IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151117

An Embedded Infrared Vision System for Security Surveillance

Parag Krishna Nikum[1], Sheetal Prashant Patil[2], Raj Ashok Sonawane[1],

Nikita Laxmikant Wagh[1], Tanvi Ritesh Badgujar[1]

Student, Department of Artificial Intelligence and Machine Learning, R. C. Patel Institute of Technology, Shirpur, India[1]

Associate Professor, Department of Artificial Intelligence and Machine Learning, R. C. Patel Institute of Technology, Shirpur, India[2]

Abstract: Surveillance cameras are only useful if they keep working when conditions turn bad: at night, in wind, under shifting shadows. We built a compact surveillance prototype on a Raspberry Pi 5 that pairs an infrared-capable camera with classical motion analysis and a small deep learning detector. Each frame goes to YOLOv8n first, which looks for people. Frames with no person are passed to a motion analysis stage, where OpenCV background subtraction and noise filtering pick out moving regions, and anything that moves but isn't recognised is flagged as a possible anomaly. An operator can label flagged objects so the model can be retrained later. We also compared YOLOv8n against MoveNet SinglePose for human detection and found YOLOv8n faster and steadier on this hardware, and running the detector through OpenVINO raised the frame rate on the Pi from about 1 to about 6 FPS. This paper describes the design, the reasoning behind each choice, and the weaknesses we ran into.

Keywords: embedded AI, infrared imaging, security surveillance, YOLOv8, OpenVINO, Raspberry Pi 5, anomaly detection, edge computing

I. INTRODUCTION

Watching a screen for hours is a poor use of a person. Attention drifts, and the one event that matters tends to happen when nobody is looking. That is the basic argument for automated surveillance, and it has grown stronger as cheap embedded boards and deep learning detectors have matured. Models in the YOLO family can now find people in a video stream fast enough for live use, and boards like the Raspberry Pi 5 can run them without a cloud connection. Processing at the edge avoids problems with latency, bandwidth, privacy and patchy connectivity, which matter a great deal in remote sites.

Real deployments are messier than benchmarks, though. Outdoor scenes have moving foliage, changing light and shadows that look a lot like intruders to a simple motion detector. Detectors trained on a fixed set of categories have a second weakness: they can only report what they were taught. Something unfamiliar crossing the frame is either misclassified or ignored. Night adds a third problem, since an ordinary camera sees almost nothing in the dark. Infrared imaging helps with that last point. An IR-CUT camera blocks infrared light during the day to keep colours accurate, then removes the filter and switches on infrared LEDs when light drops, giving a monochrome image that is still usable at night.

Our project combines these pieces into a single pipeline. It aims to detect people quickly on low-power hardware, notice unknown moving objects, cut down false alarms caused by the environment, and let a human feed corrections back into the system so it improves after deployment.

II. RELATED WORK AND RESEARCH GAP

The literature we reviewed falls into a few groups. Lightweight detectors built on YOLO variants have been shown to run on constrained hardware while keeping reasonable accuracy [2], [7], [12], and a YOLOv8 study on wildlife detection reports stable behaviour on infrared imagery [12]. Pose-based approaches such as MoveNet, distributed as a TensorFlow Lite model, are attractive for edge devices because of their small size, and they tend to do well in structured indoor scenes [9], [10]. For motion, adaptive background subtraction using MOG2 combined with contour analysis is a standard way to extract moving regions in outdoor video [8]. Anomaly detection work, including autoencoder-based methods, tries to flag events that deviate from learned normal behaviour [4], and hybrid frameworks combine detection with anomaly

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151117

analysis for outdoor scenes [13]. Finally, human-in-the-loop learning lets operators correct a system so it adapts to data it did not originally see [11], [14].

Reading these together, we identified five gaps. Detectors trained on fixed categories cannot reliably report things outside them [2], [7], [13]. Motion and anomaly methods react to foliage, shadows, rain and lighting changes [4], [8]. Accurate models want strong hardware, and models small enough for embedded boards usually give up some accuracy [7], [12]. Most systems stay static once installed [11], [14]. And many frameworks assume a powerful computer rather than a board like the Raspberry Pi 5 [6], [12]. No single work we found tackled all five together on embedded hardware, and that is the space this project tries to fill.

