International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Edge-Assisted IoT Smart Border Surveillance using YOLOv8 and Real-Time Cloud Alerts

Priyanka Arun Nehete[1], Prasad Bhosale[2], Rutika Shah[3], Sujeet More[4]

[1,2,3,4]Department of Computer Engineering, Trinity College of Engineering and Research, Pune, India

Abstract

The time lag between the detection of an event in the border and the execution of a physical response, occlusion of the object from a distance and the fatigue of the operator constitute the limitations of continuous border monitoring and restricted zone monitoring. The proposed architecture identifies a person using YOLOv8, conducts an interpretable (dominant-RGB) uniform check, sends the result to a rule set to the ESP32, and transmits a binary alarm state to the ESP32-controlled relay, buzzer and LED strip via the Firebase Realtime Database, making it an edge-assisted Internet of Things (IoT) surveillance architecture. Video is stored on a local edge computer and only small updates of the cloud are carried. It is implemented conceptually as a Python Flask application that has authentication, live video, status counts, and alert history. Public COCO, CrowdHuman and NightOwls resources support reproducibility, along with a public COCO reference benchmark for YOLOv8n, and a provided seed-controlled simulation of the colour and policy layers. The simulated uniform classifier achieved 95.56% accuracy and 95.42% F1 score at the RGB distance threshold of 55. The overall decision policy is 97.69% accurate, 96.42% precise, and 98.96% recall with 10,000 decision events. The simulated median end-to-end alarm latency was 305.9ms, the 95th percentile was 434.8ms, and network load was reduced by 99.973% compared to a 2Mb/s reference video stream. The results show that the architecture is possible, but are not a field deployment, and illumination sensitivity, camouflage ambiguity, Internet dependence, and a lack of a custom uniform detector remain important limitations.

Keywords: Border Surveillance, Edge AI, ESP32, Firebase, Internet of Things, Person Detection, Uniform Classification, YOLOv8

1. Introduction

Persistent observation of borders, military perimeters, and restricted facilities is difficult because camera networks produce more visual information than human operators can inspect reliably. The concept of IoT architectures is to integrate sensing, computation, communication, and actuation into a single operational loop [1]. This paved the way for quantitative detection research of large-scale visual data under different scenes [2]. One-stage detection took a step forward in reframing localisation and classification as a single regression problem, allowing for real-time processing [3]. Edge computing was then used to effectively process analytics near the camera, minimizing response time and bandwidth [4]. Online association methods were also shown to allow to preserve identities over short occlusions without batch processing [5]. However, benchmarks of crowded scenes also indicate that person detection is challenging when there is significant person overlap [6]. This motivation is expanded by bringing learned inference closer to the

data sources than to the remote cloud [7] (called edge intelligence). Later designs for the detector increased the accuracy-speed tradeoff by implementing a robust backbone, feature fusion, and training methods [8]. Model latency, network reliability, privacy, and energy are among the key factors to consider in surveying edge deep learning, which must be considered together [9]. YOLOv8 is recently shown to be a practical and effective anchor-free detector for monitoring applications with deployment-dependency performance [10].

The motivation for this work is therefore not only to recognise but to complete the operational loop from local perceived "see" to a guard alert "see and hear". The proposed system does not make any claim to a fully trained military-identity model, but rather a pretrained person detector that is then fused with an explainable colour rule. It is the main contributor to:

- An edge-to-cloud architecture with local processing of raw video and only a small alarm state is synced; and
- a mathematically-defined patrolling policy, which is headcount and dominant-RGB uniform consistent;
- a repeatable protocol for simulation; and
- description of an ESP32 relay interface and operator dashboard design which is appropriate for rapid prototyping.

The following literature review sets these design decisions in the context of previous detection research, colour analysis, software, dataset, and security research prior to the formal model.

2. Literature Review

2.1 Real-Time Person Detection and Edge Inference

To monitor the people through an IoT pipeline, the pipeline needs to be accurate enough to be able to identify the people and fast enough to enable the intervention. The integration of sensor, network and distributed intelligence is explained in detail in foundational IoT work [1] and COCO provides a variety of instance annotations for evaluators of detectors [2]. The single-pass model was coined by YOLO [3] and why that inference should be close to the camera is explained by edge computing [4]. The tracking-by-detection approach can alleviate duplicate counting and a Deep SORT [5] approach showed that an appearance metric can reduce identity switches. CrowdHuman [6] demonstrates the need for explicit occlusion testing in dense borders or checkpoints. The accuracy-latency trade-off is also discussed in Edge-intelligence studies [7] and YOLOv4 [8] and the edge/deep-learning survey [9] suggests the need for compute/communication joint evaluation. Although the YOLO architecture review [10] suggests that YOLOv8 is a suitable prototype, it does not imply that it is a definitive domain transfer.

2.2 Uniform Colour and Image Processing

The RGB uniform verification is cheap to compute, but is physically weak. The results of colour-constancy studies indicate that observed channel values vary as a function of illumination, and that normalization assumptions will minimize, but not remove, this variation [11]. Learned colour naming shows that the real world's colour categories are more wide-ranging than those in any laboratory [12]. The one threshold paradigm is presented as a principled example of threshold selection for separating two distributions [13] but is not valid in the presence of overlapping distributions. OpenCV provides the support required for capturing, converting colours, cropping, histogram and clustering operations for an easy implementation [14]. For this reason this paper adopts a dominant-colour centroid method and makes it explicit that a mismatch can be considered an intruder or it can be considered as “belonging” to the legitimate user.

