International Journal of Safety and Security Engineering
Vol. 14, No. 2, April, 2024, pp. 387-397
Journal homepage: http://iieta.org/journals/ijsse

Security Surveillance Using UAVs and Embedded Systems in Industrial Areas

Abdelkader Mezouari[1*], Mohamed Benaly[1], Hajar El Karch[2], Hamad Dahou[1], Laâmari Hlou[1], Rachid Elgouri[2]

[1] Laboratory of Electronic Systems, Information Processing, Mechanics and Energetics, Faculty of Sciences, Ibn Tofail University, Kenitra 14000, Morocco
[2] Laboratory of Advanced Systems Engineering, National School of Applied Sciences, Ibn Tofail University, Kenitra 14000, Morocco

Corresponding Author Email: abdelkader.mezouari1@uit.ac.ma

https://doi.org/10.18280/ijsse.140207

Received: 25 May 2023
Revised: 2 December 2023
Accepted: 11 December 2023
Available online: 26 April 2024

Keywords:
computer vision, embedded systems, industrial safety, motion opposition, object detection, unmanned aerial vehicle (UAV), 360° detection, security

ABSTRACT

This paper proposes a novel security surveillance Unmanned Aerial Vehicle (UAV) that can handle security in large industrial areas with increased surveillance efficiency. Our basic idea is that Unmanned Aerial Vehicle sighting can be treated as a motion detection problem in the surveillance area by detecting position and type simultaneously when the Unmanned Aerial Vehicle flies in the 360° detection area. To reach our target, this paper proposes a mathematical approach based on camera calibration, ordinal distortion correction, and three-dimensional reconstruction that can help us determine the exact position of a moving object in the monitored area. It is also important to recognize movements and their character and to determine their position on the ground, all of this must be done in Real-time with short processing times. The outcomes of our study demonstrate that system processing average duration and processing system consumption have slightly decreased with the utilization of the Raspberry Pi+VPU system compared to alternatives such as the Jetson Nano, Raspberry Pi 4 boards, clusters, and personal computers. This underscores the effectiveness of our proposed system in terms of processing efficiency and resource utilization.

1. INTRODUCTION

In the last decade, drones have been widely changing and their usage has steadily increased with each passing year in many application domains such as navigation, military, delivery, topography, etc. In various fields, Drones have exceeded expectations in performance, accuracy, and the tasks assigned to them. Therefore, based on our approach, we can exploit the drone for surveillance of our areas with a determinant algorithm. For surveillance purposes to provide safe and secure industrial facilities. Surveillance systems proposed here include secure real-time photography, motion, object detection, camera calibration, and tracking of the object's trajectory using coordinates constructed from the 3D coordinates of a calibrated image inserted by three drone cameras, relying solely on ground-based surveillance measures limits visibility, especially in sprawling industrial complexes with complex layouts. Blind spots and obscured areas can easily become vulnerable points for unauthorized access or security breaches. hence the need to develop advanced surveillance systems attached to UAVs, such as the system proposed in this paper.

Motion and object detection is an important security research and surveillance application area. With the presence of neural network technology, it is being used more and more in various fields of application. Peculiarly in security, motion, and object detection have a role. Several approaches have been suggested in object detection. Lightweight feature-enhanced convolutional neural network methods are used for low altitude and small size [1] to solve the problem of real-time flying and to improve guidance information to suppress black-flying UAVs. A variety of approaches [2-7] to enhance and improve the appearance of objects and increase the precision marge of the algorithm used based on deep neural network method (Faster R-CNN, SSD, YOLO, LSL\-Net....), the difference between these methods is processing architecture. As for the areas of application [8-15], (underwater, robotic arm, tracking.), researchers are limited to developing algorithms to increase quality only in the field in which they are applied, as they sometimes become unsuitable in other areas.

Our drone uses a three-camera to take 360-degree photography instead of a 360 camera. A 360-degree camera could capture all information in every degree that it sometimes lacks focus.

One of the biggest challenges is the large barrel distortion caused by the ultra-wide angle fisheye lens. Even the 360-degree camera is not versatile compared to traditional cameras. For example, all the photos taken with a 360-degree camera can look too similar, which is a problem for us since we need to detect the movement of an object in an area. Detecting the type of object becomes more complicated and can be catastrophic. Also, it is very difficult to correct radial

distortions within 360 degrees. The wide-angle camera we use with drones is severely affected by radial distortion, so correcting radial camera distortion is an important step toward 100% object coordinate accuracy. Nowakowski and Skarbek's method [16] uses a Homography of Central Points for Lens Radial Distortion Calibration.

However, the method used to determine the center of distortion gives accurate results with no errors. Zhu et al. [17] methods use QR Factorization to correct the radial distortion in a non-iterative way, which can be faster but misses some pieces of the picture. Henrique Brito et al. [18] methods propose Self-Calibration Based on observing straight lines through the distortion center. Zhang et al. [19] propose a new robust line-based distortion estimation method to correct radial distortion.

Cho et al. [20] propose the Automatic Estimation of the Distortion Coefficient method, which performs well in radial distortion estimation for more correction, the algorithm needs to be repeated to satisfy the termination condition. Kim et al. [21] solve radial distortion compensation by Illuminating the epipolar lines with a Projector [22].

Propose using a cascaded one-parameter split model that requires execution time for each block with a repetitive process to achieve a satisfactory result. The method developed by Huang et al. [23] is founded on the principles of direct linear transformation. Liao et al. [24] propose an estimation approach for distortion rectification based on the Training Process of the Proposed Network algorithm that contains two loops of repetition inside each other to increase the result performance.

In our situation, in the presence of a drone with three-camera support, things get different, and it is attributed to the drone's movement. Therefore, the Euler coordinates change every moment, making it difficult to determine the actual coordinates.

Huu et al. [25] propose introducing two fixed camera models to calculate the distance between the camera system and the installation.

In this research, we provide a mathematical model to correct camera distortion and identify genuine object coordinates in real-time. Single-board computers are preferred in these circumstances. These single-board computers have microprocessors, memory, input/output, and other useful components and are constructed on a single circuit board [26].

Due to the possibility of combining different fields of technology, single-board computer systems with a wide range of applications and a low price are often chosen [27].

AI applications for single-board computers have increased due to technological advances and the approach to PC performance. The use of deep learning, one of the sub-branches of artificial intelligence [28-37] documents [38, 39], and autonomous/mobile systems [40-43] in single-board computers have also increased as some single-board computers support both CPU and GPU.

The rest of the work can be summarized as follows: in the second section, an overview of the work that describes the basic concept of the surveillance system, and in section III, the proposed research work is explained. In the next section, we discuss experimental results. Finally, in section V, we conclude the proposed research work and suggest some future directions.

2. OVERVIEW

In this work, we propose our SoC-based computing system in which the CPU performs multitasking between inputs, large processing units, and outputs. We optimized motion detection and object detectors forever after calculating the center of mass coordinates and implementing them on different embedded systems. We found the best processing time using our proposed system that can perform a better result with low energy consumption, increasing the flying time of the UAV.

