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

1st Chidike Vincent
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, UK
C3002085@hallam.shu.ac.uk

2nd Laurence Hirsch
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, UK
l.hirsch@shu.ac.uk

3rd Najam Ul Hasan
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, UK
N.Ul-Hasan@shu.ac.uk

4th Caren Crizben Ezlin Fernandes
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, UK
Caren.Fernandes@shu.ac.uk

5th Adriana Crainic
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, UK
Adriana.Crainic@student.shu.ac.uk

6th Jonathan Zasada-James
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, UK
J.Zasada-James@shu.ac.uk

7th James Baldwin
School of Computing and Digital Technologies
Sheffield Hallam University
Sheffield, UK
j.baldwin@shu.ac.uk

***Abstract*—Rapid detection of individuals in Search and Rescue (SAR) operations is critical for minimizing 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 summarization 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 optimizes weight and power while supporting real-time edge processing. The results shows 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.**

***Index Terms*—SAR, UAV, IoT, Edge Computing, Video Summarization, Power Consumption, Battery Analytics**

I. Introduction

UAVs have become indispensable in disaster management, offering rapid environmental assessment and critical support in hazardous situations [1]. 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 worth of using UAV technology in mapping disaster zones and identifying infrastructure damage [2]. However, the traditional UAV-based disaster response systems faces significant challenges due to their reliance on continuous transmission of unfiltered, high resolution video ground stations [3]. This approach creates substantial communication bottlenecks, increases decision making latency and accelerates onboard energy depletion, particularly disaster stricken regions with limited connectivity, often leading to distorted video streams or complete transmission failures during critical moments.

To address these limitations, this research proposes an

innovative UAV system that leverages edge computing to perform onboard video summarization for SAR operations. By integrating lightweight computer vision models, such as YOLOv5n, with a Raspberry Pi 4, the system processes video streams in real-time, identifying and prioritising contextually relevant frames—such as 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 [4], [5]. and and optimised neural network architectures, including MobileNet and TensorFlow Lite, enable efficient object detection and classification tailored for emergency response [6], [7].

The proposed framework minimises bandwidth requirements and enhances operational efficiency by filtering non-essential content, ensuring that ground teams receive decision-ready data promptly.

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 Summarization Algorithm: A novel algorithm optimised for disaster response, intelligently prioritising video content based on emergency relevance criteria.
2. Resource-Efficient Edge AI Pipeline: Implementation of a processing pipeline designed for UAV deployment, operating within the constraints of limited computational power and energy availability.
3. Operational Validation: Comprehensive evaluation of the system's performance in realistic SAR scenarios, demonstrating significant reductions in data transmission and energy usage.

The remainder of this paper is organised as follows: Section 2 reviews related work in UAV video analytics and energy-aware UAV systems. Section 3 presents the system design, including both hardware assembly and software pipeline. Section 4 discusses the experimental setup and test results. Section 5 analyses system trade-offs and limitations. Section 6 summarizes the research and outlines strategic developments.

II. Related Work

This section reviews advancements in UAV-based video analytics and edge computing, focusing on energy-efficient solutions for Search and Rescue (SAR) operations in disaster response.

A. Recent Studies

Unmanned Aerial Vehicles (UAVs) are vital for disaster response, enabling rapid aerial scans to detect victims, hazards, and infrastructure damage in areas inaccessible to ground teams [8]. Modern UAVs support autonomous tasks like Simultaneous Localization and Mapping (SLAM), waypoint navigation, and collision avoidance [2]. However, their 20–30-minute flight times are limited by Size, Weight, and Power (SWAP) constraints [4]. Continuous video streaming reduces endurance, necessitating energy-efficient processing [9].

Suo et al.'s E-UAV system optimizes energy by adjusting flight parameters and algorithm settings, achieving savings in detecting people and vehicles without compromising accuracy [4].

Lightweight Convolutional Neural Networks (CNNs), particularly YOLOv5, are widely used for UAV-based object detection due to their speed-accuracy balance [9]. The YOLOv5n (nano) variant, with 1.9 million parameters, is suited for resource-constrained UAVs [10]. Boddu and Mukherjee [9] demonstrated YOLOv5's effectiveness in emergency tasks, such as detecting ambulances or fires. Mandula et al. [10] implemented YOLOv5n on an embedded Jetson platform with TensorRT optimizations for real-time inference with minimal power use.