2.3 Web, Cloud, and Embedded Alert Integration

Flask is an operator interface: the request routing and templates support dashboard pages that require authentication [15]. The Firebase Realtime Database stores the state in JSON, with REST endpoints that can be used to write an alert to be polled by embedded clients [16]. The ESP32 has been equipped with Wi-Fi and general-purpose input/output which can be used for relay control signals [17]. YOLOv8 [18] model variants, export path, and COCO reference values are published by Ultralytics. These parts make a prototype easier to develop, but there are aspects of database rules, authentication tokens, transport security, reconnect logic, and watchdog behaviour that need to be designed and built.

2.4 Robustness, Security, and Research Gap

NightOwls shows the extreme contrast, reflection, blur and lighting problems that occur when detecting pedestrians during the night. NIST's IoT baseline includes device identification, device configuration, data protection, interface restriction, secure updates, and cybersecurity-state awareness [20]. Anomaly detection studies show that richer temporal behaviour may be added to deterministic rules [21] and transformer-based real-time detectors offer a contemporary benchmark for future assessment [22].

The critical findings are edge inference lowers raw-video transmission, conventional detectors are extremely quick but domain-specific, RGB comparison is explainable but illumination-dependent, and cloud actuation adds latency and attack surface. The gaps identified are the lack of a uniform dataset across the border, insufficient reporting of perception-to-relay latency, the lack of consideration of network outages and insufficient testing covering different day, night, camouflage, and crowd conditions. The question of this study is: RQ1—Is a lightweight RGB policy viable for a high-recall patrol alert policy? RQ2: Is it possible to maintain the simulated end-to-end latency below 0.5 s for 95% of the cases? RQ3: What amount of bandwidth can be saved by sending state versus video? H1 predicts that more than 95% of alerts will be recalled, H2 predicts that latency will be below 500ms in 95% of cases, and H3 predicts that there will be a reduction of bandwidth above 99% in the case of H3. These questions are directly related to the formal problem and its system-aligned mathematical model.

3. Problem Formulation

3.1 Perception and Uniform Model

At time t let the frame I_t of video come in from R^{HxWx3}. YOLOv8 yields the detections with the class c_i, the confidence p_i, and the bounding box b_i. The accepted set, when using the person class c_p and confidence threshold γ, is

$$D_t=(b_i, p_i)|c_i=c_p \wedge p_i \geq \gamma \quad (1)$$

The possible number of people detected is therefore

$$N_t=|D_t| \quad (2)$$

A region R_i is cropped for each box b_i to minimize contamination from the sky, ground, and visible skin. Each pixel is quantized and grouped into K = 3 RGB groups, where the most dominant clothing vector is defined by the largest cluster S_i.

$$\mu_i=(1/|S_i|)\Sigma_x \in S_i x=[r_i, g_i, b_i]^T \quad (3)$$

Normalized Euclidean RGB distance is a measure of colour difference between a uniform colour μ_u and a given colour μ for the RGB colour model where each component is in the range [0,100].

$$d_i=\left|\left|\mu_i-\mu_u\right|\right|^2/\sqrt{\quad} \tag{4}$$

A detection is called uniform-consistent if the distance between the detection point and the true point is at most not larger than threshold τ:

$$u_i=1[d_i \leq \tau] \tag{5}$$

3.2 Patrolling-Aware Decision Rules

Let the variable P_t be 0 or 1, meaning patrolling or not, and S_t be the soldiers strength entered by the operator and $M_t=1[min_i u_i=0]$ be that at least one mismatch occurred. If there is no patrolling, when a person is detected, it is anomalous:

$$A_t^{(off)}=(1-P_t)\cdot 1[N_t>0] \tag{6}$$

In the patrol, excesses in numbers and colour will be assessed separately:

$$A_t^{(count)}=P_t\cdot 1[N_t>S_t] \tag{7}$$

$$A_t^{(colour)}=P_t\cdot M_t \tag{8}$$

The final binary state will be the logical AND of the three triggers:

$$A_t=A_t^{(off)} \vee A_t^{(count)} \vee A_t^{(colour)} \tag{9}$$

The edge service publishes a state, in the form of a JSON state with a timestamp to Firebase,

$$F_t=alert: A_t, people: N_t, soldiers: \Sigma_i u_i, time: t \tag{10}$$

3.3 Timing and Evaluation Objectives

The end to end response time takes into account the capture/preprocessing time, inference, decision, cloud write/read, ESP32 polling and relay actuation:

$$L_e{}^2{}_e=L_{pre}+L_{in}f+Ld_ec+Lc^{lo}{}_ud+L_{poll}+L_{relay} \tag{11}$$

The reported measures are for alert ground truth y and prediction $\hat{y}$.

$$Precision=TP/(TP+FP),$$
$$Recall=TP/(TP+FN), F1=2PR/(P+R) \tag{12}$$

This means that all variables are directly mapped to the architecture: I_t and D_t are at the edge, F_t is a cloud contract, and A_t is the ESP32 output. Facial recognition will not be used in the formulas intentionally, and intent may not be inferred.

