Анализ данных

Data science

УДК 004.932

СИСТЕМА ДЕТЕКТИРОВАНИЯ ДЫМА И САБОТАЖА НА БАЗЕ ОДНОПЛАТНОГО КОМПЬЮТЕРА ДЛЯ МОНИТОРИНГА ПОЖАРОВ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

Е. Р. АДАМОВСКИЙ[1)]***, Р. П. БОГУШ***[1)]

[1)]*Полоцкий государственный университет им. Евфросинии Полоцкой, ул. Блохина, 29, 211440, г. Новополоцк, Беларусь*

Аннотация. Представлена архитектура системы видеонаблюдения с использованием периферийных вычислений для обнаружения дыма и саботажа. Система состоит из параллельно функционирующих модулей, отвечающих за управление конфигурацией и обработку видеоданных, включая их прием, анализ и передачу. Детектирование дыма осуществляется с помощью алгоритмов компьютерного зрения и базируется на последовательном извлечении его визуальных признаков: направления движения, цветовых характеристик и влияния на фоновые объекты. Саботаж обнаруживается путем оценки резкости сцены с применением фильтра для выделения границ объектов. Обработка видеопотока высокого разрешения выполняется с использованием одноплатного компьютера, при этом производится распределение задач по ядрам на многоядерном процессоре для повышения производительности. Такое решение существенно снижает нагрузку на центральный узел системы, особенно при масштабировании на большое число камер. Результаты экспериментов демонстрируют высокую точность обнаружения дыма и саботажа, а также стабильную работу архитектуры в режиме реального времени.

Ключевые слова: детектирование дыма; компьютерное зрение; системы видеонаблюдения; одноплатный компьютер.

Образец цитирования:
Адамовский ЕР, Богуш РП. Система детектирования дыма и саботажа на базе одноплатного компьютера для мониторинга пожаров в режиме реального времени. *Журнал Белорусского государственного университета. Математика. Информатика.* 2026;1:108–120 (на англ.).
EDN: WTXHNV

For citation:
Adamovskiy YR, Bohush RP. Smoke and camera tampering detection system based on single-board computer for real-time fire monitoring. *Journal of the Belarusian State University. Mathematics and Informatics.* 2026;1:108–120.
EDN: WTXHNV

Авторы:
Егор Русланович Адамовский – старший преподаватель кафедры вычислительных систем и сетей факультета информационных технологий.
Рихард Петрович Богуш – доктор технических наук, профессор; заведующий кафедрой вычислительных систем и сетей факультета информационных технологий.

Authors:
Yahor R. Adamovskiy, senior lecturer at the department of computing systems and networks, faculty of information technologies.
e.adamovsky@psu.by
https://orcid.org/0000-0003-1044-8741
Rykhard P. Bohush, doctor of science (engineering), full professor; head of the department of computing systems and networks, faculty of information technologies.
bogushr@mail.ru
https://orcid.org/0000-0002-6609-5810

SMOKE AND CAMERA TAMPERING DETECTION SYSTEM BASED ON SINGLE-BOARD COMPUTER FOR REAL-TIME FIRE MONITORING

Y. R. ADAMOVSKIY[a]***, R. P. BOHUSH***[a]

[a]*Euphrosyne Polotskaya State University of Polotsk,*
29 Blahina Street, Navapolack 211440, Belarus

Corresponding author: Y. R. Adamovskiy (e.adamovsky@psu.by)

Abstract. An edge-based architecture for a video surveillance system with integrated smoke and camera tampering detection is presented. The system consists of parallel-executing modules responsible for configuration management and video data processing, including receive, analysis, and transmission. Smoke detection is performed using computer vision algorithms based on the sequential extraction of discriminative visual features: motion direction, colour characteristics, and impact on background visibility. Camera tampering is detected by evaluating scene sharpness through an edge-detection filter. High-resolution video processing is performed on a single-board computer, with computational tasks distributed across the cores of a multi-core central processor to optimise performance. The proposed approach significantly reduces the load on the central node of the surveillance infrastructure, which is a critical advantage for scalable deployments involving numerous cameras. Experimental results demonstrate high detection accuracy for smoke and tampering, as well as stable real-time operation of the implemented architecture.