YOLOv5n has been applied in wind turbine inspections to detect faults efficiently [11].

Continuous video streaming consumes substantial energy, prompting research into onboard video summarization. Zhang et al. [12] developed a system that transmits keyframes of detected objects, achieving over 1000x 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 [13].

B. 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 [14], [15], [16]. 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 summarization, enabling UAVs to identify and transmit only the most relevant frames. Techniques such as histogram differences or visual motion analysis select representative frames, minimizing data transmission [17]. 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 [18]. Lightweight alternatives are thus preferred for onboard processing.

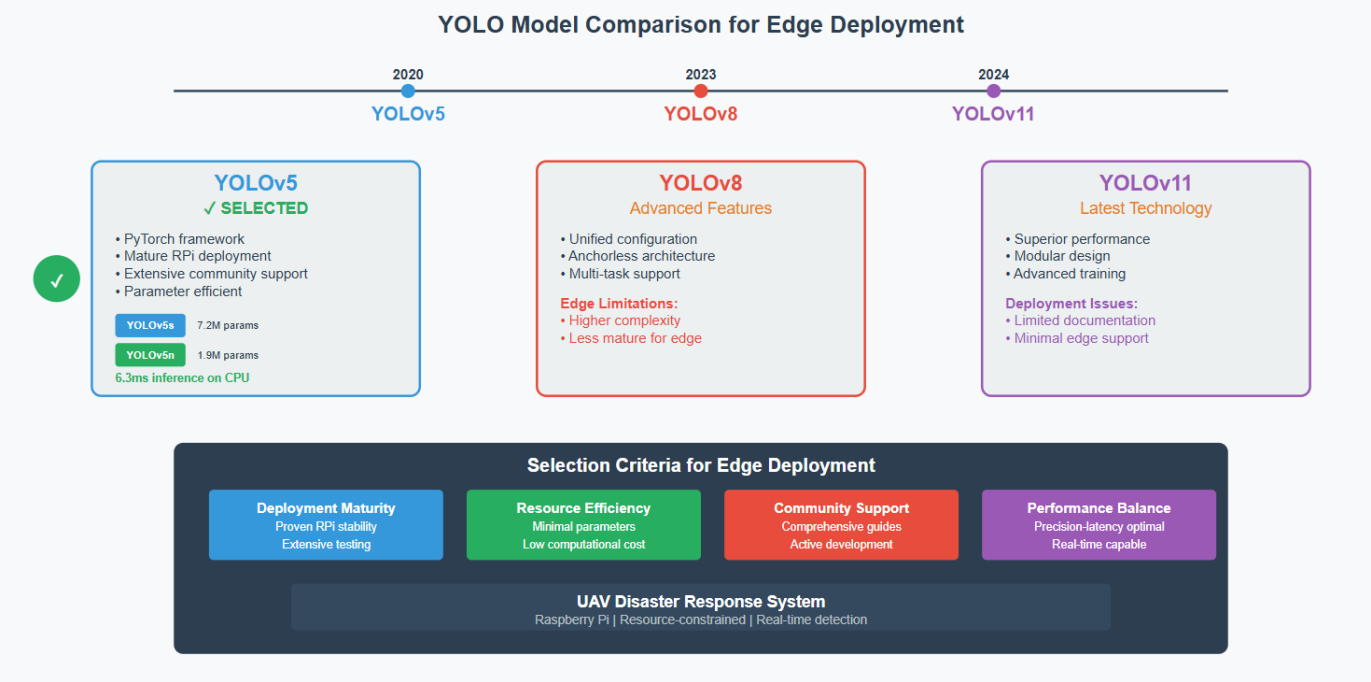


Fig. 1. YOLO Model Comparison for Edge Deployment

The YOLOv5n model, with only 1.9 million parameters, is highly optimized for resource-constrained platforms like the Raspberry Pi, delivering a robust balance of detection accuracy and computational efficiency [19]. Compared to newer models like 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 krishnan2024assistive, [20], [21]. For instance, Mastia et al. [13] utilized 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 summarization pipeline tailored for SAR. This approach optimizes onboard processing, minimizes data transmission, and extends UAV mission duration, enhancing real-time decision-making in disaster response scenarios.