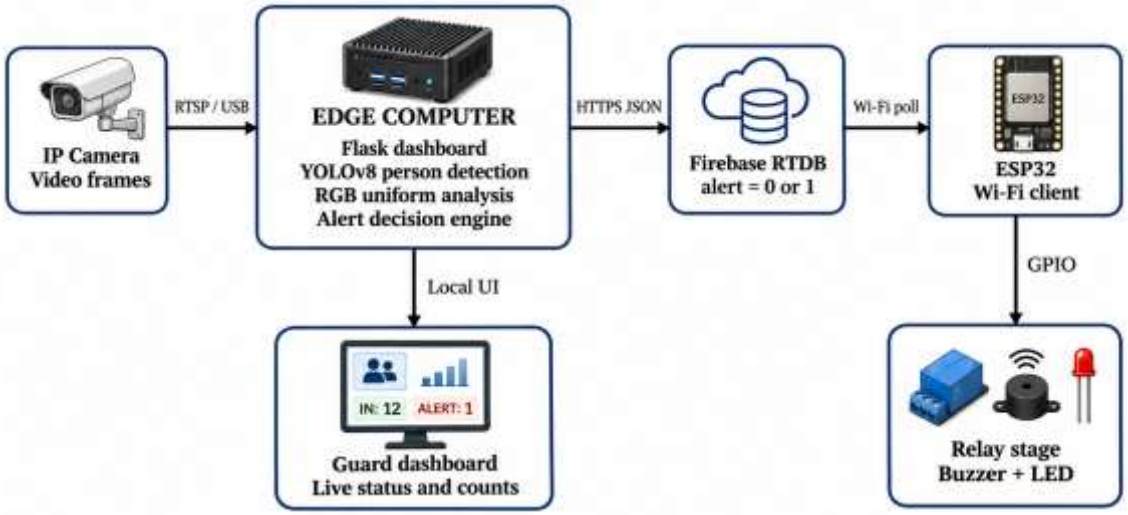
Problem Statement: Low cost local surveillance loop to detect people, flag patrol deviations based on count and clothing-colour evidence defined by an operator, and trigger a physical alarm in a remote location with bounded delay and reduced network traffic. The design needs to be replicable, decipherable and clear on uncertainty and needs to take into account that similarity in colour is insufficient to confirm identity.

4. Proposed Approach

4.1 System Architecture

Figure 1 presents the end-to-end sensing to actuation path: a camera feeds frames into a local edge computer, the edge application detects and classifies a person and carries a compact JSON state with it into Firebase, and the alarm is output via a relay through the ESP32. The guard dashboard is still local to the Flask service and therefore, even if there is a cloud outage, there is no loss of visual monitoring.

Figure Error! No sequence specified.**: Edge-to-Cloud Surveillance Architecture**



4.2 Frame Processing and Person Detection

The capture worker reads an RTSP, USB or recorded video source and records the time of each frame. The frames are resized to the input size of YOLOv8, which is 640 pixels, preserving aspect ratio and normalized in the network.The frames are resized to the input size of YOLOv8, which is 640 pixels, while maintaining aspect ratio and network-wise normalization. Class-person predictions with confidence γ = 0.50 only are kept. Boxes are deleted if they are redundant (due to non-maximum suppression). The present production configuration involves the addition of a tracker, like [5] to the production so that a count of the number of components shown on the dashboard does not change between adjacent frames; the present rule engine runs independently on every frame to match the prototype specification.

4.3 Dominant-Colour Uniform Check

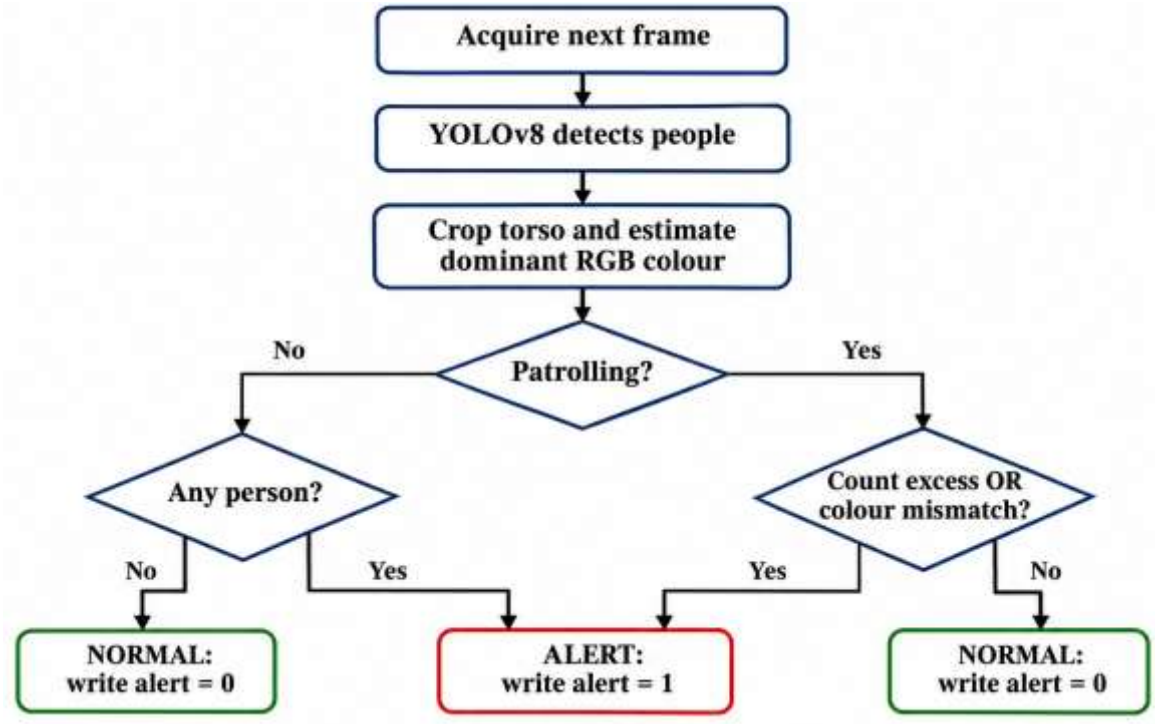
Person boxes that are accepted are attached to image bounds.Person boxes that are accepted are attached to image bounds. The proposed torso crop is about 20-65% of the box height, and 15-85% of the box