Figure 1 shows an overview of the proposed surveillance system. As shown in the figure, the autonomous drone records video data with three cameras tilted at 130 degrees to each other. Each camera recognizes the scene in its field of view. A specific processing system processes this extracted information (frame). For each camera, there is a sub-algorithm in the overall algorithm system.

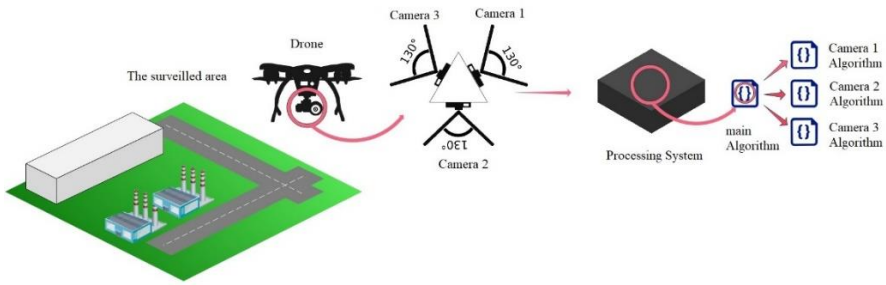

Figure 1. Proposed surveillance system overview

3. METHODOLOGY

The proposed method consists of eight main processing steps for efficient drone operation. Initially, three cameras connected to the drone record all the scenes within their respective fields of view. Subsequently, noise-removing techniques to enhance data quality are applied to the stream. The third stage involves motion detection in the recorded scenes, identifying areas with movement. Once motion is detected, the system proceeds to generate a scan of the affected region in the fourth stage. In the fifth step, the method identifies the nature of objects within the scanned area, enabling the calculation of their center of mass coordinates. These center of mass coordinates serve as a crucial input for the sixth stage, where the actual (x, y, z) coordinates are computed. These real-world coordinates are saved in a structured CSV file within the system's database. Finally, the results are displayed on the supervisor's screen in the last step, providing valuable insights and actionable information.

Figure 2 represents the eight-step process that ensures the effective use of the drone system for various applications, including surveillance and data collection.

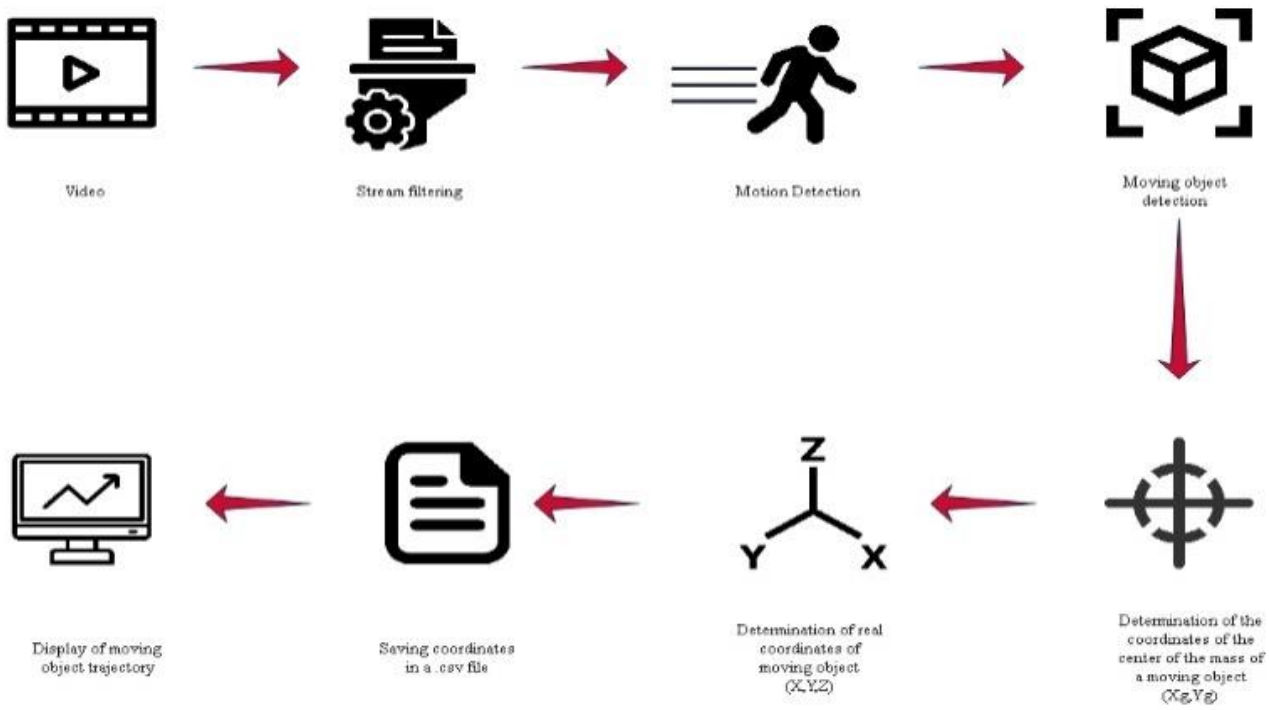

Figure 2. Approach overview

3.1 Frame extraction

Streaming from three cameras or real-time video collection (Figure 3), which can reflect a series of N images and stand for by S= (fs1; fs2; …; fsN). Given the build quality, it's better to use a triple camera instead of a 360-degree camera. Because RGB represents the red, green, and blue components, each video image is recognized as a color image. Therefore, the images are periodically moved to the next step.

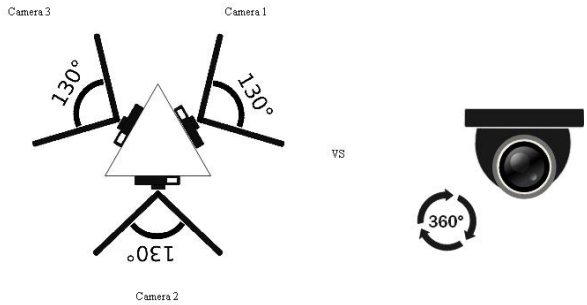

Figure 3. Three-camera position

3.2 Motion detection

Motion detection serves a variety of functions in this application. After we have detected a movement of one of the three cameras, we immediately start the security process.

Real-time capturing can be treated as a set of frames. Here, we compared different frames to the first frame using the frame differencing technique. Figure 4 shows the motion detection algorithm used for each frame.

We need to look at some tiny, moving fragments of the picture, not the whole. This will reduce our drone's energy and speed up our processing system.

3.3 Object detection

Within Deep Learning, the sub-discipline (Object Detection) involves this application to identify objects through real-time video. Essentially expressed, this detection approach aims to locate objects in the frame (object localization), which will help us to track the item in the following processing step.