Keywords: smoke detection; computer vision; video surveillance systems; single-board computer.

Introduction

Fire detection is a critical task with a wide range of applications, including residential and industrial facilities, streets, and forested areas. In its early stage, a fire can be easily mitigated; however, its further development and spread may lead to significant damage. Therefore, monitoring the initial phases of a fire is of considerable interest. Computer vision algorithms that analyse video data represent a promising approach to early fire detection, including in outdoor environments. Smoke is often the dominant or even the only visible indicator of fire at an early stage [1]. It propagates faster than flames and occupies a comparatively larger volume of space because it does not require close contact with the high-temperature region of material combustion. Therefore, when developing an early-stage fire detection algorithm based on video frame analysis, it is reasonable to focus on smoke detection.

Visually, smoke is characterised by the following features [2]: vertical upward motion of particles that transitions into chaotic motion; a limited colour range in shades of gray; semi-transparency, which causes blurring of background objects; a quasi-stable source location; an irregular shape with flickering. These features can be extracted by analysing video frames obtained from a surveillance system in real time. However, video data is subject to various distortions and limitations, including limited information due to camera resolution, colour range, and frame rate; loss of scene depth; sensor noise; optical system effects; variable lighting conditions; compression artifacts introduced by lossy video codecs. Therefore, when developing a smoke detection algorithm, it is essential to exploit as many of these features as possible through joint spatial and temporal processing to achieve high detection accuracy (AC) and minimise errors.

When video surveillance systems with smoke detection capabilities are deployed at industrial facilities, it becomes necessary to rapidly identify anomalies that cause the video signal to lack relevant information about the monitored scene or about the signal itself, which may indicate tampering. Tampering refers to an intentional action aimed at disrupting the operation of a video surveillance system while preserving the appearance of normal functionality. Tampering may involve physical interference with cameras, for example by covering their lenses with opaque or translucent objects, painting over them, or overexposing the image sensor, which renders it impossible to capture intruder activity or detect visual smoke features. Algorithms for this task must account for complex conditions encountered in real-world operation [3]: the appearance of new objects in the scene that occupy a significant portion of the frame; camera motion induced by wind; variations in lighting, time of day, and weather conditions.

Video surveillance systems are typically large-scale infrastructures equipped with numerous cameras. Continuous monitoring of the proper operation of a large number of cameras over extended periods can be challenging for security personnel and operators. Consequently, there is a risk of significant material damage due to either a missed fire in its early stage or undetected tampering. The large number of surveillance devices results in insufficient computing resources for centralised real-time processing of the resulting data volumes. One possible solution is edge computing, which offloads computation to edge devices that operate in conjunction with video cameras and perform initial analysis and processing of the video stream.

Related works

The concept of an edge or embedded system for threat detection, including fire and camera tampering, involves processing the video stream directly on the camera or endpoint device, as such devices are often deployed in remote or hard-to-access locations. Therefore, the computing unit must be compact and energy-efficient, which inherently limits its computational capacity. Existing implementations typically employ single-board computers (SBC), such as the Raspberry Pi and NVIDIA Jetson, as the hardware platform.

Fire and camera tampering can be regarded as a complex threat detectable by computer vision techniques in video surveillance systems. Certain steps in smoke and camera tampering detection algorithms are similar, as both aim to extract informative features from video frames to determine the presence of a threat.