III. System Design & Implementation

This section outlines the complete design and development of a custom-built UAV system designed for energy-efficient video summarization 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.

A. Hardware Architecture

A custom UAV was built using easily 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, doesn't consume much power, and is powerful enough to handle real-time computer vision.

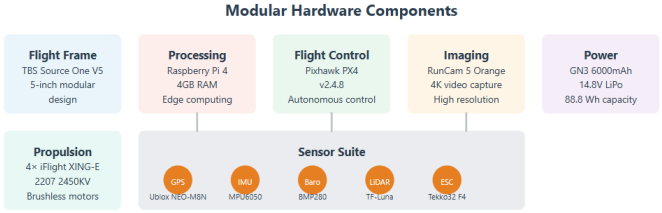


Fig. 2. Detailed breakdown of all hardware components including flight frame, processing unit, sensors, and power management systems with their specifications

The UAV uses a 5-inch TBS Source One V5 frame. This frame was picked because it provides good balance and makes it easier to mount different parts. Four iFlight XING-E Pro2207 2450KV motors were installed, each paired with 5×4.3×3 three-blade propellers. Together, they give enough lift to support the UAV's full weight, including the Raspberry Pi and sensors. Power is supplied by a 6000mAh 14.8V LiPo battery, which connects to a HolyBro power distribution board. Since the connectors did not match (XT90 on the battery and XT60 on the board), an adapter was used to join them. A Pixhawk PX4 2.4.8 controller is used for flight control, with a Tekko32 F4 4-in-1 50A ESC managing the motor signals.

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.

These were linked through I2C, UART, or GPIO, depending on what each required. When two devices needed the same port or pins were limited, small adapters and simple custom cables were used to make things work. Parts were fixed in place using standoffs, foam tape, and zip ties to prevent shaking. The battery sits underneath the drone and is held with Velcro, making it easy to remove and charge. Heat is handled through airflow and space between components; no fan was needed.

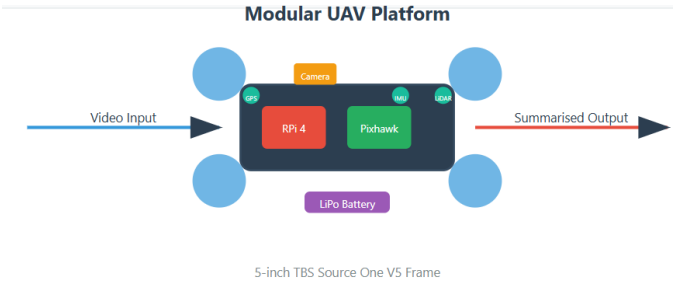


Fig. 3. Custom hardware architecture featuring Raspberry Pi 4 for edge computing, Pixhawk flight controller, and integrated sensors for autonomous operation

B. Video Summarisation Pipeline

The UAV runs a software pipeline that combines histogram-based keyframe extraction with YOLOv5n, which is a lightweight model for detecting people. The aim is to reduce how many frames are sent, so the system uses less power during search and rescue tasks.

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 key frames 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.

1) *Initial Flow:* Detection Before Summarization

An early test was carried out using a 240MB 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.[11].

TABLE I
INITIAL DETECTION PIPELINE 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

2) *Optimized Flow:* Summarization Before Detection and Transmission

The first step in the improved pipeline was to reduce the number of frames. Histogram-difference was used for this, bringing it down from 4024 to just 19. Two other methods, SSIM and MSE, were tried as well. They were slower and often picked frames that didn't show much change, so they were dropped. Histogram-difference gave better results in less time and picked frames that reflected the video more accurately.

Keyframe Extraction

This step was handled in the `extract_keyframes()` method. It used histogram comparison with parameters like `hist_bins` and `hist_threshold`, and frames were downsampled using `downsample_width` and `downsample_height` to improve speed. Keyframes were only saved when there was a clear difference between them and the ones before. If two frames looked too similar, the second one was skipped. This helped cut down on duplicates and kept the output short.

Once the keyframes were selected, YOLOv5n was used instead of YOLOv5. This smaller version was chosen because it runs better on the Raspberry Pi without overloading the system. Each of the 19 keyframes went through the model, but not all of them were kept. The pipeline was designed to save only the frames where a person was detected. The rest were dropped at this stage, so the final video only included meaningful content.

