **Dynamic Obstacle Detection:** Expand the dataset to include birds, buildings, and other aerial obstacles, enabling a more comprehensive perception system suitable for autonomous navigation.
7. **Battery Optimization and Fail-Safe Mechanisms:** Incorporate energy-aware algorithms, automated return-to-home behavior, and emergency landing strategies to enhance safety during prolonged missions.
8. **Full Integration of Navigation and Surveillance Stack:** Unify detection, tracking, mapping, and flight control into a complete end-to-end autonomous surveillance platform suitable for industrial, commercial, or defense-grade operations.

VI. REFERENCES

[1] M. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "RF-based drone detection and identification using deep learning approaches," Future Generation Computer Systems, vol. 100, pp. 86–97, 2019.

[2] H. K. Kalluri, X. Li, and Y. Tang, "A survey of object detection for UAVs based on deep learning," Remote Sensing, vol. 15, no. 3, pp. 1–29, 2023.

[3] DroneRF Dataset (Al-Sa'd et al.), GitHub repository. [Online]. Available: https://github.com/dronerf/dataset. Accessed: Feb. 2025.

[4] S. S. Alam, A. Chakma, I. B. K. Utama, and Y. M. Jang, "RF-enabled deep-learning-assisted drone detection and identification: An end-to-end approach," Sensors, vol. 23, no. 9, p. 4202, 2023.

[5] J. Glüge, A. Wietfeld, and M. Meier, "Noisy drone RF signal classification: A robust deep learning approach," IEEE Access, vol. 11, pp. 50123–50135, 2023.

[6] O. Medaiyese, J. McCall, and M. L. Sichitiu, "CARDINAL: A realistic RF dataset for drone detection and classification," AERPAW Technical Report, 2022.

[7] S. Kim, H. Seo, and D. Kim, "RF-UAV: A benchmark dataset for UAV detection and identification using RF signals," IEEE Access, vol. 10, pp. 120431–120445, 2022.

[8] D. Aydin and J. Singha, "YOLOv5-based drone detection and performance analysis," Engineering, vol. 4, no. 2, pp. 626–641, 2023.

[9] H. Akcay and S. Ozturk, "Real-time drone detection using YOLOv4-Tiny for aerial surveillance," in Proc. IEEE Int. Symp. Electronics and Telecommunications, 2022.

[10] DroneDetect Dataset (Swinney & Woods), GitHub repository. [Online]. Available: https://github.com/drone-detect/dataset. Accessed: Feb. 2025.

[11] M. Kaur, S. Muttoo, and S. Sohi, "Vision-based air-to-air drone detection using single-stage deep detectors," IEEE Access, vol. 10, pp. 116235–116246, 2022.

[12] U. Seidaliyeva, L. Ilipbayeva, and A. Zhaksylyk, "LiDAR technology for UAV detection: From fundamentals to advanced classification techniques," Sensors, vol. 25, no. 9, pp. 1–23, 2025.

[13] T. A. Abir, K. B. Shafiullah, and M. Trefftz, "Towards robust LiDAR-based 3D detection and tracking of UAVs," in Proc. ACM MobiSys Workshop on Drone Systems (DroNet), 2023.

[14] Y. Su, D. Wang, and F. Liu, “Multi-level hazard detection using a UAV-mounted multi-sensor system,” Drones, vol. 8, no. 2, pp. 1–17, 2024.
[15] A. Semenyuk and R. F. Abd-Alhameed, “A review of multi-sensor approaches for UAV detection: Radar, optical, RF, and fusion,” IEEE Access, vol. 12, pp. 55621–55645, 2024.
[16] Kaggle Drone Detection Dataset. [Online]. Available: https://www.kaggle.com/datasets. Accessed: Feb. 2025.
[17] Roboflow Drone Dataset. [Online]. Available: https://universe.roboflow.com. Accessed: Feb. 2025.
[18] pathikg, “Drone-detection-dataset,” Hugging Face, 2024. [Online]. Available: https://huggingface.co/datasets/pathikg/drone-detection-dataset. Accessed: Feb. 2025.
[19] D. K. Alqahtani, A. Cheema, and A. Toosi, “Benchmarking deep learning models for object detection on edge computing devices,” arXiv preprint, 2024.
[20] H. Feng et al., “Benchmark analysis of YOLO performance on edge devices,” 2022.
[21] Benewake, “TF-Luna LiDAR Module — Product Specification,” Datasheet, 2024. [Online]. Available: https://www.benewake.com. Accessed: Feb. 2025.
[22] Benewake, “TF02 / TF02-Pro Single-Point ToF LiDAR — Specifications,” Datasheet, 2024. [Online]. Available: https://www.benewake.com. Accessed: Feb. 2025.
[23] Garmin Ltd., “LIDAR-Lite v4 LED — Operation Manual,” User Manual, 2023. [Online]. Available: https://www.garmin.com. Accessed: Feb. 2025.
[24] Velodyne Lidar Inc., “VLP-16 (Puck) Datasheet,” Product Specification Sheet, 2024. [Online]. Available: https://velodynelidar.com. Accessed: Feb. 2025.
[25] H. Akçay, S. Ozturk, and A. Yazici, “Real-time UAV detection using YOLOv5 and low-cost LiDAR fusion,” Sensors, vol. 22, no. 21, pp. 1–18, 2022.
[26] Z. Zhang, Y. Xu, and Q. Wang, “LiDAR–vision fusion for small UAV detection in low-altitude airspace,” IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 1231–1238, 2021.
[27] J. Chen, K. Wu, and M. Liu, “Lightweight drone detection using YOLOv4-Tiny and low-cost time-of-flight LiDAR,” in Proc. IEEE International Conference on Unmanned Aircraft Systems (ICUAS), 2023.
[28] Ultralytics, “Deploying YOLO models on Raspberry Pi and Coral Edge TPU,” Ultralytics Documentation, 2024. [Online]. Available: https://docs.ultralytics.com. Accessed: Feb. 2025.
[29] D. K. Alqahtani, A. Cheema, and A. Toosi, “Benchmarking deep learning models for object detection on edge computing devices,” arXiv preprint arXiv:2403.19750, 2024.
[30] NVIDIA Corporation, “Jetson Nano Developer Kit — Technical Specifications and Benchmarks,” NVIDIA, 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano. Accessed: Feb. 2025.
[31] NVIDIA Corporation, “Jetson Xavier NX Module — Product Performance Guide,” NVIDIA, 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-xavier-nx. Accessed: Feb. 2025.