Real-Time Edge Intelligence in UAV for Search-and-Rescue: Onboard Energy-Efficient Video Summarisation with Reduced Data Transmission

VINCENT, Chidike, HIRSCH, Laurence <http://orcid.org/0000-0002-3589-9816>, UL HASAN, Najam, FERNANDES, Caren, CRAINIC, Adriana, ZASADA-JAMES, Jonathan and BALDWIN, James

RESEARCH-ARTICLE

Real-Time Edge Intelligence in UAV for Search-and-Rescue: Onboard Energy-Efficient Video Summarisation with Reduced Data Transmission

Published: 18 November 2025

IOT 2025: The 15th International Conference on the Internet of Things
November 18 - 21, 2025
Vienna, Austria

IOT '25: Proceedings of the 15th International Conference on the Internet of Things (November 2025)
https://doi.org/10.1145/3770501.3770508
ISBN: 9798400715952

Real-Time Edge Intelligence in UAV for Search-and-Rescue: Onboard Energy-Efficient Video Summarisation with Reduced Data Transmission

Chidike Vincent
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, England, United Kingdom
vchidike@outlook.com

Laurence Hirsch
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, United Kingdom
l.hirsch@shu.ac.uk

Najam Ul Hasan
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, United Kingdom
n.ul-hasan@shu.ac.uk

Caren Crizben Ezlin Fernandes
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, United Kingdom
Caren.Fernandes@shu.ac.uk

Adriana Crainic
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, United Kingdom
adriana.crainic@student.shu.ac.uk

Jonathan Zasada-James
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, United Kingdom
j.zasada-james@shu.ac.uk

James Baldwin
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, United Kingdom
j.baldwin@shu.ac.uk

Abstract

Rapid detection of individuals in search and rescue (SAR) operations is critical for minimising casualties and enabling effective relief coordination. Unmanned aerial vehicles (UAVs) provide real-time situational awareness, but continuous high-resolution video transmission causes bandwidth constraints, delays decisions, and drains onboard energy. This paper introduces a modular UAV system for energy-efficient onboard video summarisation tailored for SAR. The system integrates lightweight YOLOv5n object detection and histogram-based keyframe extraction on a Raspberry Pi 4, transmitting only critical frames to ground teams. Built with off-the-shelf components, the UAV optimises weight and power while supporting on-device edge processing. The results show significant reductions in energy consumption and data transmission compared to full-video processing, maintaining high detection accuracy. This adaptable, low-power framework enhances mission duration and decision-making speed, offering a scalable solution for bandwidth constrained, energy-aware aerial monitoring in disaster response scenarios.

IOT 2025, Vienna, Austria

ACM ISBN 979-8-4007-1595-2/25/11
https://doi.org/10.1145/3770501.3770508

CCS Concepts

• **Computer systems organisation** → **Embedded systems**; • **Computing methodologies** → *Object detection*; • **Information systems** → *Video search.*

Keywords

UAV, search and rescue, video summarisation, edge computing, energy modelling

ACM Reference Format:
Chidike Vincent, Laurence Hirsch, Najam Ul Hasan, Caren Crizben Ezlin Fernandes, Adriana Crainic, Jonathan Zasada-James, and James Baldwin. 2025. Real-Time Edge Intelligence in UAV for Search-and-Rescue: Onboard Energy-Efficient Video Summarisation with Reduced Data Transmission. In *The 15th International Conference on the Internet of Things (IOT 2025), November 18–21, 2025, Vienna, Austria.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3770501.3770508

1 Introduction

UAVs have become indispensable in disaster management, offering rapid environmental assessment and critical support in hazardous situations [4]. Equipped with advanced aerial imaging and remote sensing technologies, UAVs generate extensive video data during search and rescue (SAR) operations, where timely access to the affected areas is critical. Historical events, such as the 2015 Nepal earthquake and 2019 Mozambique cyclones, have demonstrated the value of using UAV technology in mapping disaster zones and identifying infrastructure damage [23]. However, traditional UAV-based disaster response systems face significant challenges due to reliance on continuous transmission of unfiltered, high-resolution video to

ground stations [16]. This creates substantial communication bottlenecks, increases decision-making latency, and accelerates onboard energy depletion, particularly in regions with limited connectivity.

To address these limitations, this study presents a UAV system that leverages edge computing to perform onboard video summarisation for SAR operations. By integrating lightweight computer vision models, such as YOLOv5n, with a Raspberry Pi 4, the system processes video streams on-device, identifying and prioritising contextually relevant frames—those containing human subjects or structural damage—for selective transmission to ground control. This approach significantly reduces data volume and energy consumption while maintaining the accuracy and relevance of transmitted information. Advances in compact processing platforms [25, 26] and optimised neural network architectures, including MobileNet and TensorFlow Lite [7, 20], enable efficient object detection and classification tailored for emergency response.

The proposed framework minimises bandwidth requirements and enhances operational efficiency by filtering non-essential content, ensuring that ground teams receive decision-ready data quickly. This paper presents a deployable, energy-aware, and bandwidth-efficient framework for UAV-based disaster response, addressing the critical gap between data acquisition and real-time information delivery in post-disaster environments.