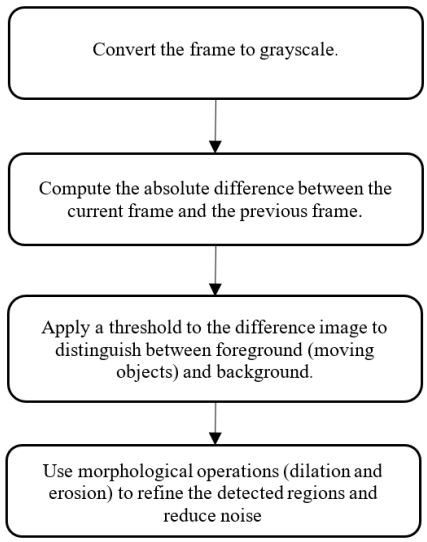

Figure 4. The motion detection algorithm

For this method, you need image processing algorithms to verify image content. We must follow several guidelines offered by the manufacturers (Intel, INVIDIA, Raspberry Pi, etc.) to implement our approach on various hardware. These guidelines enable us to apply these AI models to our UAV application. For instance, the OpenVINO toolkit offers a collection of pre-trained Intel models that can be employed for software development, learning, and demo purposes.

Various detection models can detect a set of the most common objects. Object detection with SSD-MobileNet v2 framework is widely used for real-time object detection. For the surveillance area, MobileNet V2 is the network used for the feature extractor and is the object localizer. Since most

networks are SSD-based and offer a reasonable adjustment between efficiency and performance, we decided to use Mobile Networking SSD v2 for the object detection part. Networks that detect objects and offer the option of higher accuracy/broad application at the cost of lower performance can be expected to detect objects of the same type more exactly.

3.4 Intensity weighted centroiding coordinates of moving objects for accurate area tracking

Once we have the position of a moving object in the image, we use intensity weighted centroid (IWC) to detect it.

The center of gravity (CoG) is the basis of the IWC calculations [44]. CoG is the same calculation as in physics but only applied to the image Figure 5.

$$(x_g, y_g) = \left[\frac{\sum_{ij} I_{ij}^2 x_{ij}}{I_{ij}}, \frac{\sum_{ij} I_{ij}^2 y_{ij}}{I_{ij}}\right] \quad (1)$$

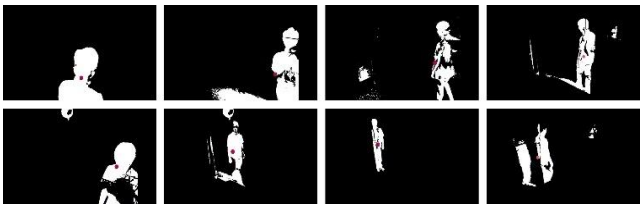

Figure 5. Object intensity weighted centroiding coordinates

3.5 Determination of the correct real coordinates of a moving object

As soon as IWC coordinates are identified, we calculate real (X,Y,Z) coordinates. For this, we use the pinhole camera model [45]. This model uses perspective transformation to project 3D points onto the image plane to create a view of the scene. Figure 6. The following equation, Eq. (2), can give us the actual coordinates [45-48].

$$x_g = A[R|T]M \quad (2)$$

where,

- x_g: 3-D IWC Coordinates
- A: Camera Matrix or a Matrix of Intrinsic Parameters
- $[R|T]$: Rotation-Translation Matrix
- M: 3-D Real Coordinates

$$\begin{bmatrix} \frac{\sum_{ij} I_{ij}^2 x_{ij}}{I_{ij}} \\ \frac{\sum_{ij} I_{ij}^2 y_{ij}}{I_{ij}} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (3)$$

As we know, the drone does not stay in its initial position, and there is an infinitely small shift δψ between the two frames, Figure 6 and Figure 7, performed at any instant in our calculation Figure 8. We can straighten the difference in the following Eq. (4).

$$x_g{}' = A[R|T]M' \quad (4)$$

where,

- $M' = M + \delta\psi$
- $x_g' = x_g + \delta x_g$
- $\delta\psi = (\delta X, \delta Y, \delta Z, 1)$
- $\delta x = (\delta x_g, \delta y_g, 1)$

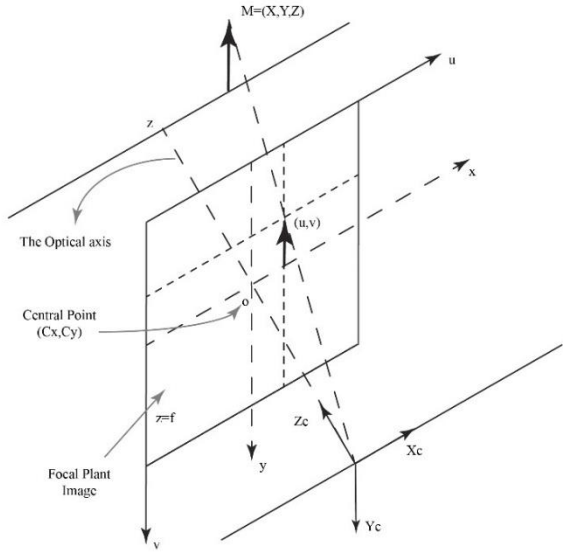

Figure 6. Optical schematic of the object coordinate

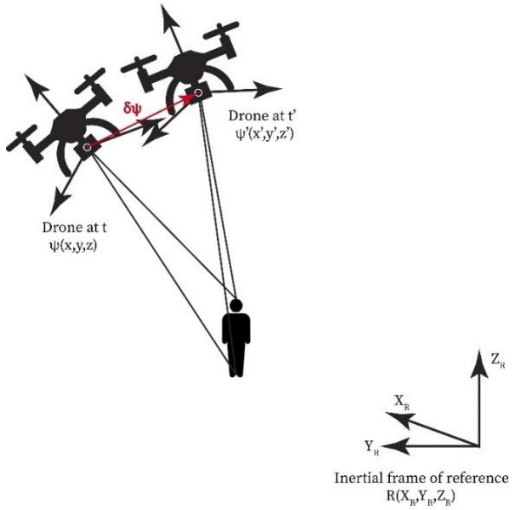

Figure 7. Drone displacement explanation

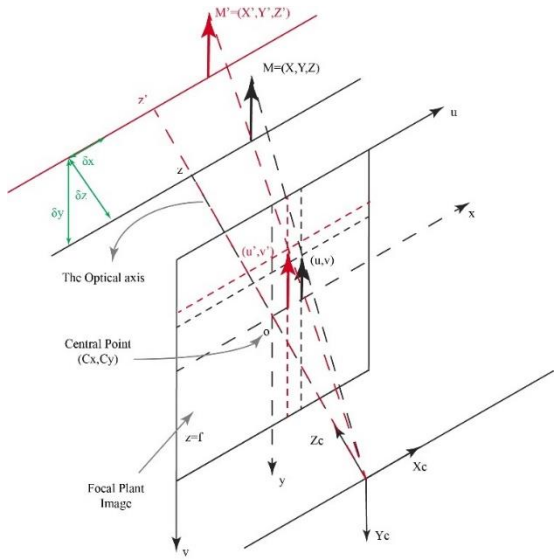

Figure 8. Optical schematic of infinitesimal displacement explanation

For camera calibration and distortion correction applications, these distortions must be corrected first. To determine these parameters, we provide some sample images of a known pattern (such as a chessboard). We find some specific points (square corners on the chessboard). We have its coordinates in the real world, and its coordinates in the image. With these data, some mathematical processes are in the background to get the distortion coefficients.

4. SYSTEM PROCESSING ARCHITECTURE OF THE IMPLEMENTATION

To figure out our approach, many processing systems can be exploited, but we need to find a compatible processing system with our surveillance system. There are two main factors to consider when choosing the suitable device for us, Suitable energy System and processing time.