width, excluding a large amount of the head, hands and background. The crop is blurred, quantized and clustered into 3 centroids RGB. The operator-calibrated uniform vector is contrasted with the centroid that has the highest membership. Calibration is an average of multiple soldier samples collected by the camera installed. If the mismatch occurs at a τ value of 55 unnormalized RGB units, it is considered as possible evidence of enemy. This is because the flash, shadows, dust, camouflage and visually similar civilian clothing can all lead to false matches [11, 12].

4.4 Rule Engine, Cloud Contract, and Hardware

Figure 2 presents the frame-level decision logic based on patrolling status, personnel count, and uniform consistency When patrolling is not in effect, a positive person count increases the alert. If the patrolling is enabled, it is raised with a count greater than the entered soldier strength or if there is a colour mismatch. Only writes when the state changes, and a periodic event to expose stale connections, with the Flask process. The security rules of Firebase should be set to only allow writes to the edge service when it's authenticated, and only allow reads when the identity of the ESP32/client is specified [16, 20]. The ESP32 polls every 250 ms, discards old messages based on the timestamp and only turns on a GPIO output when it receives a valid alert. A relay module is connected to the GPIO and it separates the low voltage controller from the 12 V buzzer and LED strip. The buck converter provides the ESP32 rail from the 12 V adapter, common ground and a fuse suitable for loading.

Figure Error! No sequence specified.**: Patrolling-Aware Alert Decision Flow**



4.5 Web Application and Failure Handling

The Flask application supports registration, login, password reset initiation, dashboard, patrol configuration and alert-history routes [15]. Passwords need to be salted and hashed, reset links need to be single use and time limited. This dashboard displays the annotated stream, number of P_t, N_t, S_t, inferred number of soldiers or enemies, synchronization with the Firebase, ESP32 health, and alarm state. Pandas may be used only for event-log aggregation, while OpenCV performs video operations [14]. Two project

modules isolate responsibilities: detection_uniform.py returns boxes, counts, dominant colours and match flags; firebase_alert.py authenticates, retries and writes F_t. On no connectivity, the application records an event locally, displays CLOUD OFFLINE and may optionally run a directly attached alarm. Once it is reconnected, it will pass on the current state but not pass on the outdated alarms. Table 1 summarizes the major hardware and software components of the surveillance architecture. The layered design separates sensing, inference, cloud communication, and physical alert actuation.

Table Error! No sequence specified.**: System Components and Responsibilities**

Layer	Component	Primary responsibility
Sensing	IP/USB camera	Acquire day or recorded video frames
Edge AI	YOLOv8n + OpenCV	Detect people and estimate dominant RGB
Application	Flask + Pandas	Authenticate users, display state, log events
Cloud	Firebase RTDB	Synchronize compact timestamped alert JSON
Actuation	ESP32 + relay	Poll state and switch buzzer/LED load
Power	12 V adapter + buck	Supply alarm load and regulated controller rail

5. Experimental Setup And Results

5.1 Reproducible Evaluation Design

There are three types of evidence that are separated in the evaluation. First, YOLOv8n detector capability is not falsely marketed as a new training run, but is anchored to the publisher's COCO validation reference: 37.3% $mAP_{50–90}$ at 640 pixels, 3.2 million parameters, 8.7 GFLOPs, 80.4 mS CPU-ONNX latency, and 0.99 mS A100-TensorRT latency [18]. The cross-platform latency numbers are not equivalent to the simulated local-host latency below. Second, COCO 2017 validation, CrowdHuman, and NightOwls are listed as public resources for a future image-level reproduction. COCO is available at https://cocodataset.org/download; CrowdHuman at https://www.crowdhuman.org; and NightOwls at https://www.nightowls-dataset.org/download/. The COCO collection includes general scenes, the CrowdHuman collection focuses on occlusion with ~470,000 instances of humans in train/validation, and the NightOwls collection contains 279,000 frames in 40 sequences during the night. Licenses and access restrictions will need to be checked prior to redistribution.

Third, the new results (colour, rule, latency, bandwidth) announced above, are generated by the provided Python script reproduce_simulation.py using a PCG64 generator and a seed value of 20261119. This script is compatible with Python 3.11+, NumPy, and the absence of any proprietary data. Figure 3 illustrates the experimental configuration, including edge processing, public benchmark resources, cloud communication, and alert actuation. The simulator produces 5,000 distances (in cm) between a soldier and a colour (from a distribution with a clipped normal distribution, $N(28,12^2)$) and 5,000 distances (in cm) between an intruder and a colour (also a clipped normal distribution, $N(105,35^2)$) The following are stress distributions, not measurements taken from public image datasets. It then produces 10,000 events with patrol probability 0.70, with intruder probability 0.35, per-person detection probability 0.94, and Poisson false-detection rate 0.03. All value files and output csv files are provided in the “supplementary package”.