Smoke detection methods deployed in embedded systems are either adapted to operate on low-performance hardware or replaced with novel algorithms designed to minimise computational complexity. For instance, in work [4] the authors evaluate the use of artificial neural networks (ANN) for edge-based fire detection across several architectures, including VGG19, ResNet18, EfficientNetV2, Vision Transformer (ViT), and ConvNeXt, on an NVIDIA Jetson Xavier platform. Experimental results demonstrate that model compression techniques, such as quantisation, pruning, and knowledge distillation, enable these networks to achieve a frame-level classification AC of 95 % while maintaining processing rates above 30 fps. Nevertheless, the Jetson Xavier significantly outperforms and exceeds the cost of typical SBC, suggesting that deploying high-complexity ANN in resource-constrained edge environments remains impractical.

In article [5] the compact MobileNetV2 model was employed for smoke and flame detection in forest fire scenarios. The model operated on a Raspberry Pi 5 (further – RPi5), processing an input video stream at 30 fps. It was trained on a dataset comprising 40 000 images and optimised through quantisation, input cropping, and conversion to TensorFlow Lite format. The system achieved a processing speed of 1–2 fps, with an F1-score of 94 % on a held-out test set. To suppress false positives, the system generated a class prediction for the current frame based on the five preceding frames and transmitted the result to the server.

In work [6] a portable smoke detection system based on the Raspberry Pi 4B (further – RPi4B) was developed. It comprised a Raspberry Pi camera, a video frame processing module, a detector, and display and notification subsystems. The data processing module included a 5×5 Gaussian filter for noise reduction and histogram equalisation for contrast enhancement. Smoke detection was performed using a cascade AdaBoost classifier based on local binary pattern (LBP) features, extracted from each video frame via a sliding window approach. The classifier was trained on a dataset of 6800 image patches of size 35×35 and 48×48 px using an NVIDIA GTX 1080 graphics processing unit. The implementation was carried out in Python using the OpenCV computer vision library. The system processes video frames of 320×240 px at 25 fps, achieving an AC of 94.7 %. The reduced input resolution, resulting from LBP feature extraction, was imposed by hardware constraints. Output data was saved to files and accessed remotely, while smoke detection alerts were delivered via email. It should be noted that real-time operation in research [6] relied on low-resolution frames, which limits the effective detection of small objects. Moreover, temporal information that could have contributed to improved detection AC was not exploited.

In article [7] a multi-stage smoke detection algorithm is proposed, which includes the following steps: motion detection using a Kalman filter; saturation segmentation in the HSV colour space; combination of the resulting binary masks using the logical AND operator; identification of regions of interest (RoI) with a sufficiently large area and a specified aspect ratio range; classification of the regions using a lightweight convolutional neural network (CNN) with an input dimension of $50 \times 50 \times 3$ and a detection threshold of 0.7. The model was trained on a dataset of 14 000 images. To suppress false positives, smoke was considered detected only if its presence was confirmed continuously for at least 3 s. The implementation was developed in Python using the Keras machine learning framework. The algorithm was evaluated on NVIDIA Jetson Nano (JN) and Raspberry Pi 3 (further – RPi3) platforms, achieving an average AC of 87.7 % at processing speeds of 55 and 9 fps, respectively, with an input frame resolution of 320×240 px. A high false positive rate was observed, which may result from suboptimal alarm triggering duration or limited generalisation capability of the employed CNN. Furthermore, the algorithm's performance on the RPi3 is insufficient for real-time operation. Key features of the approach include incorporation of temporal context and preprocessing aimed at reducing data volume and computational load on the CNN.

A CNN was also employed for fire detection based on the YOLOv2 architecture and implemented on JN [8]. The implementation was developed in MATLAB using CUDA technology. The model was trained on a dataset of 400 images of size 128×128 px and achieved an F1-score of 95.4 % on test videos at a processing speed of 21 fps. Fires were detected 1–2 s after their actual occurrence. It should be noted that video frames were downscaled to extremely low resolution, yet real-time processing was not achieved. The algorithm incurs

significant computational costs and relies on proprietary CUDA technology developed by NVIDIA, which restricts its applicability to other SBC.