III.SYSTEM DESIGN

The prototype is deliberately small, and each part was picked for a practical reason, mostly to keep a board running continuously without trouble. The main unit is a Raspberry Pi 5 with 8 GB of RAM, a quad-core 64-bit Cortex-A76 at 2.4 GHz, which leaves room for frame buffers and several AI libraries at once. A Waveshare IR-CUT camera captures colour video by day and infrared video at night, switching filters automatically. Table I lists the components and Fig. 1 shows the two main ones.

TABLE I HARDWARE COMPONENTS

Component	Role
Raspberry Pi 5 (8 GB)	Main processor; enough RAM for frame buffers and AI libraries
Waveshare IR-CUT camera	Colour video by day, infrared video at night
Samsung EVO Plus microSD	Reliable storage under constant use
Official power supply	Avoids voltage drops that crash the board under load
Active cooler	Prevents thermal throttling during long runs
Micro HDMI cable, bumper case, card reader, capture card	Display output, protection, flashing the OS, video capture during development



Fig. 1. (a) Raspberry Pi 5 and (b) Waveshare IR-CUT camera.

The board runs 64-bit Ubuntu 22.04 LTS, chosen for stability and good support for AI frameworks. The code is written in Python because nearly every relevant library exposes a Python interface. OpenCV handles frame capture, MOG2 background subtraction, contour analysis and drawing. PyTorch and TorchVision run YOLOv8n, which Ultralytics implements natively in PyTorch, TensorFlow Lite runs MoveNet for the comparison described later, and OpenVINO is used to speed up inference on the Pi.

The processing flow is shown in Fig. 2. Every input frame goes to YOLOv8n first. If a human is detected, the region is checked by a custom ape detection model, which decides whether it is really a person (a human ROI) or an ape (an ape ROI). If no human is detected, the frame goes to motion analysis, then to multi-model object detection, which produces a region for whatever was found. All three branches end at the same output.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151117

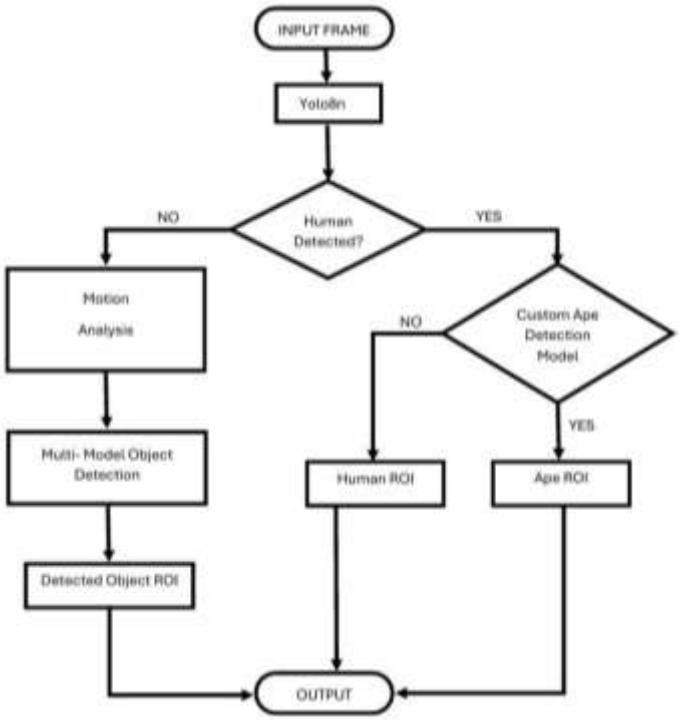


Fig. 2. System design flowchart.