4.1 CPU: Parallel computing

The "core," or heart of the CPU, is where all computing and reasoning takes place. A core normally runs through a process known as the "instruction cycle," in which instructions are read from memory, converted to processing language, and then executed through the core's logical gates (execute). Initially, all processors were single-core, but as multi-core processors became more common, computing power increased, and parallel processing arrived Figure 9.

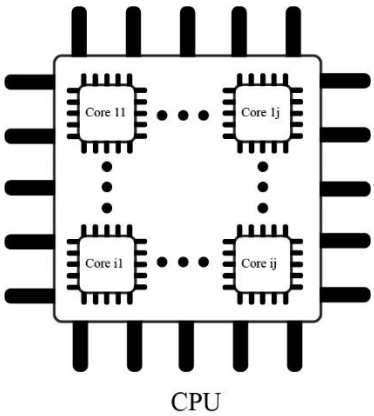

Figure 9. CPU internal architect

Most parallel computing hardware is often housed in a single data center with multiple processors (or cores within a processor) advanced over a server rack. The application server distributes compute requests in small chunks, subsequently executed concurrently on each server.

Security surveillance using UAVs and embedded systems, CPUs are suitable for managing lower-complexity tasks like data preprocessing, basic motion detection, and simple object identification.

4.2 CPU and GPU computing

Graphical processing is one of these activities, typically regarded as one of the CPU's more difficult processing tasks. Because of the complexity of the solution, technology now has uses far beyond graphics. The difficulty in processing graphics is that, to render properly, visuals require complicated mathematics, which must be computed in parallel. For instance, a graphically demanding computer game may simultaneously have hundreds or thousands of polygons on the screen, each with its movement, color, lighting, and other characteristics. Such a workload is not designed for CPUs. Graphical processing units (GPUs) are used in this situation.

GPUs perform similarly to CPUs, having cores, memory, and other parts. GPU acceleration emphasizes parallel data processing with a high number of cores rather than context switching to manage many activities. Typically, each of these cores is less powerful than the CPU core. Additionally, GPUs are frequently incompatible with various hardware APIs and homeless storage. They perfectly support the simultaneous transfer of many processed data.

The GPU takes batch instructions and transmits them at high volume for faster processing and display instead of switching between graphics processing tasks Figure 10.

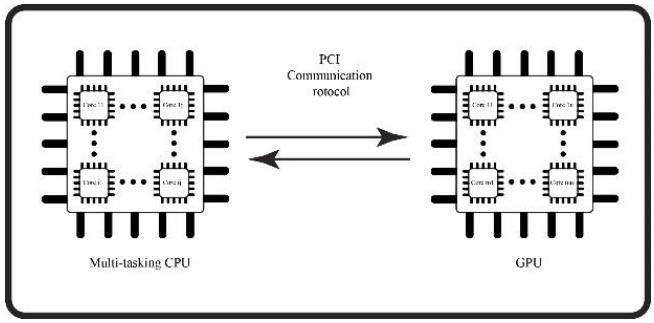

Figure 10. CPU and GPU internal architect

With GPUs, they can simultaneously increase application data throughput and the number of active computations.

Because of parallelism, the GPU can do more work than the CPU in a given period.

GPUs excel in parallel processing and are well-suited for tasks that require high-throughput computations, such as image and video processing. For security surveillance, GPUs are advantageous in scenarios that involve real-time object detection, tracking, and advanced image analysis.

4.3 Cluster computing

In cluster computing, a group of closely related or ad hoc computers work together to function as a single unit. The collective action of connected computers creates the idea of a single system. Typically, fast local area networks are used to connect the clusters (LANs). Cluster computing offers a great low-cost alternative to huge server or mainframe platforms. The demand for content criticality is met, and services are processed more quickly Figure 11.

Many businesses use cluster computing and IT companies to improve scalability, availability, processing speed, and resource management at a reasonable cost. It guarantees constant access to computing power. It offers a unique, superior approach to designing and operating high-performance parallel systems not dependent on specific hardware dealers or their product line choices [49]. However, clusters might be less suitable in the context of UAVs due to their inherent resource constraints and the need for real-time processing.

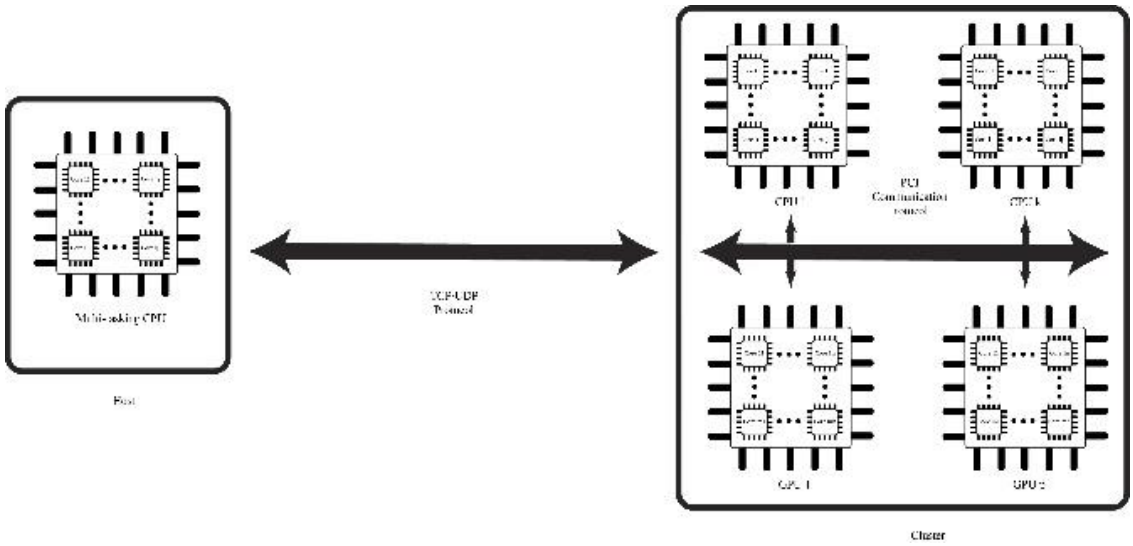

Figure 11. Cluster internal architect

4.4 Vision Processing Unit (VPU) computing

A later class of microprocessors called the Vision Processing Unit (VPU) is a special type of AI accelerator designed specifically to speed up operations that use computer vision. The new Vision Processing Unit (VPU) is a quick 500MHz DSP (ISP) linked with the Image Signal Processors Figure 12.