Human Detection

People were detected using the `detect()` method, which loaded the `yolov5n.pt` model. It was set up to focus only on people using `--classes 0`. Detections were saved with confidence values. If no person was found in a frame, that frame was dropped.

Video Rebuilding and Transmission

A simple script was used to rebuild the final video. Only the frames where a person had been detected were included. Nothing else. The output came out as a compressed MP4,

around 32.7 MB in size. Running everything on the Raspberry Pi took 12.63 minutes, and CPU usage stayed close to 59.25% throughout. This was 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 was much more practical for real-time use on the UAV, especially in time-critical search and rescue work.

The video was put back together using the `rebuild_video()` script. It used frames that were already processed by YOLOv5n and stored in the `runs/detect` folder. Only frames where people were found were added. For transmission, the `transmit_video()` script saved a copy to another folder and kept a small log to show if the process worked or failed.

TABLE II
KEYFRAME PIPELINE PERFORMANCE

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

Detection was performed using YOLOv5 and YOLOv5n; YOLOv5n was preferred on the Raspberry Pi due to its reduced overhead and faster inference time. The transmission was not fully automated. It was just simulated for testing purposes. The time it would take to send the file can be worked out using this formula:

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

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[22] 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 summarized version. The match confirmed that key human appearances were still present after summarization, which showed that the logic behind frame selection was working reliably.

The pipeline runs efficiently on the Raspberry Pi when using YOLOv5n. But for longer flights or to reduce power use even more, a better option would be to transmit just the summarized 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.

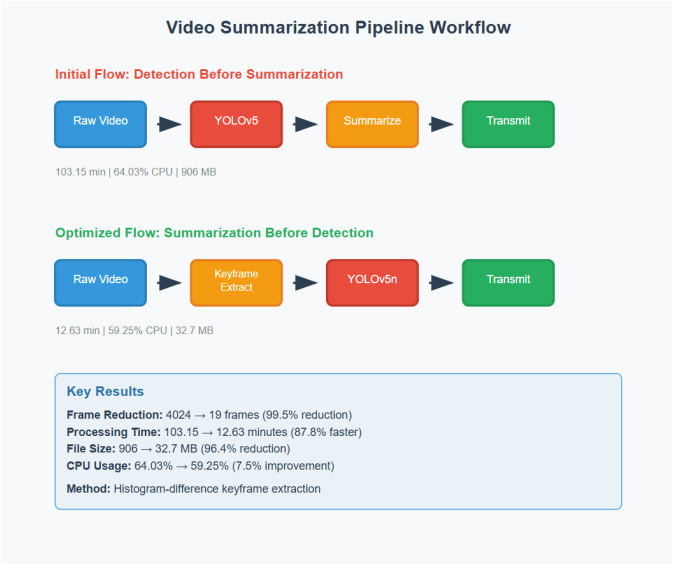


Fig. 4. Video Summarisation Pipeline Workflow

C. Power Modelling & 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 4 cores at 100%) [30], yielding 1.6 W per core.

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

To estimate power usage during processing, the research adopted a simplified approach using CPU utilization as a multiplier.

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

where CPU Avg. Usage is the total CPU average reported by the system (e.g., 117.92% for all combined cores).

During summarized detection (final method), the total CPU average was 117. 92% for 12.63 minutes, giving:

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

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

During full-video detection (YOLOv5), the CPU average was 64. 03%, with 103.15 min of run-time:

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

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

A built-in LiPo battery pack indicated at 6000 mAh and 14.8 V supplied a **total energy capacity of 88.8 Wh)**.

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

The percentage of drain per method was then estimated as:

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

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

This represents over a 4× reduction in energy use for the final method compared to full detection, allowing significantly more operations per charge cycle [23].

Note: The power model presented here is used strictly for comparison between the two processing methods. All CPU usage values were normalized to a 100% scale by dividing total CPU load (e.g., up to 400% for four cores) by the number of cores.The value of 1.6 W per core was used in a straightforward linear estimate to compare both pipelines. A fixed value of 1.6 W per core was used to keep the estimate simple and consistent across both pipelines. The goal was not to capture total system power, but to compare how much energy each method roughly used under the same assumption.