IV. IMPLEMENTATION

The work is split into three modules: human detection, motion analysis, and feedback-based learning. The first question was which model should find people. We tried two candidates on the Raspberry Pi 5. MoveNet SinglePose, run through TensorFlow Lite, estimates body keypoints and infers a person from them. It is light and performs well in controlled indoor scenes, but it was built for pose estimation rather than general detection, and it is sensitive to occlusion and cluttered backgrounds [9], [10]. YOLOv8n is a nano-sized object detector that localises whole people directly [2], [7]. In continuous operation on the Pi, YOLOv8n ran at nearly twice the speed of MoveNet (Fig. 3). It also gave steadier bounding boxes, more consistent detections and fewer misses, including under infrared input (Fig. 4). We chose it as the primary detector. Its main limitation is the one every pretrained detector shares: it is restricted to the COCO classes it was trained on [7]. That limit showed up in practice: in one test frame an ape was reported as a person (Fig. 5b), which is a real problem for a system meant to watch forests.

```
0: 384x640 29 persons, 5.6ms
Speed: 1.2ms preprocess, 5.6ms inference, 1.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 28 persons, 6.3ms
Speed: 1.5ms preprocess, 6.3ms inference, 1.2ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 28 persons, 6.4ms
Speed: 1.3ms preprocess, 6.4ms inference, 1.4ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 28 persons, 6.4ms
Speed: 1.4ms preprocess, 6.4ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 28 persons, 7.9ms
Speed: 1.5ms preprocess, 7.9ms inference, 1.5ms postprocess per image at shape (1, 3, 384, 640)
```

Fig. 3. Per-frame timing output of YOLOv8n on the Raspberry Pi 5.

Speed was the main practical problem. Running the detector on the Pi at first gave only about 1 frame per second, which is too slow to follow a moving person. Switching to OpenVINO raised this to about 6 frames per second, as the FPS counter in Fig. 4 shows, even in an infrared view at night.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151117



Fig. 4. Human tracking on the Raspberry Pi 5 with OpenVINO, running at 6 FPS under infrared.

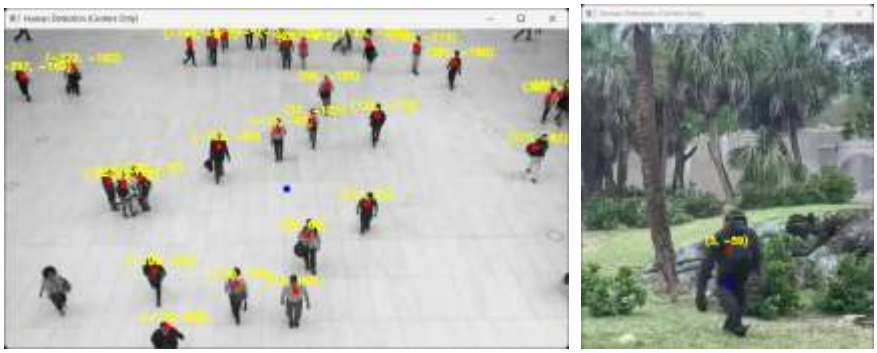


Fig. 5. (a) Human detection with YOLOv8n and (b) YOLOv8n mistaking an ape for a human.

The second module is what lets the system react to objects the detector has never heard of. OpenCV's MOG2 algorithm models each pixel as a mixture of Gaussians and updates the model continuously, so static scenery fades into the background while moving objects stand out as white shapes on black (Fig. 6) [8]. It is cheap enough for embedded use. Raw MOG2 output is noisy, though: shadows, swaying branches and brightness changes all produce small blobs. We discard any contour whose area falls below a threshold, which removes most of this clutter without touching real objects [4], [8].