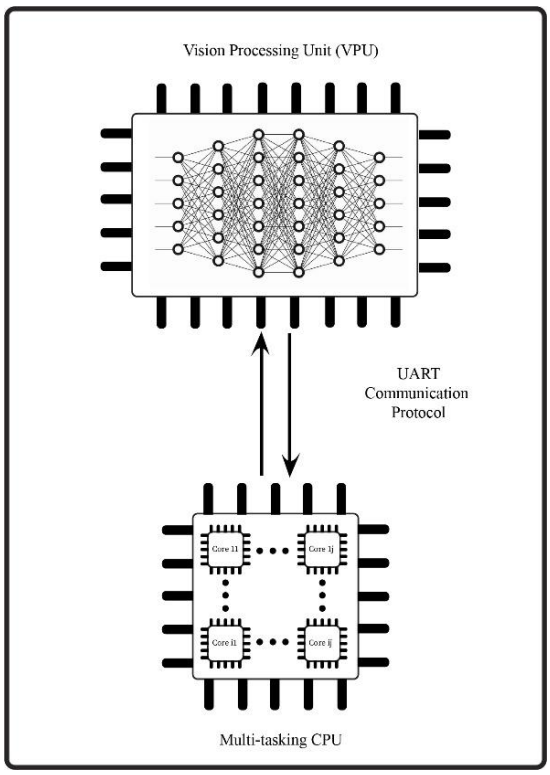

Figure 12. Vision Processing Unit (VPU) internal architect

Real-time Depth of Field is just one of the fascinating camera functions. It offers a dedicated processing platform for freeing up the CPU and GPU to conserve power and computing resources.

The platform provided by VPU allows companies to differentiate their products by customizing camera features.

It is a specialized hardware that supports cameras and can carry out real-time processing tasks. In the past, these were often passed on to the CPU or GPU, but the VPU consumes only a fraction of the power. The VPU can still be used alone or in conjunction with the CPU/GPU in a truly heterogeneous computing environment on the same memory subsystem for complex multi-application or multi-function activities.

For security surveillance using UAVs, VPUs are highly suitable as they provide efficient and fast processing of visual data. They are particularly valuable for real-time object detection, tracking, and other computationally intensive vision tasks. VPUs optimize power consumption while delivering the required processing power for surveillance applications, making them well-suited for embedded systems on UAVs.

In summary, as shown in Table 1, while each architecture has its strengths, a VPU emerges as a promising option for security surveillance using UAVs and embedded systems. It offers a well-balanced combination of high processing speed, power efficiency, real-time capabilities, and accuracy, making it suitable for real-time object detection, tracking, and image analysis-all essential for effective security surveillance applications.

Table 1. The key attributes of each embedded system

Attribute	CPU	GPU	Cluster	VPU
Processing Speed	Moderate	High	High	High
Power Efficiency	Good	Moderate	High	Good
Precision/ Accuracy	Moderate	High	High	High
Parallel Processing	Limited	Excellent	Excellent	Limited
Real-Time Capability	Yes (For some)	Yes	Possible	Yes
Scalability	Limited	Moderate	High	Limited
Portability	Yes	No	No	Yes
Cost	Affordable	Moderate to High	Variable	Moderate
Hardware Integration	Common	Specialized	Variable	Specialized
Suitable Application	Basic processing, Coordination	Real-time object detection, tracking	Large-scale data processing	Real-time image analysis object detection

5. SYSTEMS PERFORMANCE AND EVALUATION RESULTS

5.1 Algorithm

Algorithm 1

Begin:
Input: $frame_i$, $frame_{i+1}$
Camera Matrix or a Matrix of Intrinsic Parameters
Rotation-Translation Matrix $[\boldsymbol{R}|\boldsymbol{T}]$
UAV Initial Position
Initialize: $\boldsymbol{frame_i}$**=CaptureFirstFrame()**
While ($frame_{i+1}$ captured)
Save $frame_{i+1}$
Pretreatment, noise removal, and filtering
Match ($frame_{i+1}$ pixels) with (*frame* pixels)
If (any motion is generated in the vision area)
Localization of motion area
Objection Identification in the Motion Area
Calculation of IWC coordinates (x_g,y_g) for every object in the motion area
Determination of δψ between *frames* and $frame_{i+1}$
Calculation of the real coordinates x_g'
Save in .csv file (Time (t)||Object id||(x_g,y_g)||x_g')
Print real objects' trajectory
Remove *frame*
frame ← $frame_{i+1}$
Else (capture new $frame_{i+1}$)

5.2 Implementation

The different hardware systems of processing system architects discussed earlier are employed to figure out our approach. Raspberry Pi 4b (Figure 13), jetson nano (Figure 14), Raspberry Pi 4b+Intel Neural Compute Stick 2 (Intel NCS2) (Figure 15), Personnel Computer (Figure 16), and Google collab cluster to work out the performance of successively Parallel CPU (CPU and GPU), Cluster, Vision Processing Unit (VPU) computing. For model execution on Movidius NCS 2, Intel provides an Opensource deep learning toolkit package, OpenVINO. The OpenVINO toolkit allows us to deploy pre-trained deep learning models, via a high-level Python programming language Inference Engine API paired with application logic. The model must be restructured into an Intermediate Representation (IR) network, which can be inferred by the Inference Engine. IR consists of two binary files, which are.xml and .bin files. Our work is performed on a Windows system for Personal computers and clusters, and in Linux for other systems by using the model optimizer built-in to the OpenVINO package.

System processing the average processing time and power consumption are the most expensive parts that we have to calculate. It depends on the parameters of the model and its variables, including the number of layers, the number of cores, the size of the core, and the activation function. Figure 17 shows the video processing results of a moving object in the monitored area.

To measure processing time, the system uses a time stamp at the start and end of each processing step. The difference between the start and end time stamps provides the elapsed time for that step. This time is measured in milliseconds. The time module integrated into the Python programming language was used in our case. Power is quantified using sensors that measure components' electrical current and voltage. These measurements are then used to calculate power consumption using the following formula:

$$P[Watts] = U\,[Volts] \times I[Amps] \quad (5)$$

The workflow scenario included in this work concerns motion detection and identification of moving vehicles in an industrial complex and tracking their trajectory in an open space using the different monitoring platforms. Input images were captured from a video. The initial settings default to the coordinates of the proposed drone system. As mentioned in Table 1, the Raspberry Pi 4 has poor performance and high power consumption. Jetson Nano and Cluster are proven to have higher power sources and consumption due to their high hardware capabilities. If we analyze the model classification over time, we see that it happens faster (Raspberry Pi 4+VPU). But as for the power consumption, we can see that (Raspberry Pi 4+VPU) is a lower power consumption system. This system is the preferred processing system of our surveillance system, consumes less power and has a short average processing time. NVIDIA Jetson Nano has adequate performance for our surveillance UAV applications. For resolutions up to 4K, real-time performance and energy consumption.

It uses CUDA "Compute Unified Device Architecture" an architecture developed by NVIDIA for parallel calculations. On the opposite side, some of the limits of this card have been encountered, the setting up of the environment is rather complicated in addition it does not support some last tools versions.

Intel hardware offers high performance, deep learning, simplified development, write once, and deploy anywhere.

Intel's generation of the OpenVINO toolkit makes accepting and maintaining our approach easier.