The primary contributions of this work include:

(1) **Adaptive video summarisation algorithm**: a novel algorithm optimised for disaster response that intelligently prioritises video content based on emergency-relevance criteria.
(2) **Resource-efficient edge AI pipeline**: implementation of a processing pipeline designed for UAV deployment that operates within the constraints of limited computational power and energy availability.
(3) **Operational validation**: evaluation on recorded SAR-like footage and device-level measurements, demonstrating large reductions in transmitted data and energy use under on-device processing.

2 Related Work

2.1 Recent Studies

Unmanned aerial vehicles are vital for disaster response, enabling rapid aerial scans to detect victims, hazards, and infrastructure damage in areas inaccessible to ground teams [1]. Modern UAVs support autonomous tasks such as simultaneous localisation and mapping, waypoint navigation, and collision avoidance [23]. However, their 20–30-minute flight times are limited by size, weight, and power (SWAP) constraints [25]. Continuous video streaming reduces endurance, necessitating energy-efficient processing [3].

Suo et al.'s E-UAV system optimises energy by adjusting flight parameters and algorithm settings, achieving savings in detecting people and vehicles without compromising accuracy [25]. Lightweight convolutional neural networks (CNNs), particularly the YOLO family, are widely used for UAV-based object detection because of their speed–accuracy balance [3]. The YOLOv5n (nano) variant, with 1.9 million parameters, is suited for resource-constrained UAVs [17]. Boddu and Mukherjee [3] demonstrated YOLOv5's effectiveness in emergency tasks such as detecting ambulances or fires. Mandula et al [17] implemented YOLOv5n on an embedded Jetson platform with TensorRT optimisations for real-time inference with minimal power use.

YOLOv5n has also been used for embedded wind-turbine fault detection, delivering practical accuracy with low compute cost [18], illustrating suitability for resource-constrained aerial analytics. Continuous video streaming consumes substantial energy, prompting research into onboard video summarisation. Zhang et al [27] developed a system that transmits keyframes of detected objects, achieving over 1000× data reduction. Recent studies use video synopsis, combining UAV camera and LiDAR data to create compressed clips of salient trajectories, reducing bandwidth and energy demands [9].

This work integrates histogram-based keyframe selection with a lightweight detector on a Raspberry Pi 4 to transmit only mission-relevant frames, reducing energy use and uplink in SAR. A recent study reports comparable energy use between on-board (YOLOv3-Tiny on a Raspberry Pi) and off-board Wi-Fi streaming when motors are off, highlighting that transmission can rival compute in some setups [19]. In contrast, this work reduces the detection workload via keyframe summarisation before inference, so fewer frames are processed and transmitted, which lowers total energy despite short, higher CPU bursts on selected frames.

2.2 Edge Computing and Energy-Efficient Visual Inference

Edge computing empowers UAVs to process video analytics onboard, significantly reducing latency and energy consumption compared to cloud offloading, which is often infeasible in disaster zones due to unreliable or non-existent network connectivity [8, 14, 30]. This approach is critical for real-time SAR operations, where immediate decision-making is essential to locate victims or assess hazards.

Keyframe extraction is a cornerstone of efficient video summarisation, enabling UAVs to identify and transmit only the most relevant frames. Techniques such as histogram differences or visual motion analysis select representative frames, minimising data transmission [24]. However, advanced models, such as CNNs combined with long short-term memory (LSTM) networks, provide superior semantic understanding but are computationally prohibitive for resource-constrained embedded platforms like the Raspberry Pi [21]. Lightweight alternatives are therefore preferred for onboard processing.

The YOLOv5n model, with only 1.9 million parameters, is highly optimised for resource-constrained platforms like the Raspberry Pi, delivering a robust balance of detection accuracy and computational efficiency [10]. Figure 1 compares YOLO variants commonly used on edge platforms. Compared to newer models such as YOLOv8 and YOLOv11, YOLOv5n benefits from extensive community support, mature deployment pipelines, and compatibility with ARM-based edge devices, making it ideal for UAV applications [12, 13, 15]. Mastia et al [9] utilised YOLOv5n for wind-turbine fault detection, transmitting only critical images tagged with GPS coordinates, which significantly reduced energy and bandwidth usage. By integrating YOLOv5n with histogram-based keyframe extraction, this study develops an adaptive video summarisation pipeline tailored