The YOLOv8 architecture was employed for fire detection with several adaptations for the Ingenic T41 (further – T41) low-power embedded system-on-chip: reparameterisation, replacement of the SiLU activation function with LeakyReLU to simplify model quantisation, pretraining on the COCO dataset, fine-tuning on the domain-specific D-Fire dataset comprising 21 500 images, and minimisation of data merging and splitting operations [9]. The trained model achieved 79.4 % mAP at a processing speed of 47 fps. Despite these strong results, it should be noted that the optimisations were tailored to a specific hardware architecture, and their effectiveness may diminish when deployed on other SBC.

Another approach using YOLO for forest fire detection with unmanned aerial vehicles is presented in work [10]. The YOLOv5 architecture was deployed on the RPi4B with the following modifications: replacement of the CNN in the backbone with a lightweight ShuffleNetV2 model, network thinning, and subsequent retraining. The model was trained on a dataset of 24 400 images of size 320 × 240 px using an NVIDIA RTX 3050 graphics processing unit. The CPU clock rate was increased by 33 %. The system achieved a processing speed of 9 fps with 92.5 % mAP. Comparison with [9] shows that the modifications in [10] were less platform-specific, yet delivered lower processing speed despite using smaller input resolution, even though the RPi4B offers superior computing power compared to the T41. This indicates that hardware-aware optimisations can significantly enhance algorithm performance in computer vision tasks.

An analysis of the YOLO and CNN-RCNN architectures for UAV-based fire detection on the JN was conducted in [11]. The YOLOv8n model was employed for object detection, while a hybrid CNN-RCNN model performed classification using temporal information. Training was carried out on a dataset of 2947 images of size 640 × 640 px. The system achieved a processing speed of 12–20 fps with an F1-score of 88 %. The results indicate that the use of a sequence of multiple ANN did not yield high performance; however, the approach demonstrated good AC in smoke detection.

In work [12] tampering detection is performed by evaluating the sharpness of object boundaries in images using a Canny edge detector. The results are analysed over time, and a Kalman filter is applied to suppress false alarms. The following tampering scenarios were considered: defocusing, spray-painting, occlusion, and camera redirection. The precision (P) of the proposed method was 96.6 %.

Video camera tampering detection is performed using deep neural networks in article [13]. The authors employ the AlexNet, ResNet18, ResNet50, and DenseNet161 architectures. AlexNet achieved the highest AC at 74.9 %, but generated over 2000 false alarms per hour, which is unacceptable for real-world deployment. The authors note that incorporating temporal context would improve the results. Moreover, the computational requirements of the considered neural networks preclude their real-time operation on SBC.

The analysis shows that existing algorithmic solutions for smoke and camera tampering detection are generally unsuitable for deployment on SBC, when high-resolution video must be processed in real time. Therefore, the smoke detection algorithms and models employed on these devices reduce the input data size, which improves performance but may lead to missed detections of small objects. The solutions discussed employ lightweight, trainable classifiers and modified ANN, the combination of which with classical computer vision algorithms and hardware-specific optimisations can improve the performance of smoke detection algorithms. To reduce false positives, some approaches accumulate extracted features over several seconds and base smoke detection decisions on the aggregated data. However, most methods based on ANN rely solely on information from the current video frame, thereby neglecting temporal context. Both rule-based and neural network-based methods can be employed for camera tampering detection. However, the use of ANN is limited by their high computational complexity. The limited computing power of SBC restricts the number of services that can run simultaneously, thereby preventing complex video analysis in addition to smoke detection. Therefore, developing an effective smoke and camera tampering detection system with high performance and low computational requirements for embedded video surveillance modules remains a pressing challenge.

Efficient real-time smoke and tampering detection video system

System architecture. The proposed edge-based system for smoke and camera tampering detection comprises server and client components. The server component includes a SBC, specifically a RPi4B, an IP camera, and a network switch. The client component consists of an operator workstation equipped with dedicated software, including video surveillance tools and a web application for interaction with the server. The block diagram of the system is shown in fig. 1.