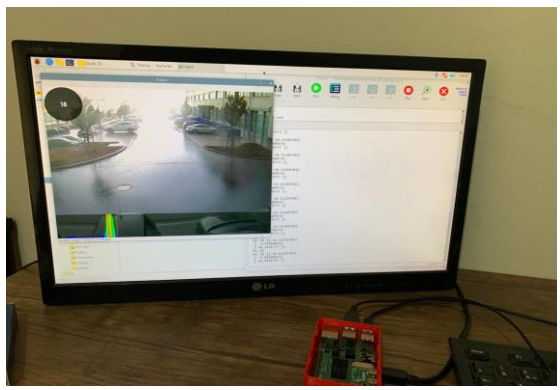

Figure 13. Raspberry Pi processing system

Figure 14. Nvidia Jetson Nano processing system

Figure 15. (Raspberry Pi+VPU) processing system

Figure 16. Personnel computer processing system

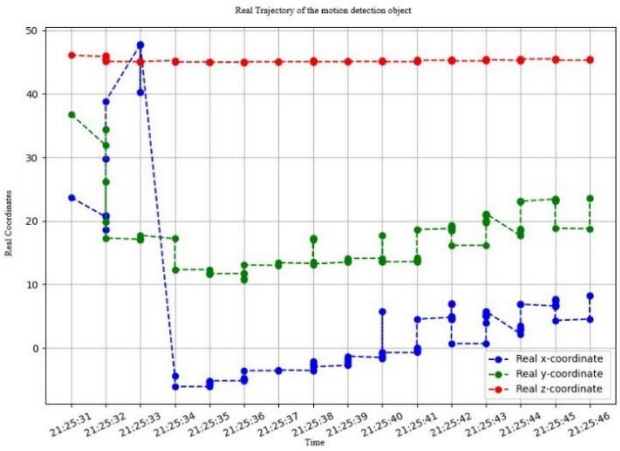

Figure 17. The real trajectory of the moving object in the surveillance area

By building an optimized network and controlling inference processes on specific devices, we can use the runtime (inference engine) to optimize performance. Optimization is also done automatically by detecting peripherals, balancing load, and inferring parallelism between CPU, GPU, and VPU.

The results prove that though system processing average processing time and processing system consumption marginally went down with (The Raspberry Pi+VPU) system, we noticed a significant increase in (Personal computer and Cluster) systems. Our (Raspberry Pi+VPU) system Figure 18. Achieves those two parameters, 1watt and 18 ms lower than (Personal computer and Cluster), respectively. At the same time, the parameters of each system are shown in Figure 17.

In this work, the results attained are compared with the ones achieved by the Small-Scale Object Detection for Unmanned Aerial Vehicles (UAVs) system proposed by Saeed et al. which modified the architecture of the detection network and executed on different embedded systems, as we present early our system can detect and locate the object in the surveillance area in the real-time [50]. Singhal and Barick also proposes an application-aware Multi-Path Weighted Load-balancing (MWL) routing protocol for managing congestion, this system executes its process in the ground center, which increases the processing time and makes it out of service and powerless in the event of interruption or penetration [51]. Teng et al. developed a trajectory planner based on particle swarm optimization with surveillance area priority, exploiting highly consumed existing UAVs to obtain optimal trajectories [52]. In our work, we provide a surveillance system that can reduce the energy consumed and processing time to locate any object in the surveilled area, whatever the object's trajectory.

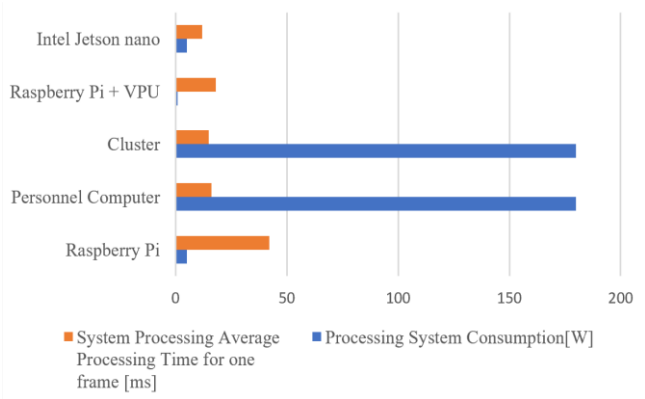

Figure 18. System processing average time and processing system consumption of each system

6. CONCLUSION

This paper presents a novel approach to industrial area surveillance, which combines the Raspberry Pi 4B with an Intel Neural Compute Stick 2 VPU as the edge computing device to provide high image processing with low consumption of energy which can increase the UAV flying time in the surveilled area. In this study, performance tests of our big data approach were performed on Jetson Nano and Raspberry PI 4 boards, clusters, and personal computers (Raspberry PI 4+Intel VPU). Performance benchmarks included power consumption and average processing time.

We want to ensure a minimum of hardware, cost, and hardware choice in our real-time monitoring applications after the benchmark evaluation. In this context, the model was developed using the CNN algorithm from deep learning algorithms, and mathematical equations were transferred between real-world coordinates and image (pixel) coordinates.

According to the test results, the cluster consumes more power but delivers better performance with a shorter average processing time. The major challenge of this system in our application is the certainty of achieving a shorter average processing time. We need to transfer data over UDP instead of TCP communication protocol, and UDP protocol is less secure than TCP. In the last part, the system (Raspberry PI 4+Intel VPU) is the preferred processing system for our surveillance system. It uses less power and has a pretty good average processing time Table 2.

Table 2. Technical specifications of processing systems and results

Processing Systems	Raspberry Pi	Personnel Computer	Cluster	Raspberry Pi+VPU & Intel	Jetson Nano
Processing System Features	Quad-core Cortex-A72 (ARM v8) 64-bit 1.5GHz and 8GB RAM	Intel Core i7-8750H CPU with 12 cores, 16 GB of RAM	800 computer nodes having from 2 to 16 virtual processors with 4 to 32GB of RAM	1 GB of RAM 4 GB free stockage space	quad-core Cortex-A57, GPU 128-core 4Gb RAM
Processing System Consumption	2,89-7,28 W	180-watt	180-watt	1 watt	& 5 watts
System Processing Average Processing Time for One Frame	42 ms	16 ms	15 ms	18 ms	12 ms
Price	330$	1150$	-	520$	800$
FPS	32	32	32	16	32

REFERENCES

[1] Ye, T., Zhao, Z.Y., Zhang, J., Chai, X.H., Zhou, F.Q. (2021). Low-altitude small-sized object detection using lightweight feature-enhanced convolutional neural network. Journal of Systems Engineering and Electronics, 32(4): 841-853. https://doi.org/10.23919/JSEE.2021.000073

[2] Li, X.X., Zhu, L.Q., Yu, Z.J., Wan, Y.Q. (2019). Adaptive auxiliary input extraction based on vanishing point detection for distant object detection in high-resolution railway scene. In 2019 14th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), Changsha, China, pp. 522-527. https://doi.org/10.1109/ICEMI46757.2019.9101454

[3] Chiu, C.C., Lo, W.C. (2020). An object detection algorithm with disparity values. In 2020 4th International Conference on Imaging, Signal Processing and Communications (ICISPC), IEEE, Kumamoto, Japan, pp. 20-23. https://doi.org/10.1109/ICISPC51671.2020.00011

[4] Wang, S.Q., Chen, Z.Q., Ding, Z.Y. (2019). The unified object detection framework with arbitrary angle. In 2019 5th International Conference on Big Data and Information Analytics (BigDIA), IEEE, Kunming, China, pp. 103-107. https://doi.org/10.1109/BigDIA.2019.8802710