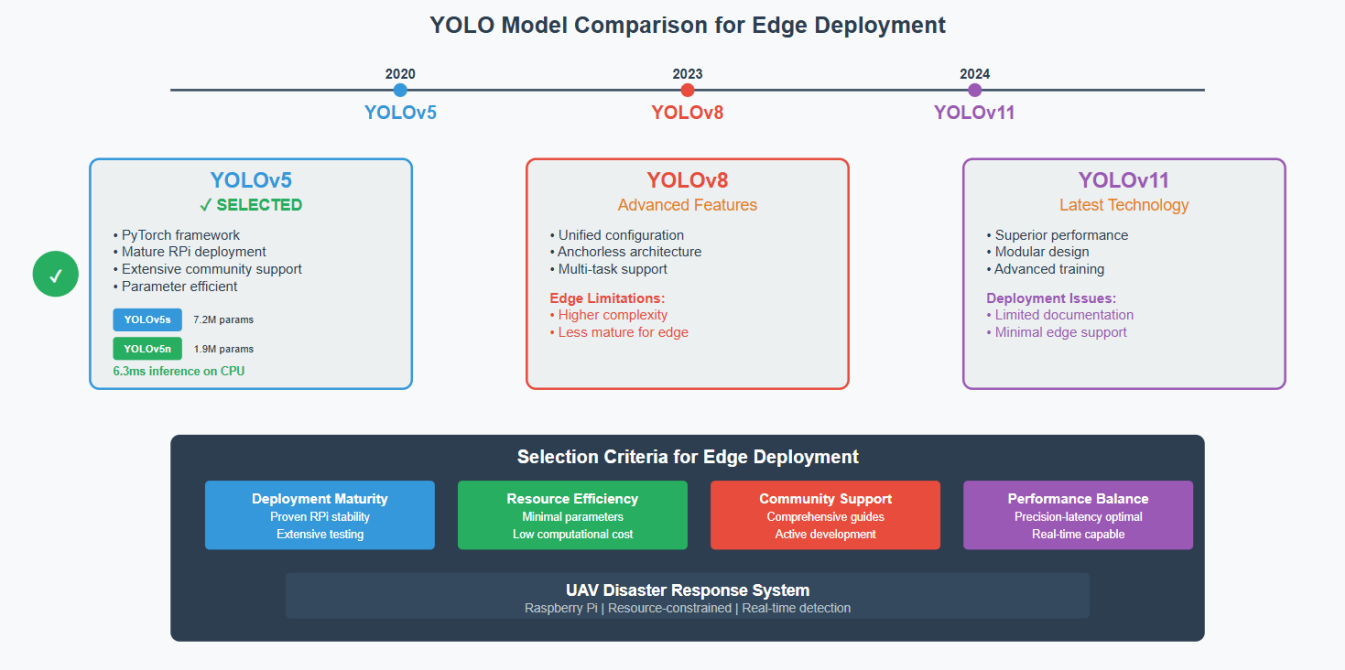

Figure 1: YOLO model comparison for edge deployment. Variants differ in parameter counts and inference speed.

for SAR. This approach optimises onboard processing, minimises data transmission, and extends UAV mission duration, enhancing real-time decision-making in disaster response scenarios.

Unlike prior edge-only or cloud-offload pipelines, this study applies summarisation first and runs detection only on selected keyframes, with energy quantified from device-level CPU measurements.

3 System Design and Implementation

3.1 Overview

This section outlines the complete design and development of a custom-built UAV system designed for energy-efficient video summarisation in search and rescue operations. It covers both the hardware construction of the drone and the software pipeline developed to process and reduce live video footage before transmission. The overall system architecture is depicted in Figure 5.

3.2 Hardware Architecture

A custom UAV was built using readily available components, making it both scalable and energy-efficient for onboard processing. The Raspberry Pi 4 Model B (4 GB RAM) was used as the main computing unit because it is small, has moderate power consumption, and is powerful enough to handle computer vision tasks in real-time. The component stack and specifications are summarised in Figure 2.

The UAV uses a 5-inch TBS Source One V5 frame. Four iFlight XING-E Pro2207 2450 KV motors, each paired with 5×4.3×3 three-blade propellers, provide enough thrust to support the UAV's full weight, including the Raspberry Pi and sensors. Power is supplied

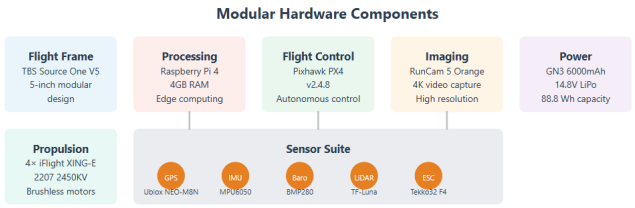

Figure 2: Detailed breakdown of all hardware components, including flight frame, processing unit, sensors, and power management systems with their specifications.

by a 6000 mAh 14.8 V LiPo battery, which connects to a HolyBro power distribution board; an adapter was used to join XT90 (battery) to XT60 (board). A Pixhawk PX4 2.4.8 controller manages flight control, with a Tekko32 F4 4-in-1 50 A ESC driving the motors.

The UAV includes several sensors: a RunCam 5 Orange 4K camera, a u-blox NEO-M8N GPS, a BMP280 pressure sensor, an MPU6050 motion sensor, and a TF-Luna LiDAR. Sensors were linked via I2C, UART, or GPIO as required. When two devices needed the same port or pins were limited, small adapters and simple custom cables were used. Components were fixed in place using standoffs, foam tape, and zip ties to prevent vibration. The battery sits underneath the drone and is held with Velcro, making it easy to remove and charge. Heat is managed through airflow and spacing between components; no fan was needed. The assembled platform layout is shown in Figure 3.

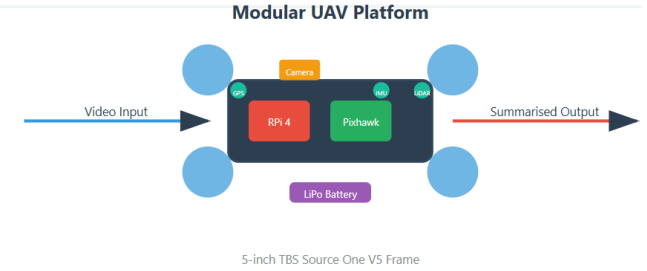

Figure 3: Custom hardware architecture featuring Raspberry Pi 4 for edge computing, Pixhawk flight controller, and integrated sensors for autonomous operation.