Figure Error! No sequence specified.**: Reproducible Experimental System Topology**

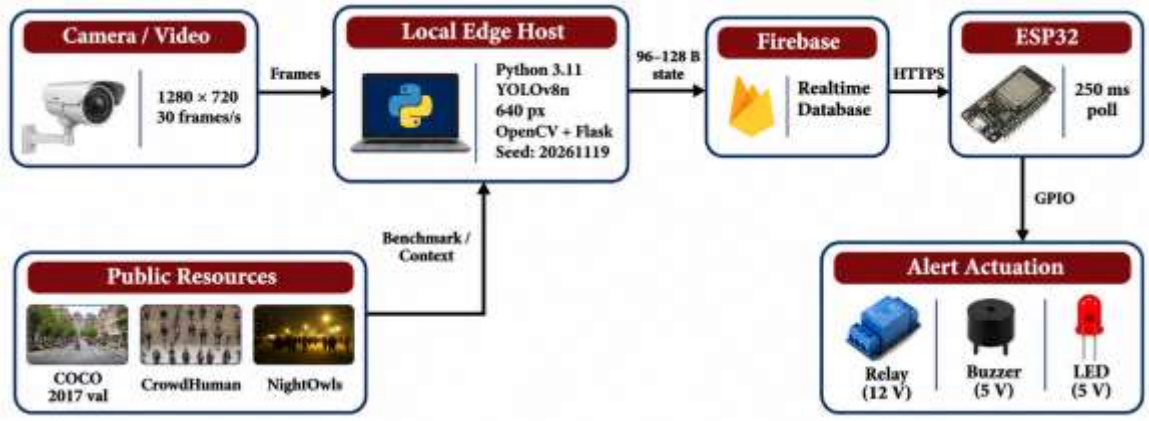


Table 2 presents the reference datasets, simulation parameters, and operating-point configuration used in the evaluation.

Table Error! No sequence specified.**: Datasets and Simulation Configuration**

Resource / parameter	Value	Role
COCO 2017 validation	5,000 images	Published YOLO reference context
CrowdHuman	15,000 train; 4,370 val; 5,000 test images	Crowd/occlusion extension
NightOwls	279,000 frames; 40 sequences	Night-time extension
Simulation seed	20261119	Exact random-state reproduction
RGB samples	10,000 total	Threshold sensitivity
Policy events	10,000	End-to-end rule evaluation
Confidence / RGB threshold	0.50 / 55 units	Operating point
ESP32 poll interval	250 ms	Latency stress parameter

5.2 Uniform-Threshold Results

At a threshold of 35, the precision went down to 77.71%, while 97.82% of the recall was kept with 1403 false alerts. The best observed accuracy was 95.56% ($\tau = 55$) with 92.42% recall and 65 false-positive, thus producing the best accuracy.The best accuracy (95.56%) was obtained at $\tau = 55$ with 92.42% recall and 65 false positive. At 75, precision was at 100%, while recall dropped to 81.12%, which is not the type of setting that should be used for a safety oriented alarm. The compromise operating point should thus be a compromise of nuisance alarms and mismatches missing, not only a point that achieves the highest accuracy. The distance distributions assumed are used to determine these values, and during a real deployment these must be recalibrated using images from the installed camera. Table 3 presents the simulated classification results across five RGB-distance thresholds.

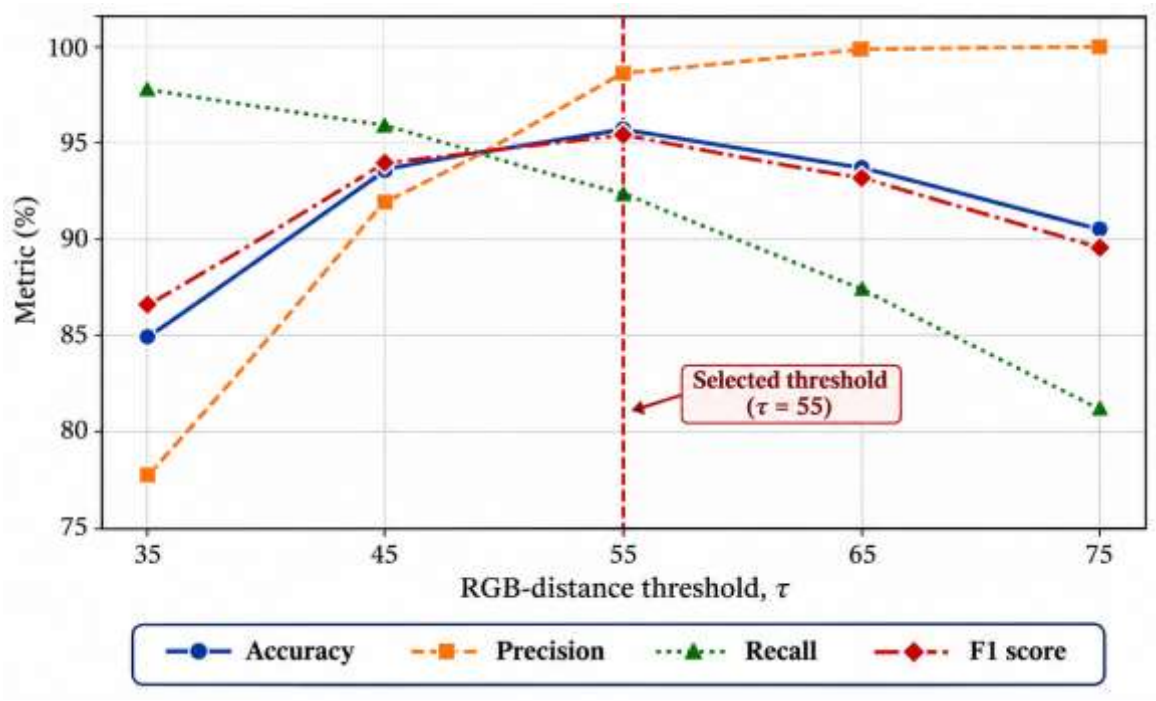
Table Error! No sequence specified.**: RGB Threshold Sensitivity Results**