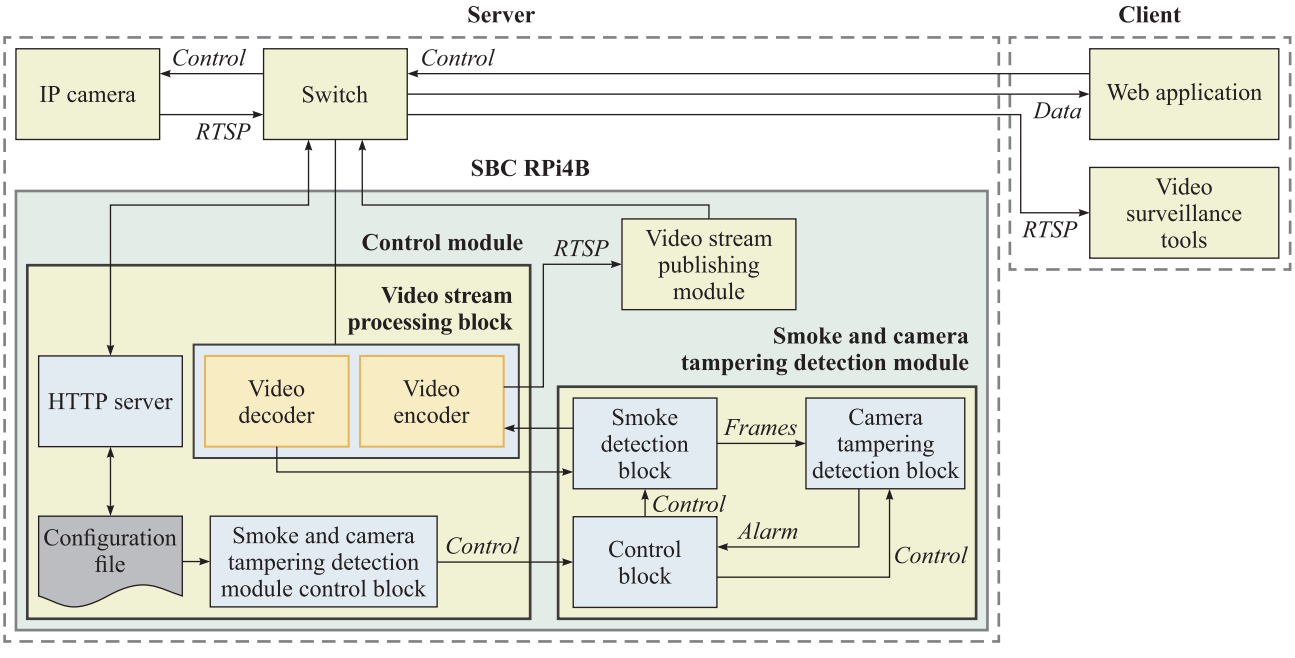

Fig. 1. Edge computing architecture for real-time smoke and camera tampering detection on a RPi4B

The server-side software stack deployed on the SBC comprises three core modules: a control module, a smoke and camera tampering detection module, and a video stream publishing module.

The functionality of the control module can be divided into interconnected blocks: a client web application control block, which is essentially a hypertext transfer protocol (HTTP) server, a smoke and camera tampering detection module control block, and a video stream processing block. The control module loads parameters from the configuration file, including general settings, operational parameters for the smoke and camera tampering detection module, and the real-time streaming protocol uniform resource locator of the video stream source. The web application control block processes HTTP requests and provides file server functionality. These requests include updating the configuration file, serving static web content, transmitting the current video frame, and delivering archived frames.

The smoke and camera tampering detection module control block launches the detectors, applies the configuration settings, and monitors their operation. The video stream processing block decodes the input video stream and forwards frames to the smoke and camera tampering detection module. It then receives the processed frames from this module, encodes them, and transmits them to the video stream publishing module. The smoke and camera tampering detection module comprises three blocks: a smoke detection block, a camera tampering detection block, and a control block that manages data exchange between the two.