TABLE III
POWER MODELLING & BATTERY USAGE

Metric	Full Detection	Final Pipeline
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

The pipeline integrates with the UAV's hardware, with all scripts running on the Raspberry Pi 4 as the edge inference unit.

The 1080p RunCam 5 footage is processed live, and once started, the entire pipeline runs automatically to carry out keyframe extraction, detection, rebuilding, and transmission.

IV. DISCUSSION

The system employs the YOLOv5n model to prioritize fast inference on the Raspberry Pi, trading some detection accuracy for efficiency due to its low 1.9 million parameter count [24]. This trade-off is justified, as 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 [25]. However, sustained high CPU loads can trigger thermal throttling, reducing processing speed and throughput [26].

Content-aware keyframe selection using histogram-difference significantly enhances video summarization compared to fixed-interval sampling, which often captures redundant frames and is less effective for dynamic scenes [27]. 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 near-real-time summarization on the Raspberry Pi.

The prototype revealed practical challenges. The payload, including the Raspberry Pi 4, RunCam 5 camera, GPS, and battery, nears the UAV's lift limit, slightly reducing flight stability and endurance due to increased power demands. The bulky battery mount, secured with Velcro straps, further impacts agility. Thermal management is a concern, as the Raspberry Pi reaches 80°C under heavy workloads, triggering throttling that lowers frame rates [26]. The pipeline's manual initiation is impractical for SAR missions; automating summarization at takeoff 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 [25]. These optimizations enable near-real-time SAR operations, balancing efficiency and effectiveness.

V. CONCLUSION AND FUTURE WORK

This work presents an energy-efficient UAV system for SAR integrating a Raspberry Pi 4 for onboard video summarization. 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 [25]. Field tests demonstrate that this approach maintains high detection