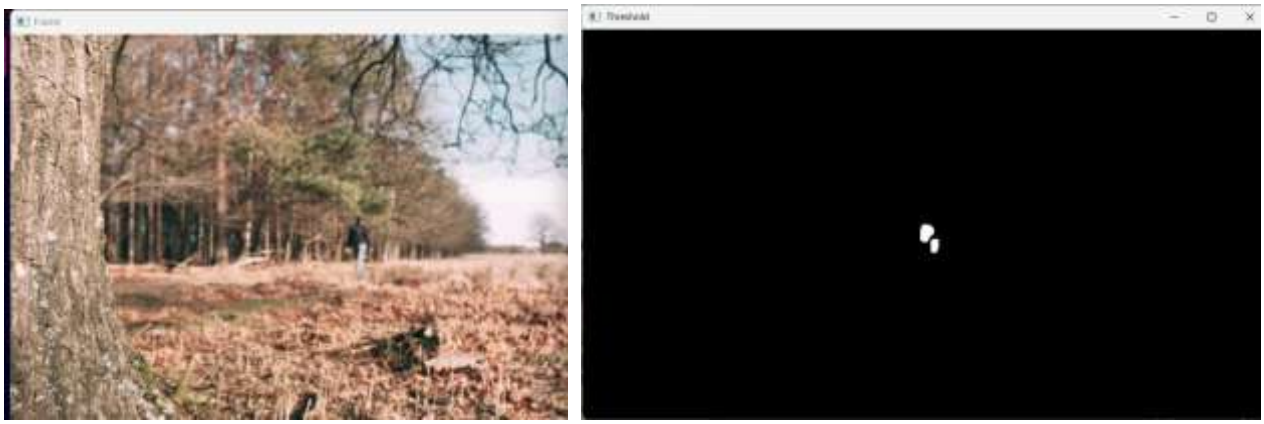


Fig. 6. (a) An input frame and (b) its MOG2 motion mask; moving pixels are white.

Surviving contours become regions of interest (ROIs). If the background changes abruptly, the whole frame briefly counts as motion. In testing, the system recovered within a few seconds and went back to isolating real movement (Fig. 7). Each ROI is then classified against the known classes, and a moving region that matches none of them is marked as an anomaly rather than dropped [4], [13].

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151117



Fig. 7. Adaptive motion detection with ROIs drawn around moving people.

Flagging an unknown object is only half of the job, because the system still has to learn what it was. When an operator labels a flagged ROI, the sample is saved locally along with its label, timestamp, confidence score and some environmental metadata. The files are organised in the directory layout YOLO expects, so retraining needs little preparation. The intended cycle is to collect labelled samples, format them, fine-tune the pretrained YOLOv8n weights, validate the result, and deploy the new weights to the device [11], [14]. We tested this idea on a concrete problem. Using around 3000 monkey images from a public Kaggle dataset [16], which covered different poses, lighting, infrared-style imagery, partial occlusion and varied backgrounds, we fine-tuned the pretrained yolov8n.pt model to add a monkey class (Fig. 8). The goal was to stop ape-like shapes from being mistaken for humans in forest scenes, and the added class improved separation between the two. The point is more general than monkeys: a deployment-specific dataset can teach the system things a stock model cannot know.



Fig. 8. Combined human and monkey detection after fine-tuning.

V. APPLICATIONS AND LIMITATIONS

The design suits places where people are scarce and nights are dark. Forest and remote-area monitoring can use it to spot intrusion, poaching or illegal logging. Wildlife researchers can use it to log animal movement without watching footage. Rescue teams could use infrared human detection to find people in low-visibility conditions after floods, landslides or fires, and the compact hardware could be mounted on patrol robots.

The prototype has clear limits. YOLOv8n is bound to its trained classes, so anything outside them depends on the motion stage and the operator rather than on the model itself. MOG2 needs a few seconds to settle after a sudden scene change, during which false alerts are likely. The learning loop depends on someone actually labelling samples, and retraining happens offline rather than on the device. The monkey class also only covers one confusion that we happened to observe. This report documents design and qualitative behaviour, and a larger evaluation with measured precision, recall and false-alarm rates would be the natural next step.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151117

VI. CONCLUSION

We built an embedded surveillance prototype that joins infrared imaging, a lightweight YOLOv8n detector, background-subtraction motion analysis and an operator feedback loop on a Raspberry Pi 5. YOLOv8n beat MoveNet on speed and consistency for this task, OpenVINO lifted the frame rate from about 1 to about 6 FPS, unrecognised movers are flagged instead of ignored, and fine-tuning with about 3000 monkey images showed how site-specific data can fix a known failure. The result is a small, practical base for systems that keep improving after they are installed.