The processed video stream publishing module makes the frames available for viewing on the operator workstation using standard video surveillance tools.

System operation algorithm. The proposed system operates using the developed algorithms for smoke and camera tampering detection. The smoke detection algorithm sequentially applies computer vision methods to the original video frames to extract discriminative smoke features. The processing order is determined by two factors: computational complexity and the discriminative power of the feature for smoke. If no features are detected at a given stage, subsequent stages are skipped, thereby reducing the computational load on the device. A key aspect of the approach is the division of video frames into a regular grid, with features averaged within each cell. This strategy simplifies and accelerates computation through uniform processing and parallelisation.

The camera tampering detection block, designed for real-time video processing on the SBC, exploits a characteristic property of video surveillance cameras: their minimum focusing distance is typically tens of centimetres or more, depending on the lens. Therefore, if the camera is obscured by a nearby object, such as a sheet of paper, image sharpness is reduced, even if the sheet reproduces the original scene. This reduction in sharpness can be detected automatically.

The system operation algorithm describes the modules and functional blocks, including the smoke and camera tampering detection techniques, and consists of the following steps.

1. The control module is launched. The module loads parameters from the configuration file. The HTTP server is enabled to support the client web application, and initiates periodic broadcast messages on the local network to allow the operator to discover the RPi4B and access the web application.

2. The smoke and camera tampering detection module is launched. Its control block loads the parameters from the configuration file obtained in step 1. In the event of an emergency shutdown of the detectors, the module automatically restarts. A restart is also triggered if the configuration file is updated via the HTTP server.

3. The video stream processing block receives an input video stream from a source, whose uniform resource locator is specified in the configuration file, decodes it, and transmits each frame I_t to the smoke and camera tampering detection module. If the input video stream becomes unavailable, the block generates a synthetic video stream containing an error message and notifies the web application control block.

4. The smoke and camera tampering detection module receives the frame I_t as input and launches detectors that process it according to the described algorithms.

4.1. Operating algorithm of the smoke detection block [14] within the smoke and camera tampering detection module.

4.1.1. The module is initialised in standby mode upon system startup.

4.1.2. Each frame I_t is resised to a fixed spatial dimension for further processing.

4.1.3. The histogram of I_t is normalised.

4.1.4. An initial background model B is constructed using the MOG2 algorithm [15].

4.1.5. The module transitions to operating mode after a background initialisation period T_b, required to construct the initial background model.

4.1.6. Cells C_m containing motion are extracted using background subtraction followed by morphological erosion and dilation as post-processing steps.

4.1.7. The background model B is updated, excluding cells where motion was detected in frame I_{t-1}. If smoke is detected, the update is suspended entirely.

4.1.8. Cells $C_{m.st}$ exhibiting stable motion over a time window T_m are selected.

4.1.9. Cells C_c, whose colour characteristics correspond to those of smoke, are selected.

4.1.9.1. Frame I_t is converted to the HSV colour space, yielding I_t^{HSV}.

4.1.9.2. Threshold binarisation is applied to I_t^{HSV}, S and V channels separately. The colour properties of smoke are more effectively captured in the HSV colour space, enabling robust segmentation [16].

4.1.9.3. Masks obtained in step 4.1.9.2 are combined using the logical AND operator to produce the final smoke candidate mask C_c.

4.1.10. Candidate cells C_1 of the primary RoI are computed by combining the results of steps 4.1.8 and 4.1.9 into a weight array. If a cell in the current iteration exhibits steady motion and matches smoke in colour, its weight in the array is set to the maximum value. Otherwise, its weight is decremented by a fixed step in each subsequent iteration until it reaches zero. A candidate cell is defined as any cell in the weight array with a non-zero value. Further processing is performed only on the I_t fragment that encompasses all current candidate cells within a bounding rectangle.