τ	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	FP	FN
35	84.88	77.71	97.82	86.61	1403	109
45	93.81	92.07	95.88	93.94	413	206
55	95.56	98.61	92.42	95.42	65	379

τ	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	FP	FN
65	93.69	99.84	87.52	93.28	7	624
75	90.56	100.00	81.12	89.58	0	944

Figure 4 presents the influence of RGB-distance thresholds on accuracy, precision, recall, and F1-score.

Figure Error! No sequence specified.**: Uniform-Colour Threshold Sensitivity**



5.3 Alert-Policy Results

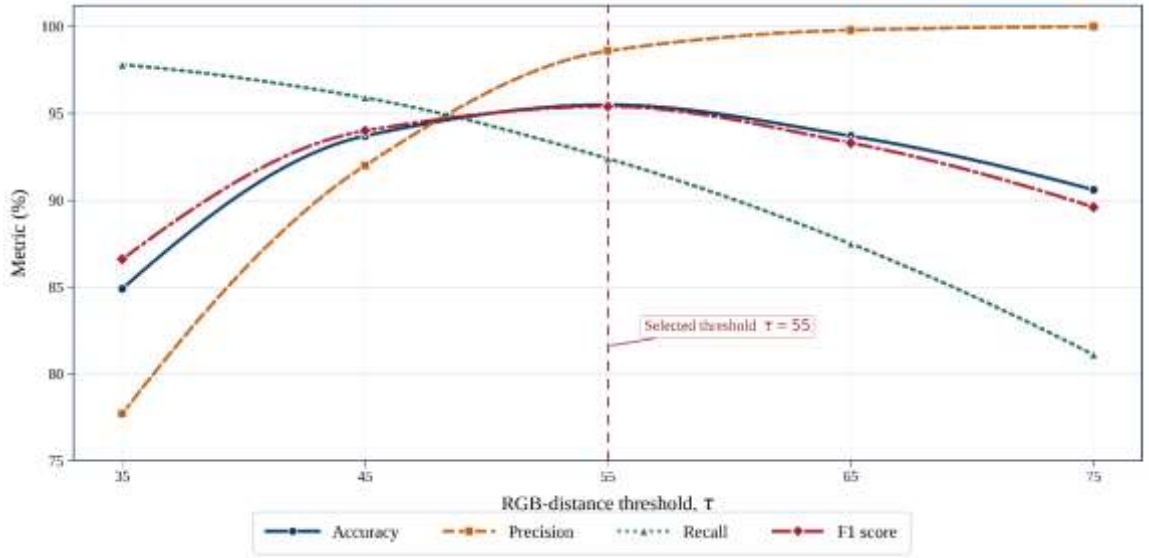
The event policy resulted in a total of 4,850 true positive events, 4,919 true negative events, 180 false positive events and 51 false negative events. Accuracy was 97.69%, precision 96.42%, recall 98.96%, and F1 97.67%. H1 is thus enabled in the simulation which was declared. This logic ORing of count excess with colour mismatch gives rise to high alert recall. Nuisance alarms are the same thing that occurs during the same fusion, when a false person detection raises N_t or an authorized uniform raises τ. This means that the results are guidelines for the implementation of the rule, rather than an indication of 97.69% accuracy in the field. Table 4 presents the confusion-matrix counts and classification metrics for the simulated alert policy.

Table Error! No sequence specified.**: Simulated Alert-Policy Performance**

TP	TN	FP	FN	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
4850	4919	180	51	97.69	96.42	98.96	97.67

Figure 5 illustrates the confusion matrix obtained from 10,000 simulated alert-policy events. The policy identifies 4,850 true positives and 4,919 true negatives, achieving 97.69% accuracy and 98.96% recall.