REFERENCES

[1] J. Park, J. Chen, Y. K. Cho, D. Y. Kang, and B. J. Son, "CNN-Based Person Detection Using Infrared Images for Night-Time Intrusion Warning Systems," *Sensors*, vol. 20, no. 34, pp. 1–18, 2020.

[2] H. H. Nguyen, T. N. Ta, N. C. Nguyen, V. T. Bui, H. M. Pham, and D. M. Nguyen, "YOLO Based Real-Time Human Detection for Smart Video Surveillance at the Edge," in *Proc. IEEE ICCE*, Phu Quoc Island, Vietnam, 2021, pp. 439–444.

[3] N. Chandra and S. P. Panda, "A Human Intruder Detection System for Restricted Sensitive Areas," in *Proc. 2nd Int. Conf. Range Technology (ICORT)*, Chandipur, India, 2021, pp. 1–4.

[4] Z. Wang, H. Liu, and Y. Zhang, "Autoencoder-Based Anomaly Detection for Video Surveillance in Dynamic Scenes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 2456–2467, 2022.

[5] I. Shamta and B. E. Demir, "Development of a Deep Learning-Based Surveillance System for Forest Fire Detection and Monitoring Using UAV," *PLOS ONE*, vol. 19, no. 3, p. e0299058, Mar. 2024.

[6] Y. Abu Awwad, O. Rana, and C. Perera, "Anomaly Detection on the Edge Using Smart Cameras under Low-Light Conditions," *Sensors*, vol. 24, no. 3, p. 772, Jan. 2024.

[7] Q. Wang, G. Feng, and Z. Li, "A Lightweight Person Detector for Surveillance Footage Based on YOLOv8n," *Sensors*, vol. 25, no. 2, p. 436, Jan. 2025.

[8] M. Rashid, A. Khan, and S. Hussain, "Effective Background Subtraction for Outdoor Surveillance Using Adaptive MOG2 and Contour Analysis," *J. Vis. Commun. Image Represent.*, vol. 85, pp. 1–12, 2022.

[9] Y. Zhang and H. Liu, "MoveNet Pose Estimation for Embedded Human Detection: Evaluation and Optimization on Raspberry Pi," *IEEE Embedded Syst. Lett.*, vol. 15, no. 2, pp. 78–81, 2023.

[10] Google TensorFlow Team, "Pose Estimation and Classification on Edge Devices with MoveNet and TensorFlow Lite," *The TensorFlow Blog*, Aug. 2021.

[11] H. Ali, M. Farooq, and K. Ahmad, "Adaptive Deep Learning with Human-in-the-Loop for Unknown Attack Detection in Network Surveillance," *Applied Intelligence*, vol. 53, pp. 14231–14248, 2023.

[12] W. Kim, S. Park, and J. Lee, "Real-Time Wildlife Detection Using YOLOv8 and Infrared Camera Systems in Forest Environments," *Ecological Informatics*, vol. 75, p. 102064, 2023.

[13] P. Kumar, R. Mishra, and A. Sharma, "Hybrid AI Framework for Outdoor Surveillance: Combining Motion Detection and Object Classification," *Expert Syst. Appl.*, vol. 213, p. 118953, 2024.

[14] R. Singh and P. Mehta, "Human-in-the-Loop Adaptive Video Analytics for Continuous Learning Surveillance Systems," *IEEE Access*, vol. 12, pp. 45231–45248, 2024.

[15] H. S. Yoon and E. T. Kim, "YOLOv5-Based Lightweight Methodology for Real-Time Thermal Image Object Detection for Edge Devices," *J. Broadcast Eng.*, vol. 29, pp. 703–712, 2024.

[16] ishworsubedii, "Monkey and Person Detection," *Kaggle* dataset. [Online]. Available: https://www.kaggle.com/datasets/ishworsubedii/monkey-and-person-detection (accessed Oct. 27, 2026).

 This work is licensed under a Creative Commons Attribution 4.0 International License