[5] Chen, Z.Y., Gao, H.L., Wang, K. (2020). A motion based object detection method. In 2020 2nd International Conference on Information Technology and Computer Application (ITCA), IEEE, Guangzhou, China, pp. 280-283. https://doi.org/10.1109/ITCA52113.2020.00067

[6] Lee, T.H., Kang, Y.G., Ryu, S., Lee, H.J. (2020). An ASMO method for CNN-based occluded object detection. In 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), IEEE, Seoul, Korea (South), pp. 1-2. https://doi.org/10.1109/ICCE-Asia49877.2020.9277112

[7] Xue, Z.J., Chen, W.J., Li, J. (2020). Enhancement and fusion of multi-scale feature maps for small object detection. In 2020 39th Chinese Control Conference (CCC), IEEE, Shenyang, China, pp. 7212-7217. https://doi.org/10.23919/CCC50068.2020.9189352

[8] Saini, A., Biswas, M. (2019). Object detection in underwater image by detecting edges using adaptive thresholding. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE, Tirunelveli, India, pp. 628-632. https://doi.org/10.1109/ICOEI.2019.8862794

[9] Siaho, D., Ghislain, P.K., Lambert, K.T., Ernest, K.K., Souleymane, O., Emmanuel, A.N. (2023). Modeling artificial neural network of insect's proliferation during cocoa beans storage. Ingénierie des Systèmes d'Information, 28(2): 291-298. https://doi.org/10.18280/isi.280204

[10] Kilic, I., Aydin, G. (2020). Traffic sign detection and recognition using tensorflow's object detection API with a new benchmark dataset. In 2020 International Conference on Electrical Engineering (ICEE), IEEE, Istanbul, Turkey, pp. 1-5. https://doi.org/10.1109/ICEE49691.2020.9249914

[11] Moon, S.W., Lee, J., Lee, J., Nam, D., Yoo, W. (2020). A comparative study on the maritime object detection performance of deep learning models. In 2020 International Conference on Information and Communication Technology Convergence (ICTC), IEEE, Jeju, Korea (South), pp. 1155-1157. https://doi.org/10.1109/ICTC49870.2020.9289620

[12] Jakkulla, P.K., Ganesh, K.M., Jayapal, P.K., Malla, S.J., Chandanapalli, S.B., Sandhya, E. (2023). Selection of features using adaptive tunicate swarm algorithm with optimized deep learning model for thyroid disease classification. Ingenierie des Systemes d'Information, 28(2): 299-308. https://doi.org/10.18280/isi.280205

[13] Li, W.T., Li, W.Y., Yang, F., Wang, P. (2019). Multi-scale object detection in satellite imagery based on YOLT. In IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, pp. 162-165. https://doi.org/10.1109/IGARSS.2019.8898170

[14] Yedla, S.K., Manikandan, V.M., Panchami, V. (2020). Real-time scene change detection with object detection for automated stock verification. In 2020 5th International Conference on Devices, Circuits and Systems (ICDCS), IEEE, Coimbatore, India, pp. 157-161. https://doi.org/10.1109/ICDCS48716.2020.243571

[15] Vengurlekar, S.G., Jadhav, D., Shinde, S. (2019). Object detection and tracking using Zernike moment. In 2019 International Conference on Communication and Electronics Systems (ICCES), IEEE, Coimbatore, India, pp. 12-17. https://doi.org/10.1109/ICCES45898.2019.9002556

[16] Nowakowski, A., Skarbek, W. (2007). Lens radial distortion calibration using homography of central points. In EUROCON 2007-The International Conference on" Computer as a Tool", IEEE, Warsaw, Poland, pp. 340-

343. https://doi.org/10.1109/EURCON.2007.4400578
[17] Zhu, W., Diao, C., Huang, J. (2010). Calibration of radial distortion via QR factorization. In 2010 IEEE International Conference on Progress in Informatics and Computing, 2: 728-732. https://doi.org/10.1109/PIC.2010.5687940
[18] Henrique Brito, J., Angst, R., Koser, K., Pollefeys, M. (2013). Radial distortion self-calibration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1368-1375.
[19] Zhang, L., Shang, H., Wu, F., Wang, R., Sun, T., Xie, J. (2019). Robust line-based radial distortion estimation from a single image. IEEE Access, 7: 180373-180382. https://doi.org/10.1109/ACCESS.2019.2959204
[20] Cho, W., Lee, M., Kim, H., Paik, J. (2018). Automatic estimation of distortion coefficient for correcting radial distortion. In 2018 International Conference on Electronics, Information, and Communication (ICEIC), Honolulu, HI, USA, IEEE, pp. 1-3. https://doi.org/10.23919/ELINFOCOM.2018.8330606
[21] Kim, Y., Kim, D., Lee, S. (2010). Camera lens radial distortion compensation by illuminating the epipolar lines with a projector. In ICCAS 2010, IEEE, Gyeonggi-do, Korea (South), pp. 1871-1874. https://doi.org/10.1109/ICCAS.2010.5670171
[22] Mei, X., Yang, S., Rong, J., Ying, X., Huang, S., Zha, H. (2015). Radial lens distortion correction using cascaded one-parameter division model. In 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, pp. 3615-3619. https://doi.org/10.1109/ICIP.2015.7351478
[23] Huang, K., Ziauddin, S., Zand, M., Greenspan, M. (2020). One shot radial distortion correction by direct linear transformation. In 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, pp. 473-477. https://doi.org/10.1109/ICIP40778.2020.9190749
[24] Liao, K., Lin, C., Zhao, Y. (2021). A deep ordinal distortion estimation approach for distortion rectification. IEEE Transactions on Image Processing, 30: 3362-3375. https://doi.org/10.1109/TIP.2021.3061283
[25] Huu, P.N., Van, T.T., Thi, N.G. (2019). Proposing distortion compensation algorithm for determining distance using two cameras. In 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), IEEE, Hanoi, Vietnam, pp. 172-177. https://doi.org/10.1109/NICS48868.2019.9023875
[26] Oleksandr, O., Molga, A., Kochan, V., Sachenko, A. (2020). Method of ensuring the noise immunity at measurement of single-board microcontroller average energy within IoT environment. In 2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO), Kyiv, Ukraine, pp. 807-810. https://doi.org/10.1109/ELNANO50318.2020.9088880
[27] Vavrenyuk, A.B., Matveeva, D.V., Lukyantsev, N.M., Makarov, V.V. (2021). Analysis of an efficiency of parallelization of algorithms running on computing cluster based on single-board diskless computers raspberry PI 3 model B. In 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Petersburg, Moscow, Russia, pp. 728-730. https://doi.org/10.1109/ElConRus51938.2021.9396277
[28] Süzen, A.A., Duman, B., Şen, B. (2020). Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-CNN. In 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), IEEE, Ankara, Turkey, pp. 1-5. https://doi.org/10.1109/HORA49412.2020.9152915
[29] Inthanon, P., Mungsing, S. (2020). Detection of drowsiness from facial images in real-time video media using NVidia Jetson Nano. In 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), IEEE, Phuket, Thailand, pp. 246-249. https://doi.org/10.1109/ECTI-CON49241.2020.9158235
[30] Rehman, A., Yar, H., Ayesha, N., Sadad, T. (2020). Dermoscopy cancer detection and classification using geometric feature based on resource constraints device (Jetson Nano). In 2020 13th International Conference on Developments in eSystems Engineering (DeSE), IEEE, Liverpool, United Kingdom, pp. 412-417. https://doi.org/10.1109/DeSE51703.2020.9450750
[31] Koubaa, A., Ammar, A., Kanhouch, A., AlHabashi, Y. (2021). Cloud versus edge deployment strategies of real-time face recognition inference. IEEE Transactions on Network Science and Engineering, 9(1): 143-160. https://doi.org/10.1109/TNSE.2021.3055835
[32] Slater, W.S., Tiwari, N.P., Lovelly, T.M., Mee, J.K. (2020). Total ionizing dose radiation testing of NVIDIA Jetson nano GPUs. In 2020 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, pp. 1-3. https://doi.org/10.1109/HPEC43674.2020.9286222
[33] Kamal, R., Chemmanam, A.J., Jose, B.A., Mathews, S., Varghese, E. (2020). Construction safety surveillance using machine learning. In 2020 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, Montreal, QC, Canada, pp. 1-6. https://doi.org/10.1109/ISNCC49221.2020.9297198
[34] Ullah, S., Kim, D.H. (2020). Benchmarking jetson platform for 3D point-cloud and hyper-spectral image classification. In 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Korea (South), pp. 477-482. https://doi.org/10.1109/BigComp48618.2020.00-21
[35] Kim, J.U., Ro, Y.M. (2019). Attentive layer separation for object classification and object localization in object detection. In 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, pp. 3995-3999. https://doi.org/10.1109/ICIP.2019.8803439
[36] Marshall, M.R., Hellfeld, D., Joshi, T.H.Y., Salathe, M., Bandstra, M.S., Bilton, K.J., Cooper, R.J., Curtis, J.C., Negut, V., Shurley, A.J., Vetter, K. (2020). 3-d object tracking in panoramic video and lidar for radiological source-object attribution and improved source detection. IEEE Transactions on Nuclear Science, 68(2): 189-202. https://doi.org/10.1109/TNS.2020.3047646
[37] Ye, T., Zhang, J., Li, Y., Zhang, X., Zhao, Z., Li, Z. (2022). CT-Net: An efficient network for low-altitude object detection based on convolution and transformer. IEEE Transactions on Instrumentation and Measurement, 71: 1-12. https://doi.org/10.1109/TIM.2022.3165838
[38] Kozlov, D.V., Stepanov, A.B. (2021). The main features of a multichannel continuous wavelet transform