4.1.11. The selection of cells C_{hf}, for which a significant drop in the energy of high-frequency components relative to the background model B, is detected. For this purpose a two-dimensional wavelet transform is applied [17].

4.1.12. Cells $C_{hf.st}$ exhibiting a stable reduction in high-frequency energy over a time window T_{hf} are selected.

4.1.13. Selection of cells C_v, for which a sufficient contrast reduction relative to the background model, is detected.

4.1.14. Candidate cells C_2 for the secondary RoI are computed by applying the logical AND operator to C_1 and the results of the previous steps 4.1.12 and 4.1.13. Further processing is performed only for the fragment I_t that encompasses all candidate cells within a bounding rectangle.

4.1.15. Cells C_f exhibiting bottom-up motion are selected using the optical flow algorithm [18] applied to C_2 regions in frames I_t and I_{t-1}. A cell is selected if its motion vector lies within a specified angular range and exceeds a predefined amplitude threshold.

4.1.16. Cells $C_{f.st}$ exhibiting stable bottom-up motion over a time window T_f are retained.

4.1.17. Sets $C_{f.st}$ and C_2 are combined using the logical AND operator to produce C_t.

4.1.18. Cells C_a are selected by accumulating C_t over time and comparing the accumulated value with a presence decision threshold.

4.1.19. A decision on smoke presence is made if at least one non-zero element exists in C_a.

4.2. Operating algorithm of the camera tampering detection block: periodic analysis of video frames for tampering indicators during normal operation.

4.2.1. Frame I_t is converted to a grayscale image I_t^{bw}.

4.2.2. The Laplacian L_t of I_t^{bw} is computed by convolving the image with the Laplacian kernel. Each element of L_t represents the second-order spatial derivative at the corresponding pixel $I_t^{bw}(x, y)$.

4.2.3. The standard deviation σ for L_t is computed and used as a metric for image sharpness assessment.

4.2.4. Camera tampering in the frame is declared if $\sigma < \sigma_{th}$ is met, where σ_{th} is a predefined threshold.

4.2.5. The number of consecutive frames N with detected tampering is counted.

4.2.6. Camera tampering is confirmed if N exceeds the threshold N_{th}.

4.2.7. If camera tampering is not detected, then steps 4.2.1–4.2.6 are repeated, otherwise step 4.2.8 is executed.

4.2.8. The module enters tampering mode: smoke detection is suspended, and the output video stream is annotated with a tampering alert.

4.2.9. Steps 4.2.1–4.2.6 are repeated until camera tampering is no longer detected.

4.2.10. The control block halts the module. Subsequently, the control block of the smoke and camera tampering detection module restarts it, as described in step 2.

Multi-threaded real-time video processing architecture on quad-core embedded platform. The control module is implemented in the Rust programming language. The smoke and camera tampering detection module is implemented in C++ using the OpenCV library. The stream publishing module is implemented using the MediaMTX media server.

To improve performance, data processing steps are executed in parallel across the four CPU cores of the RPi4B with tasks distributed among dedicated threads to ensure efficient operation and real-time processing.

Thus, video stream transmission, encoding, and decoding are performed on CPU cores 0 and 1. On CPU core 2, thread 0 receives decoded frames in the smoke and camera tampering detection module and executes the tampering detection. Thread 1 is dedicated exclusively to motion detection using the MOG2 algorithm, as this operation is computationally intensive. On CPU core 3, thread 0 executes the remaining stages of the smoke detection algorithm, and thread 1 transmits processed frames back to the video stream processing module.

The block diagram of the proposed model implementation, reflecting the distribution of tasks across CPU cores, is shown in fig. 2.