3.3 Video Summarisation Pipeline

The UAV runs a software pipeline that combines histogram-based keyframe extraction with YOLOv5n, a lightweight model for detecting people. The aim is to reduce the number of frames sent so that the system uses less power during search and rescue tasks. An overview of the workflow is shown in Figure 4. The approach merges computer vision and video filtering: it first compares frames using histograms to remove similar ones, then uses object detection to check for the presence of people. Only keyframes containing relevant information are kept and sent. This makes the system well-suited for edge deployment on the Raspberry Pi.

Testing was carried out on both a Raspberry Pi 4 and a MacBook M1 to measure accuracy and performance. The final deployment was optimised for the Raspberry Pi.

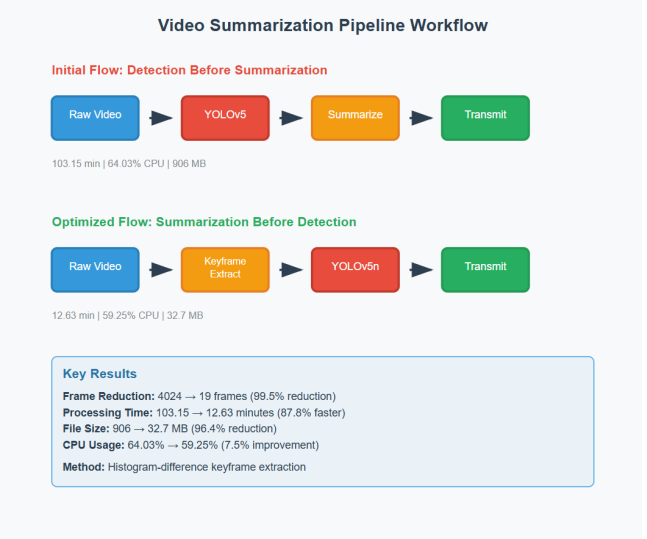

Figure 4: Video summarisation pipeline workflow. Raw frames are filtered via histogram comparison to extract keyframes; a lightweight detector processes the keyframes, after which detected frames are rebuilt and transmitted.

3.3.1 Conventional Flow: detection before summarisation. An early test used a 240 MB HD video containing 4024 frames. The first version of the pipeline ran YOLOv5 on every frame to detect people. After detection, the frames were passed into a script that rebuilt the video into a compressed MP4 format. No filtering or frame reduction was done at this stage.

On the Raspberry Pi 4, this process used 64.03 % of the CPU and took 103.15 minutes. The same task on a MacBook M1 used 23.03 % CPU and took 21.72 minutes. These results showed that running YOLOv5 on every frame without optimisation placed a heavy load on the Raspberry Pi, which raised concerns about its use in real-time UAV applications [18].

Table 1: Conventional detection performance

Metric	Raspberry Pi	MacBook M1
Model	YOLOv5	YOLOv5
CPU (%)	64.03	23.03
Time (min)	103.15	21.72
Size	906 MB	906 MB

3.3.2 Proposed Flow: summarisation before detection and transmission. The improved pipeline first reduces the number of frames. Histogram difference was used, bringing the video from 4024 frames down to just 19. Two other methods (structural similarity index and mean squared error) were tried; they were slower and often selected frames with minimal change, so they were discarded. Histogram difference gave better results in less time and selected frames that better reflected the video content.

Keyframe extraction. This step is implemented in the `extract_keyframes()` method. Histogram comparison uses parameters such as `hist_bins` and `hist_threshold`, and frames are downsampled via `downsample_width` and `downsample_height` to improve speed. Keyframes are saved only when there is a clear difference from the previous frame. If two frames look too similar, the second is skipped, reducing duplicates and keeping the output short.

Once the keyframes are selected, YOLOv5n is used instead of YOLOv5. The smaller model runs better on the Raspberry Pi without overloading the system. Each of the 19 keyframes passes through the model, but not all are kept: the pipeline saves only the frames where a person is detected; others are dropped, so the final video includes only meaningful content.

Human detection. People are detected using the `detect()` method, which loads the `yolov5n.pt` model. It is configured to focus only on people using `–classes 0`. Detections are saved with confidence values. If no person is found in a frame, that frame is dropped. While this prototype filters for people, the pipeline is class-agnostic. The class list can be extended (e.g., vehicles, backpacks, dogs, debris) by updating the detector configuration to include the corresponding class IDs and, where needed, fine-tuning a lightweight model on SAR-relevant examples. To preserve performance on the Raspberry Pi, detection remains keyframe-only and can apply class-specific confidence thresholds. When higher accuracy or more classes are

required, summarised keyframes can be re-processed on a ground station with a richer model.

Video rebuilding and transmission. A simple script rebuilds the final video: only the frames where a person has been detected are included. The output is a compressed MP4 around 32.7 MB. The end-to-end average CPU over the full 12.63-minute run was approximately 59 % (combined 0–400 % scale). This is a major improvement over the earlier version, which took more than 103 minutes and used over 64 % CPU while processing every frame. The final setup is much more practical for onboard use on the UAV, since detection is applied only to selected keyframes, reducing compute and transmission overhead.