Figure Error! No sequence specified.**: Alert-Policy Confusion Matrix**



5.4 Latency and Network Load

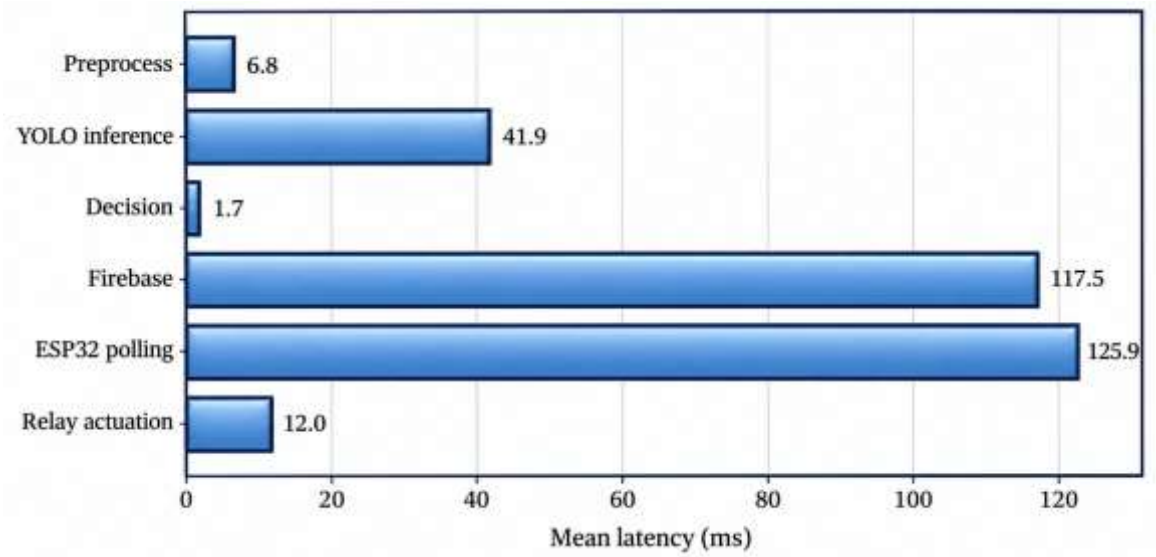
For a simulated 10,000 transactions for alerts, latency was simulated. However, in the simulation, there is no support for H2 because of the following design considerations: 6.8 ms for preprocessing, 41.9 ms for inference, 1.7 ms for decision means, 117.5 ms for Firebase transport, 125.9 ms for ESP32 polling interval, and 12.0 ms for relay actuation, yielding a total median, mean, and 95 percentile latency of 305.9, 305.8, and 434.8 ms respectively. The budget is dominated by cloud transport and polling, so that a smaller detector optimization or server-sent updates, MQTT or shorter adaptive poll interval would be more beneficial. The alarm path requires just 0.00053248 Mb/s for a 64-byte heartbeat per second and 128-byte alerts at 0.02 events/s at an available link of 1 Mb/s. This corresponds to a reduction of 99.973% when compared to a 2 Mb/s reference stream based on H.264, which is supported by H3. This is a comparison without protocol headers and it may need to be confirmed by packet capture. Table 5 presents the mean, median, and 95th-percentile latency measurements obtained from simulation.

Table Error! No sequence specified.**: Simulated Latency Summary**

Measure	Mean / value	Median	95th percentile
Preprocessing	6.8 ms	6.8 ms	8.5 ms
YOLO inference	41.9 ms	41.9 ms	50.3 ms
Firebase transport	117.5 ms	111.8 ms	185.1 ms
ESP32 polling	125.9 ms	126.6 ms	237.7 ms
End-to-end latency	305.8 ms	305.9 ms	434.8 ms

Figure 6 presents the simulated latency contributions of preprocessing, inference, decision logic, cloud transmission, polling, and relay actuation. Firebase communication and ESP32 polling dominate the mean response time, indicating the principal opportunities for latency reduction.

Figure Error! No sequence specified.**: Simulated Alert-Latency Components**



5.5 Dashboard Evidence and Reproduction Procedure

Figure 7 illustrates the surveillance dashboard using a representative simulated patrol-alert scenario. The patrolling is on, the assigned strength is 4, 5 people were detected, 4 matched the reference colour, 1 was flagged, Firebase stored alert = 1 and the selected latency displayed was 306 ms. This is a design artefact, not a picture of an actual deployed border system. To reproduce the numerical results, create a clean Python environment, install NumPy, run python reproduce_simulation.py from the package root, and compare the resulting CSV and JSON files with Tables 3–5; also keep the person class as is, set the image size to 640 and the confidence to 0.50, record software/hardware versions, and report image-level AP and event-level recall separately. For any trained model at least three independent runs should be reported.

Figure Error! No sequence specified.**: Experimental Dashboard for Surveillance**



6. Discussion

The results validate the feasibility of the architecture and reveal the source of the apparent strength of this architecture. Additionally, count excess and colour mismatch act as additional triggers leading to 98.96% simulated recall. That's an attractive operational redundancy, but the 180 false alerts demonstrate the price of treating any colour deviation as suspicious. A further check of threshold sensitivity can help establish that a small change in τ has a significant impact on precision and recall. Median (305.9 ms) and 95th percentile (434.8 ms) delays are within the proposed sub-0.5 s simulation target, but 80% of mean delay is caused by the transport delay of Firebase and polling delay of ESP32, not the inference delay. However, Edge AI does save bandwidth and enable fast perception, and cloud and embedded scheduling remain in play for alarm responsiveness. There are also many advantages in terms of the fact that the traffic reduction is very high, here 99.973% (raw video is not uploaded, this reduces the exposure of identifiable footage). However, if a binary Firebase value doesn't include authenticated timestamps, sequence numbers and expiry logic, it can be replayed or stale. Most of all clothing colour isn't identity. If that same kind of clothing, different lighting, partial obscuration, camouflage or intentional spoofing is used, the rule is broken. It should be designed as a prototype for decision making rather than as an autonomous lethal or identity verification system with human verification.