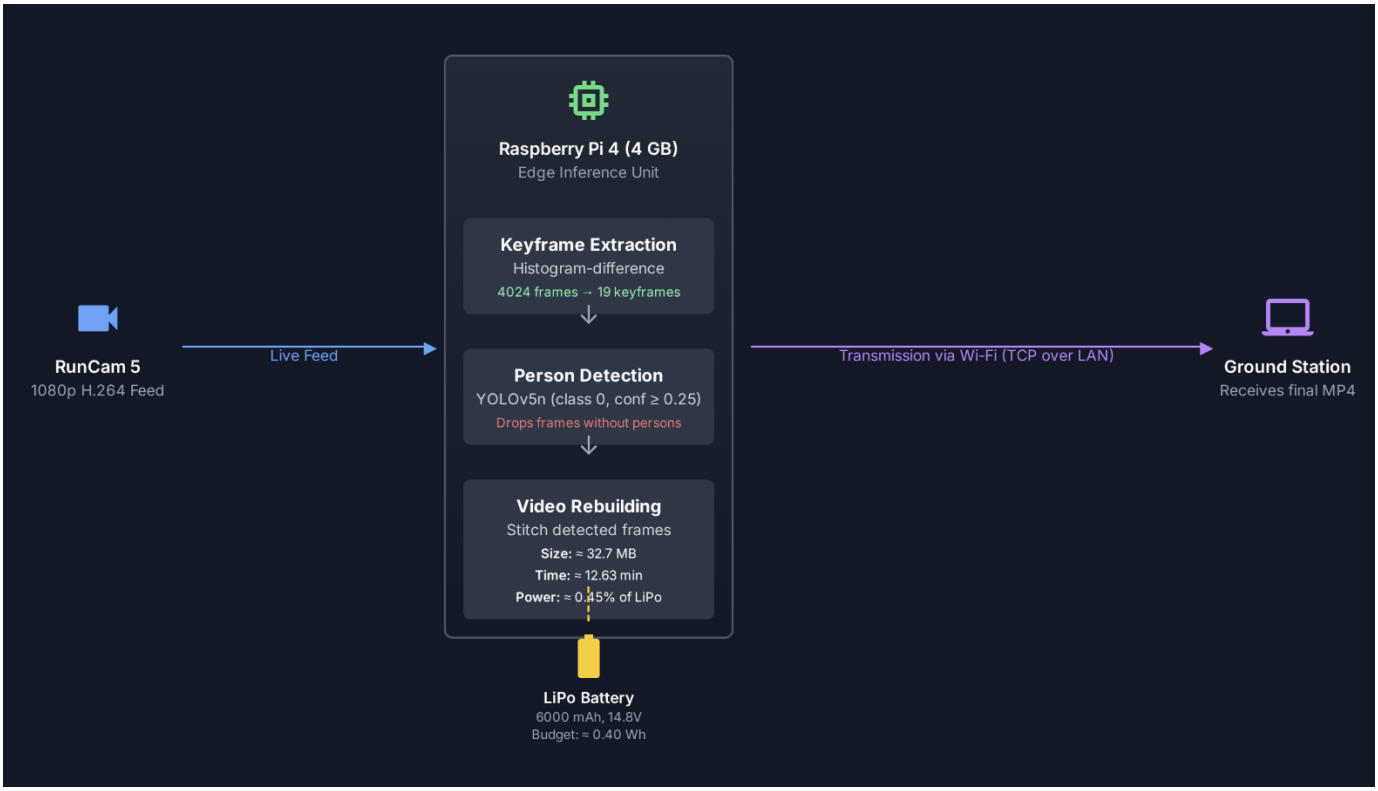


Fig. 5. System Architecture

accuracy while enabling extended mission durations and rapid decision-making for SAR operations.

Future work includes automating the pipeline to initiate summarization and detection at takeoff or GPS-defined areas, eliminating manual intervention. A hybrid architecture could stream summarized 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 summarized frames could provide richer insights for rescue teams. These enhancements aim to improve autonomy, responsiveness, and effectiveness in real-world SAR deployments.

Acknowledgment

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

References

[1] O. M. Bushnaq, D. Mishra, E. Natalizio, and I. F. Akyildiz, "Chapter 9 - unmanned aerial vehicles (uavs) for disaster management," in *Nanotechnology-Based Smart Remote Sensing Networks for Disaster Prevention*, ser. Micro and Nano Technologies, A. Denizli, M. S. Alencar, T. A. Nguyen, and D. E. Motaung, Eds. Elsevier, 2022, pp. 159–188. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323911665000136

[2] F. Salehi, M. Ozger, N. Neda, and C. Cavdar, "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)*, 2022, pp. 166–171. [Online]. Available: https://doi.org/10.48550/arXiv.2205.06046

[3] S. MahmoudZadeh, A. Yazdani, Y. Kalantari, B. Ciftler, F. Aidarus, and M. O. Al Kadri, "Holistic review of uav-centric situational awareness: Applications, limitations, and algorithmic challenges," *Robotics*, vol. 13, no. 8, p. 117, 2024. [Online]. Available: https://www.mdpi.com/2218-6581/13/8/117

[4] J. Suo, X. Zhang, W. Shi, and W. Zhou, "E3-uav: An edge-based energy-efficient object detection system for unmanned aerial vehicles," *IEEE Internet of Things Journal*, 2024. [Online]. Available: https://doi.org/10.1109/JIOT.2023.3301623

[5] K. Thangavel, D. Spiller, R. Sabatini, S. Amici, S. T. Sasidharan, H. Fayek, and P. Marzocca, "Autonomous satellite wildfire detection using hyperspectral imagery and neural networks: A case study on australian wildfire," *Remote Sensing*, vol. 15, no. 3, p. 720, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/15/3/720

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[7] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, 2020. [Online]. Available: https://doi.org/10.3102/1076998619872761

[8] N. Bachir and Q. Memon, "Investigating yolov5 for search and rescue operations involving uavs," in *Proceedings of the 5th International Conference on Control and Computer Vision (ICCCV)*, 2022, pp. 200–204. [Online]. Available: https://doi.org/10.1145/3561613.3561644

[9] S. Boddu and A. Mukherjee, "Yolov5-based object detection for emergency response in aerial imagery," 2025, accessed: 2025-07-05.