Table 2: Keyframe pipeline performance: full pipeline

Method	Frames	Time (min)	CPU (%)	Size
YOLOv5 (full video)	4024	103.15	64.03	906 MB
Keyframe extraction (final)	19	12.63	59.25	32.7 MB

Note: The reported 12.63 minutes is the end-to-end batch runtime on the Raspberry Pi 4 to process the full 4024-frame clip after reduction to 19 keyframes; it is not per-frame online latency. In deployment, detection runs only on selected keyframes, and input downscaling further reduces per-frame latency.

Detection was performed using YOLOv5 and YOLOv5n; YOLOv5n was preferred on the Raspberry Pi due to its reduced overhead and faster inference time. Transmission was simulated (copy-to-folder plus log) rather than executed over a live radio link; the send time was therefore estimated using the formula below:

$$\text{Transmission Time (sec)} = \frac{\text{File Size (MB)}}{\text{Upload Speed (Mbps)}}. \tag{1}$$

With a 100 Mbps upload speed, a 33 MB file transmits in 0.33 seconds. To check that no important detections were lost during summarisation, a DeepSort tracker [29] was used. It assigned IDs to all detected people across the full 4024-frame video. These IDs were then compared with those found in the 19-frame summarised version. The match confirmed that key human appearances were still present, showing that the frame-selection logic is reliable.

The pipeline runs efficiently on the Raspberry Pi when using YOLOv5n. For longer flights or to reduce power use even more, a better option would be to transmit only the summarised keyframes; detection and video rebuilding could then be done later at a ground station, if bandwidth allows. This approach would shift more of the load off the UAV while keeping important data intact.

3.4 Power Modelling and IoT Integration

Power consumption is critical for UAVs with onboard processing. This subsection details the energy modelling for the pipeline, focusing on CPU usage and battery impact. Benchmarks indicate the Raspberry Pi 4 consumes 2.7 W idle and 6.4 W at peak (all four cores at 100 %) [6], yielding 1.6 W per core:

$$\frac{6.4\,\text{W}}{4\ \text{cores}} = 1.6\,\text{W/core}. \tag{2}$$

To estimate power usage during processing, a simplified linear model is used, scaling the per-core power by the combined CPU utilisation:

$$\text{Estimated Power (W)} = 1.6\,\text{W} \times \left(\frac{\text{CPU Avg. Usage (\%)}}{100}\right). \tag{3}$$

Here, CPU Avg. Usage is the total CPU average reported by the system (e.g., 117.92 % for all combined cores). Within the 12.63-minute run, during the detection slices only, the combined CPU averaged 117.92 %. For a conservative upper-bound on energy, this average is applied across the full 12.63-minute run, giving:

$$\text{Power} = 1.6\,\text{W} \times 1.1792 = 1.89\,\text{W}. \tag{4}$$

$$\text{Energy Used} = 1.89\,\text{W} \times \left(\frac{12.63\,\text{min}}{60}\right) = 0.40\,\text{Wh}. \tag{5}$$

During full-video detection (YOLOv5), the CPU average was 64.03 %, with 103.15 minutes of run-time:

$$\text{Power} = 1.6\,\text{W} \times 0.6403 \approx 1.02\,\text{W}. \tag{6}$$

$$\text{Energy Used} = 1.02\,\text{W} \times \frac{103.15\,\text{min}}{60} \approx 1.75\,\text{Wh}. \tag{7}$$

A built-in LiPo battery pack of 6000 mAh and 14.8 V supplies a total energy capacity of 88.8 Wh:

$$\text{Battery Capacity} = 6.0\,\text{Ah} \times 14.8\,\text{V} = 88.8\,\text{Wh}. \tag{8}$$

The percentage of drain per method is then estimated as:

$$\text{Full detection:} \quad \frac{1.75\,\text{Wh}}{88.8\,\text{Wh}} \times 100 \approx 1.97\,\%. \tag{9}$$

$$\text{Final method:} \quad \frac{0.40\,\text{Wh}}{88.8\,\text{Wh}} \times 100 \approx 0.45\,\%. \tag{10}$$

This represents more than a four-fold reduction in energy use for the final method compared to full detection, allowing significantly more operations per charge cycle [11].

Table 3: Power modelling and battery usage: detection-only comparison

Metric	Raw Video	Filtered Video
CPU (%)	64.03	117.92
Time (min)	103.15	12.63
Power (W)	1.02	1.89
Energy (Wh)	1.75	0.40
Battery (%)	1.97	0.45

Although the instantaneous power of the final pipeline is higher (1.89 W vs 1.02 W), its much shorter runtime (12.63 min vs 103.15 min) yields lower total energy (0.40 Wh vs 1.75 Wh). Final-pipeline CPU is the detection-slice average; for a conservative energy bound it is applied across the full 12.63 minutes. The end-to-end average for the final pipeline is approximately 59 %.