7. Conclusion

This paper proposed a smart border surveillance prototype system using IoT, which consists of local YOLOv8 person detection, dominant-RGB uniform screening, patrolling-aware rules, Firebase state synchronization and ESP32 relay actuation. The contribution is not a statement of a completed field validation, but rather a perception-to-alarm model that is coherent and reproducible. With the provided seed-driven stress simulation, the chosen RGB threshold set yielded an accuracy of 95.56% and F1 of 95.42% while the full alert policy yielded an accuracy of 97.69% and a recall of 98.96%. With median end-to-end latency of 305.9 ms and 95th percentile at 434.8 ms, compact state exchange resulted in reducing the estimated traffic by 99.973% when compared with a 2 Mb/s video stream. These results only justify the three hypotheses with the assumptions made in the documented scope. For deployment in real-world situations, border-specific data, camera-level calibration, secure device provisioning, network-failure testing, night and thermal evaluation, and independent ethical and operational review is required.

8. Future Work

Future studies should use a custom, uncertainty calibrated uniform detector trained on consented, geographically representative data instead of the colour rule. An evaluation of a multi-spectral pipeline with visible, low-light and thermal cameras should be considered based on night, dust, rain, crowd, camouflage and occlusion strata. Add GPS coordinates, signed mobile notifications, acknowledgement workflows and offline alarm fallback. Measures of energy and latency would be possible if quantized inference was performed on hardware such as a Jetson or a Raspberry Pi. Duplicate suppression by tracking and temporal anomaly models [21] may be useful and RT-DETR-style detectors [22] may be used as a baseline. Last but not least, before conducting a field pilot, there should be penetration testing, secure over-the-air updates, audit logs and human in the loop operating procedures.

References

1. L. Atzori, A. Iera, G. Morabito, “The Internet of Things: A survey”, Computer Networks, 2010, 54

(15), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010
2. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, "Microsoft COCO: Common objects in context", Computer Vision—ECCV 2014, 2014, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48
3. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You only look once: Unified, real-time object detection", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 779–788. https://doi.org/10.1109/CVPR.2016.91
4. W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, "Edge computing: Vision and challenges", IEEE Internet of Things Journal, 2016, 3 (5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198
5. N. Wojke, A. Bewley, D. Paulus, "Simple online and realtime tracking with a deep association metric", Proceedings of the IEEE International Conference on Image Processing, 2017, 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962
6. S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, J. Sun, "CrowdHuman: A benchmark for detecting human in a crowd", arXiv preprint arXiv:1805.00123, 2018. https://doi.org/10.48550/arXiv.1805.00123
7. Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing", Proceedings of the IEEE, 2019, 107 (8), 1738–1762. https://doi.org/10.1109/JPROC.2019.2918951
8. A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, "YOLOv4: Optimal speed and accuracy of object detection", arXiv preprint arXiv:2004.10934, 2020. https://doi.org/10.48550/arXiv.2004.10934
9. X. Wang, Y. Han, V.C.M. Leung, D. Niyato, X. Yan, X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey", IEEE Communications Surveys & Tutorials, 2020, 22 (2), 869–904. https://doi.org/10.1109/COMST.2020.2970550
10. J. Terven, D.-M. Córdova-Esparza, J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS", Machine Learning and Knowledge Extraction, 2023, 5 (4), 1680–1716. https://doi.org/10.3390/make5040083
11. G.D. Finlayson, E. Trezzi, "Shades of gray and colour constancy", Proceedings of the 12th Color and Imaging Conference, 2004, 37–41. https://doi.org/10.2352/CIC.2004.12.1.art00008
12. J. van de Weijer, C. Schmid, J. Verbeek, D. Larlus, "Learning color names for real-world applications", IEEE Transactions on Image Processing, 2009, 18 (7), 1512–1523. https://doi.org/10.1109/TIP.2009.2019809
13. N. Otsu, "A threshold selection method from gray-level histograms", IEEE Transactions on Systems, Man, and Cybernetics, 1979, 9 (1), 62–66. https://doi.org/10.1109/TSMC.1979.4310076
14. G. Bradski, "The OpenCV library", Dr. Dobb's Journal of Software Tools, 2000.
15. Pallets Projects, "Flask documentation, version 3.1.x", 2026, accessed on November 19, 2026. https://flask.palletsprojects.com/
16. Google, "Firebase Realtime Database REST API: Installation and setup", 2026, accessed on November 19, 2026. https://firebase.google.com/docs/database/rest/start
17. Espressif Systems, "ESP32 Series Datasheet", 2025, accessed on November 19, 2026. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
18. Ultralytics, "Ultralytics YOLOv8 documentation and performance metrics", 2026, accessed on November 19, 2026. https://docs.ultralytics.com/models/yolov8/

19. L. Neumann, M. Karg, S. Zhang, C. Scharfenberger, E. Piegert, S. Mistr, O. Prokofyeva, R. Thiel, A. Vedaldi, A. Zisserman, B. Schiele, "NightOwls: A pedestrians at night dataset", Computer Vision—ACCV 2018, 2019, 691–705. https://doi.org/10.1007/978-3-030-20887-5_43
20. M. Fagan, K.N. Megas, K. Scarfone, M. Smith, "IoT Device Cybersecurity Capability Core Baseline", National Institute of Standards and Technology Interagency Report (NISTIR) 8259A, 2020. https://doi.org/10.6028/NIST.IR.8259A
21. G. Pang, C. Shen, L. Cao, A. van den Hengel, "Deep learning for anomaly detection: A review", ACM Computing Surveys, 2021, 54 (2), Article 38, 1–38. https://doi.org/10.1145/3439950
22. Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, J. Chen, "DETRs beat YOLOs on real-time object detection", Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, 16965–16974. https://doi.org/10.1109/CVPR52733.2024.01605

Licensed under Creative Commons Attribution-ShareAlike 4.0 International License