implementation on the Nvidia Jetson single board computers. In 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Petersburg, Moscow, Russia, pp. 1632-1636. https://doi.org/10.1109/ElConRus51938.2021.9396291

[39] Fontaine, J., Shahid, A., Elsas, R., Seferagic, A., Moerman, I., De Poorter, E. (2020). Multi-band sub-GHz technology recognition on NVIDIA's Jetson Nano. In 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Victoria, BC, Canada, pp. 1-7. https://doi.org/10.1109/VTC2020-Fall49728.2020.9348566

[40] Uddin, M.I., Alamgir, M.S., Rahman, M.M., Bhuiyan, M.S., Moral, M.A. (2021). Ai traffic control system based on deepstream and iot using NVidia jetson nano. In 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), IEEE, DHAKA, Bangladesh, pp. 115-119. https://doi.org/10.1109/ICREST51555.2021.9331256

[41] Vijitkunsawat, W., Chantngarm, P. (2020). Comparison of machine learning algorithm's on self-driving car navigation using Nvidia Jetson Nano. In 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), IEEE, Phuket, Thailand, pp. 201-204. https://doi.org/10.1109/ECTI-CON49241.2020.9158311

[42] Wang, L., Ye, X., Xing, H., Wang, Z., Li, P. (2020). Yolo nano underwater: A fast and compact object detector for embedded device. In Global Oceans 2020: Singapore-US Gulf Coast, IEEE, Biloxi, MS, USA, pp. 1-4. https://doi.org/10.1109/IEEECONF38699.2020.9389213

[43] Nguyen, S.T., Le, B.N., Dao, Q.X. (2021). AI and IoT-powered smart university campus: design of autonomous waste management. In 2021 International Symposium on Electrical and Electronics Engineering (ISEE), IEEE, Ho Chi Minh, Vietnam, pp. 139-144. https://doi.org/10.1109/ISEE51682.2021.9418672

[44] Akondi, V., Roopashree, M.B., Budihala, R.P. (2010). Improved iteratively weighted centroiding for accurate spot detection in laser guide star based Shack Hartmann sensor. In Atmospheric and Oceanic propagation of Electromagnetic Waves IV, SPIE, 7588: 44-54. https://doi.org/10.1117/12.841331

[45] Bräuer-Burchardt, C., Ramm, R., Kühmstedt, P., Notni, G. (2022). The duality of ray-based and pinhole-camera modeling and 3D measurement improvements using the ray-based model. Sensors, 22(19): 7540. https://doi.org/10.3390/s22197540

[46] Wang, G., Tsui, H.T., Hu, Z., Wu, F. (2005). Camera calibration and 3D reconstruction from a single view based on scene constraints. Image and Vision Computing, 23(3): 311-323. https://doi.org/10.1016/j.imavis.2004.07.008

[47] Poulin-Girard, A.S., Thibault, S., Laurendeau, D. (2016). Influence of camera calibration conditions on the accuracy of 3D reconstruction. Optics Express, 24(3): 2678-2686. https://doi.org/10.1364/OE.24.002678

[48] Gu, C., Cong, Y., Sun, G., Gao, Y., Tang, X., Zhang, T., Fan, B. (2021). MedUCC: Medium-driven underwater camera calibration for refractive 3-D reconstruction. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52(9): 5937-5948. https://doi.org/10.1109/TSMC.2021.3132146

[49] Zhao, X., Yin, J., Chen, Z., Lu, X. (2013). Distance-aware virtual cluster performance optimization: A hadoop case study. In 2013 IEEE International Conference on Cluster Computing (CLUSTER), Indianapolis, IN, USA, pp. 1-8. https://doi.org/10.1109/CLUSTER.2013.6702618

[50] Saeed, Z., Yousaf, M.H., Ahmed, R., Velastin, S.A., Viriri, S. (2023). On-board small-scale object detection for unmanned aerial vehicles (UAVs). Drones, 7(5): 310. https://doi.org/10.3390/drones7050310

[51] Singhal, C., Barick, S. (2022). ECMS: Energy-efficient collaborative multi-UAV surveillance system for inaccessible regions. IEEE Access, 10: 95876-95891. https://doi.org/10.1109/ACCESS.2022.3206375

[52] Teng, H., Ahmad, I., Msm, A., Chang, K. (2020). 3D optimal surveillance trajectory planning for multiple UAVs by using particle swarm optimization with surveillance area priority. IEEE Access, 8: 86316-86327. https://doi.org/10.1109/ACCESS.2020.2992217