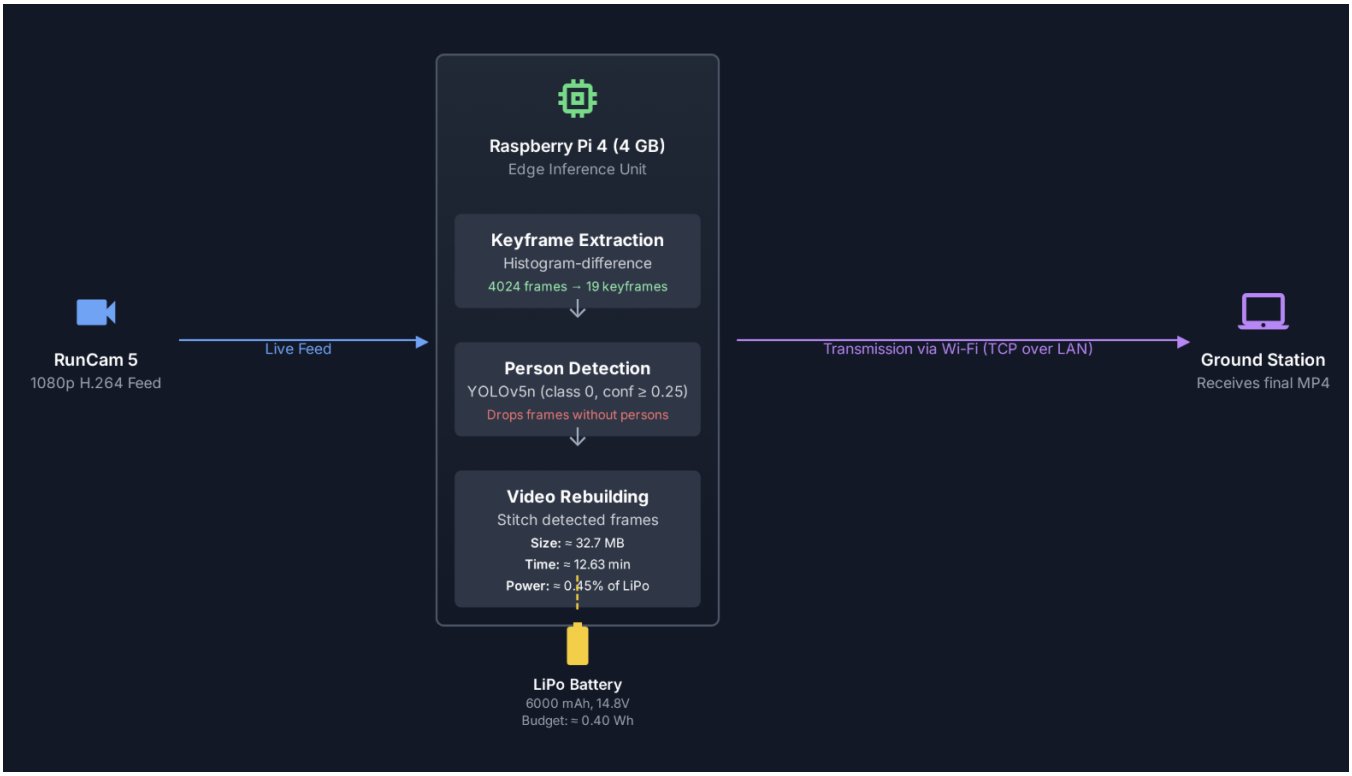

Figure 5: System architecture. Modules for keyframe extraction, lightweight detection, and transmission run on Raspberry Pi; flight control and sensors interface via Pixhawk and peripherals.

4 Discussion

The system employs the YOLOv5n model to prioritise fast inference on the Raspberry Pi, trading some detection accuracy for efficiency due to its low parameter count [28]. Larger YOLO models are impractical for real-time processing on resource-constrained UAVs due to power and computational limits. The Raspberry Pi 4 proves compact and energy-efficient, enabling onboard AI for small UAVs [22]. However, sustained high CPU loads can trigger thermal throttling, reducing processing speed and throughput [2]. In flight, the airframe provides continuous airflow over the avionics (prop wash across the electronics bay), reducing Raspberry Pi temperatures compared with static bench runs. In addition, detection is invoked only on selected keyframes and inputs are downscaled before inference, which lowers sustained CPU load and reduces the likelihood of throttling during operation.

Content-aware keyframe selection using histogram difference significantly enhances video summarisation compared to fixed-interval sampling, which often captures redundant frames and is less effective for dynamic scenes [5]. By identifying frames with significant scene changes, the histogram-based method reduces the video from thousands to tens of frames, preserving critical information with minimal loss. This approach, while requiring additional computation for histogram analysis, justifies the cost by drastically reducing data for detection and transmission, enabling onboard summarisation on the Raspberry Pi under constrained compute and power. As additional classes are enabled, the pipeline retains efficiency by running detection only on selected keyframes and, if required, offloading multi-class re-analysis of those keyframes to the ground.

The prototype revealed practical challenges. The payload, including the Raspberry Pi 4, RunCam 5 camera, GPS, and battery, nears the UAV's lift limit, which can reduce flight stability and endurance on small airframes. The bulky battery mount, secured with Velcro straps, further impacts agility. Thermal management is a concern, as the Raspberry Pi reaches approximately 80 °C under heavy workloads, triggering throttling that lowers frame rates [2]. The pipeline's manual initiation is impractical for SAR missions; automating summarisation at take-off or waypoints would enhance usability. The energy-aware design, leveraging keyframe pre-filtering and selective transmission, achieves over fourfold reductions in runtime and energy use compared to full-frame processing, significantly extending mission duration [22]. Evaluation used pre-recorded SAR-like footage processed on-device; uplink was simulated. Timings are end-to-end batch for the full clip after summarisation. These findings are complementary: when motors are off, off-board streaming may consume energy similar to on-device compute, but summarisation-first pipelines reduce the overall work (and uplink), yielding lower end-to-end energy.