[Online]. Available: https://arxiv.org/abs/2412.05394

[10] J. Mandula, J. Kühne, L. Pascarella, and M. Magno, "Towards real-time fast unmanned aerial vehicle detection using dynamic vision sensors," in *2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2024, pp. 1–6. [Online]. Available: https://doi.org/10.1109/I2MTC60896.2024.10561168

[11] K. Masita, A. N. Hasan, T. Shongwe, and H. A. Hilal, "Deep learning in defect detection of wind turbine blades: A review," *IEEE Access*, vol. 13, pp. 98 399–98 425, 2025. [Online]. Available: https://doi.org/10.1109/ACCESS.2025.3569799

[12] H. Trinh, J. Li, S. Miyazawa, J. Moreno, and S. Pankanti, "Efficient uav video event summarization," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*. Los Alamitos, CA, USA: IEEE Computer Society, November 2012, pp. 2226–2229. [Online]. Available: https://doi.org/10.1109/ICPR.2012.552

[13] P. Y. Ingle, Y. Kim, and Y.-G. Kim, "Dvs: A drone video synopsis towards storing and analyzing drone surveillance data in smart cities," *Systems*, vol. 10, no. 5, 2022. [Online]. Available: https://www.mdpi.com/2079-8954/10/5/170

[14] T. Hussain, K. Muhammad, A. Ullah, J. D. Ser, A. H. Gandomi, M. Sajjad, S. W. Baik, and V. H. C. de Albuquerque, "Multiview summarization and activity recognition meet edge computing in iot environments," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9634–9644, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9208765

[15] Y. Xiao, Y. Cui, P. Savolainen, M. Siekkinen, A. Wang, L. Yang, A. Ylä-Jääski, and S. Tarkoma, "Modeling energy consumption of data transmission over wi-fi," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1760–1773, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6506081

[16] J. King, L. Huang, D. Wu, Y. Zhou, and Y. C. Lee, "Edgesum: Edge-based video summarization with dash cams," in *Proceedings of the 2020 IEEE International Conference on Cloud Engineering (IC2E)*, 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9096491

[17] J. Sunuwar and S. Borah, "A comparative analysis on major key-frame extraction techniques," *Multimedia Tools and Applications*, vol. 83, pp. 73 865–73 910, 2024. [Online]. Available: https://doi.org/10.1007/s11042-024-18380-z

[18] B. A. Plummer, M. Brown, and S. Lazebnik, "Enhancing video summarization via vision-language embedding," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1052–1060. [Online]. Available: https://ieeexplore.ieee.org/document/8099601

[19] G. Jocher, A. Chaurasia, J. Borovec *et al.*, "YOLOv5 by ultralytics," https://github.com/ultralytics/yolov5, 2020, accessed: 2025-07-03.

[20] S. K, A. K. Srivastava, S. Allam, and D. Lilaramani, "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)*, 2021, pp. 738–743. [Online]. Available: https://ieeexplore.ieee.org/document/9641594

[21] R. Khanam, T. Asghar, and M. Hussain, "Comparative performance evaluation of yolov5, yolov8, and yolov11 for solar panel defect detection," *Solar*, vol. 5, no. 1, p. 6, 2025. [Online]. Available: https://doi.org/10.3390/solar5010006

[22] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649. [Online]. Available: https://doi.org/10.1109/ICIP.2017.8296962

[23] S. Jumah, A. Elezab, O. Zayed, R. Ahmed, M. Narimani, and A. Emadi, "State of charge estimation for ev batteries using support vector regression," in *Proceedings of the 2022 IEEE Transportation Electrification Conference & Expo (ITEC)*, 2022, pp. 964–969. [Online]. Available: https://ieeexplore.ieee.org/document/9813811

[24] Ultralytics, "Yolov5 vs yolov7: Detailed technical comparison for object detection," https://docs.ultralytics.com/compare/yolov5-vs-yolov7/, 2024, accessed: 2025-07-05.

[25] S. B. Rosende, S. Ghisler, J. Fernández-Andrés, and J. Sánchez-Soriano, "Implementation of an edge-computing vision system on reduced-board computers embedded in uavs for intelligent traffic management," *Drones*, vol. 7, no. 11, p. 682, 2023. [Online]. Available: https://doi.org/10.3390/drones7110682

[26] T. Benoit-Cattin, D. Velasco-Montero, and J. Fernández-Berni, "Impact of thermal throttling on long-term visual inference in a cpu-based edge device," *Electronics*, vol. 9, no. 12, p. 2106, 2020. [Online]. Available: https://doi.org/10.3390/electronics9122106

[27] Y. Ding, D. Shen, L. Ye, and W. Zhu, "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)*, 2022, pp. 260–264. [Online]. Available: https://ieeexplore.ieee.org/document/9974364