5 Conclusion and Future Work

This work presents an energy-efficient UAV system for SAR that integrates a Raspberry Pi 4 for onboard video summarisation. By

combining histogram-based keyframe extraction with YOLOv5n object detection, the system transmits only critical frames, reducing data rate and power consumption by over fourfold compared to full-video processing [22]. Bench and ground evaluations indicate that the approach maintains high detection accuracy while supporting extended mission durations and timely decision-making in SAR scenarios.

Future work includes automating the pipeline to initiate summarisation and detection at take-off or GPS-defined areas, eliminating manual intervention. A hybrid architecture could stream summarised videos to ground stations for advanced analysis when bandwidth permits. Upgrading to a more powerful onboard computer, such as an NVIDIA Jetson Xavier, would support larger models or higher-resolution processing. Deploying a UAV swarm could enhance coverage and robustness in large-scale SAR missions. Additionally, integrating advanced analytics, such as emotion or gesture recognition, into summarised frames could provide richer insights for rescue teams. These enhancements aim to improve autonomy, responsiveness, and effectiveness in real-world SAR deployments.

Acknowledgments

The authors would like to thank collaborators for their support and feedback during the development of this work.

References

[1] Namat Bachir and Qurban Memon. 2022. Investigating YOLOv5 for Search and Rescue Operations Involving UAVs. In *Proceedings of the 5th International Conference on Control and Computer Vision (ICCCV)*. 200–204. doi:10.1145/3561613.3561644

[2] Théo Benoit-Cattin, Delia Velasco-Montero, and Jorge Fernández-Berni. 2020. Impact of Thermal Throttling on Long-Term Visual Inference in a CPU-Based Edge Device. *Electronics* 9, 12 (2020), 2106. doi:10.3390/electronics9122106

[3] Sindhu Boddu and Arindam Mukherjee. 2025. YOLOv5-Based Object Detection for Emergency Response in Aerial Imagery. arXiv:2412.05394 [cs.CV] https://arxiv.org/abs/2412.05394 Accessed: 2025-07-05.

[4] Osama M. Bushnaq, Debashisha Mishra, Enrico Natalizio, and Ian F. Akyildiz. 2022. Chapter 9 - Unmanned Aerial Vehicles (UAVs) for Disaster Management. In *Nanotechnology-Based Smart Remote Sensing Networks for Disaster Prevention*, Adil Denizli, Marcelo S. Alencar, Tuan Anh Nguyen, and David E. Motaung (Eds.). Elsevier, 159–188. doi:10.1016/B978-0-323-91166-5.00013-6

[5] Yujie Ding, Danning Shen, Liang Ye, and Wenhao Zhu. 2022. A Keyframe Extraction Method Based on Transition Detection and Image Entropy. In *Proceedings of the 2022 7th International Conference on Communication, Image and Signal Processing (CCISP)*. 260–264. doi:10.1109/CCISP55629.2022.9974364

[6] Jeff Geerling. 2019. Power Consumption Benchmarks. https://www.pidramble.com/wiki/benchmarks/power-consumption. Raspberry Pi 4 B: Idle 540 mA (2.7 W); 400% CPU load 1280 mA (6.4 W).

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* (2017). https://arxiv.org/abs/1704.04861

[8] Tanveer Hussain, Khan Muhammad, Amin Ullah, Javier Del Ser, Amir H. Gandomi, Muhammad Sajjad, Sung Wook Baik, and Victor Hugo C. de Albuquerque. 2021. Multiview Summarization and Activity Recognition Meet Edge Computing in IoT Environments. *IEEE Internet of Things Journal* 8, 12 (2021), 9634–9644. doi:10.1109/JIOT.2020.3027483

[9] Palash Yuvraj Ingle, Yujun Kim, and Young-Gab Kim. 2022. DVS: A Drone Video Synopsis towards Storing and Analyzing Drone Surveillance Data in Smart Cities. *Systems* 10, 5 (2022). doi:10.3390/systems10050170

[10] Glenn Jocher, Ayush Chaurasia, Jirka Borovec, et al. 2020. YOLOv5 by Ultralytics. https://github.com/ultralytics/yolov5. Accessed: 2025-07-03.

[11] Sarah Jumah, Ahmed Elezab, Omar Zayed, Ryan Ahmed, Mehdi Narimani, and Ali Emadi. 2022. State of Charge Estimation for EV Batteries Using Support Vector Regression. In *Proceedings of the 2022 IEEE Transportation Electrification Conference & Expo (ITEC)*. 964–969. doi:10.1109/ITEC53557.2022.9813811

[12] Shalini K, Abhishek Kumar Srivastava, Surendra Allam, and Dilip Lilaramani. 2021. Comparative Analysis on Deep Convolution Neural Network Models Using PyTorch and OpenCV DNN Frameworks for Identifying Optimum Fruit Detection Solution on RISC-V Architecture. In *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*. 738–743. doi:10.1109/MysuruCon52639.2021.9641594

[13] Rahima Khanam, Tahreem Asghar, and Muhammad Hussain. 2025. Comparative Performance Evaluation of YOLOv5, YOLOv8, and YOLOv11 for Solar Panel Defect Detection. *Solar* 5, 1 (2025), 6. doi:10.3390/solar5010006

[14] Jayden King, Lily Huang, Di Wu, Yipeng Zhou, and Young Choon Lee. 2020. EdgeSum: Edge-Based Video Summarization with Dash Cams. In *Proceedings of the 2020 IEEE International Conference on Cloud Engineering (IC2E)*. 1–6. doi:10.1109/IC2E48712.2020.00011

[15] Pranav Krishnan, Sahithya Kattamuri, Gayathri R. Prabhu, and Manazhy Rashmi. 2024. Assistive Eye: A Comparative Analysis of YOLO Object Detection Models on Edge Devices. In *Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing (IC3-2024)*. Association for Computing Machinery, New York, NY, USA, 104–108. doi:10.1145/3675888.3676037

[16] S. MahmoudZadeh, A. Yazdani, Y. Kalantari, B. Ciftler, F. Aidarus, and M. O. Al Kadri. 2024. Holistic Review of UAV-Centric Situational Awareness: Applications, Limitations, and Algorithmic Challenges. *Robotics* 13, 8 (2024), 117. doi:10.3390/robotics13080117

[17] Jakub Mandula, Jonas Kühne, Luca Pascarella, and Michele Magno. 2024. Towards Real-Time Fast Unmanned Aerial Vehicle Detection Using Dynamic Vision Sensors. In *2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 1–6. doi:10.1109/I2MTC60896.2024.10561168

[18] Katleho Masita, Ali N. Hasan, Thokozani Shongwe, and Hasan Abu Hilal. 2025. Deep Learning in Defect Detection of Wind Turbine Blades: A Review. *IEEE Access* 13 (2025), 98399–98425. doi:10.1109/ACCESS.2025.3569799

[19] Aleksandar Blagoj Mitrev and Biljana Risteska Stojkoska. 2025. Comparison of On-Board and Off-Board Processing Power Consumption for Drone Camera Images. In *ICT Innovations 2024: TechConvergence*. Communications in Computer and Information Science, Vol. 2436. Springer, Cham, 136–146. doi:10.1007/978-3-031-86162-8_10

[20] Bo Pang, Erik Nijkamp, and Ying Nian Wu. 2020. Deep Learning With TensorFlow: A Review. *Journal of Educational and Behavioral Statistics* 45, 2 (2020), 227–248. doi:10.3102/1076998619872761

[21] Bryan A. Plummer, Matthew Brown, and Svetlana Lazebnik. 2017. Enhancing Video Summarization via Vision-Language Embedding. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1052–1060. doi:10.1109/CVPR.2017.118

[22] Sergio Bemposta Rosende, Sergio Ghisler, Javier Fernández-Andrés, and Javier Sánchez-Soriano. 2023. Implementation of an Edge-Computing Vision System on Reduced-Board Computers Embedded in UAVs for Intelligent Traffic Management. *Drones* 7, 11 (2023), 682. doi:10.3390/drones7110682

[23] Fateme Salehi, Mustafa Ozger, Naaser Neda, and Cicek Cavdar. 2022. Ultra-Reliable Low-Latency Communication for Aerial Vehicles via Multi-Connectivity. In *Proceedings of the 2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. 166–171. doi:10.48550/arXiv.2205.06046

[24] Jhuma Sunuwar and Samarjeet Borah. 2024. A Comparative Analysis on Major Key-Frame Extraction Techniques. *Multimedia Tools and Applications* 83 (2024), 73865–73910. doi:10.1007/s11042-024-18380-z

[25] Jiashun Suo, Xingzhou Zhang, Weisong Shi, and Wei Zhou. 2024. E3-UAV: An Edge-Based Energy-Efficient Object Detection System for Unmanned Aerial Vehicles. *IEEE Internet of Things Journal* (2024). doi:10.1109/JIOT.2023.3301623

[26] K. Thangavel, D. Spiller, R. Sabatini, S. Amici, S. T. Sasidharan, H. Fayek, and P. Marzocca. 2023. Autonomous Satellite Wildfire Detection Using Hyperspectral Imagery and Neural Networks: A Case Study on Australian Wildfire. *Remote Sensing* 15, 3 (2023), 720. doi:10.3390/rs15030720

[27] Hoang Trinh, Jun Li, Sachiko Miyazawa, Juan Moreno, and Sharath Pankanti. 2012. Efficient UAV Video Event Summarization. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2226–2229. doi:10.1109/ICPR.2012.552

[28] Ultralytics. 2024. YOLOv5 vs YOLOv7: Detailed Technical Comparison for Object Detection. https://docs.ultralytics.com/compare/yolov5-vs-yolov7/. Accessed: 2025-07-05.

[29] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple Online and Realtime Tracking with a Deep Association Metric. In *2017 IEEE International Conference on Image Processing (ICIP)* (Beijing, China). IEEE, 3645–3649. doi:10.1109/ICIP.2017.8296962

[30] Yu Xiao, Yong Cui, Petri Savolainen, Matti Siekkinen, An Wang, Liu Yang, Antti Ylä-Jääski, and Sasu Tarkoma. 2014. Modeling Energy Consumption of Data Transmission Over Wi-Fi. *IEEE Transactions on Mobile Computing* 13, 8 (2014), 1760–1773. doi:10.1109/TMC.2013.51