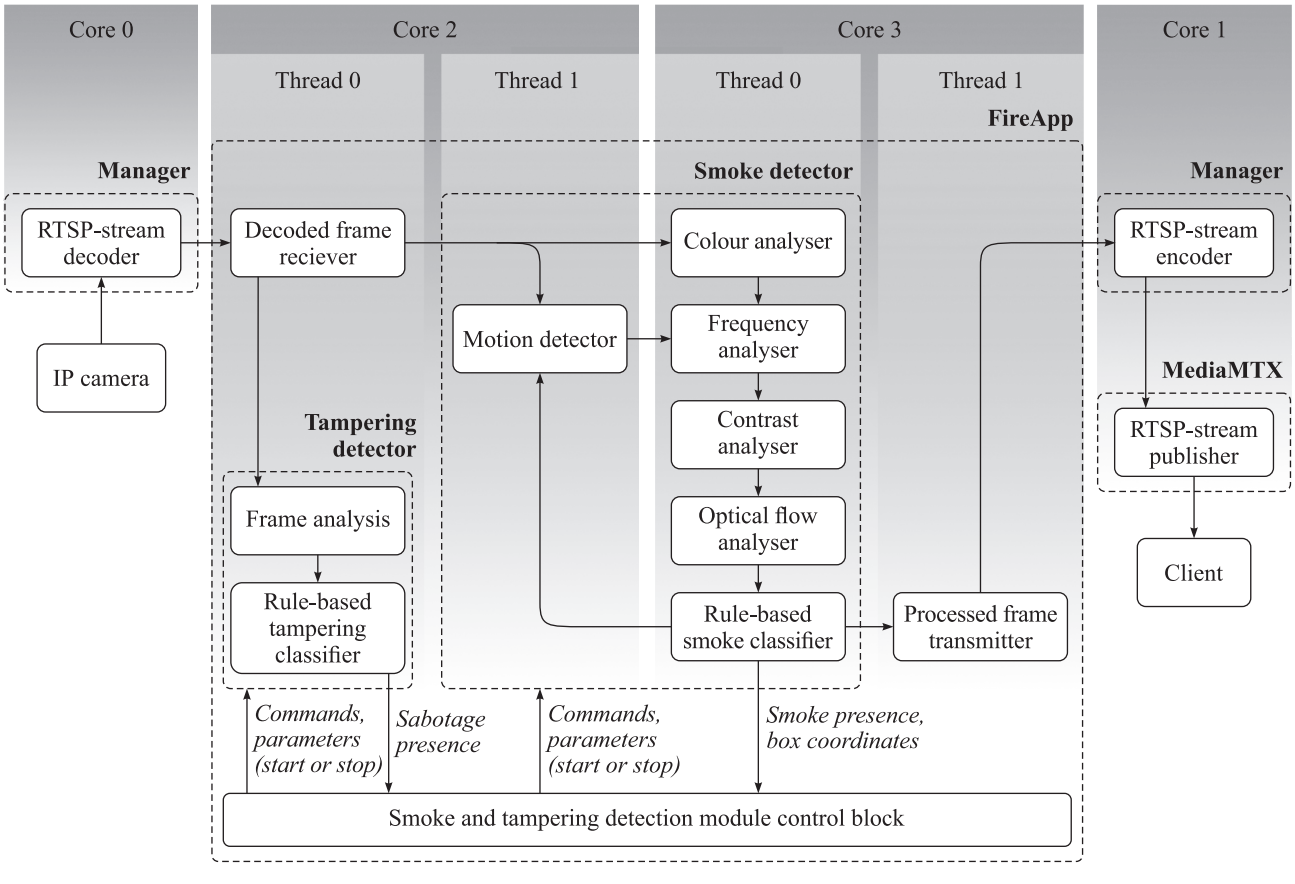

Fig. 2. Block diagram of the distribution of tasks in a model based on a 4-core CPU SBC

Experimental results

Performance evaluation of the smoke detection algorithm. To evaluate the performance of the algorithm, a dataset comprising video recordings of smoke and smoke-like phenomena at resolutions up to 1280 × 720 px (HD) was collected from open sources. Additional recordings were captured by the authors to supplement the dataset.

Experimental results show that the proposed smoke detection algorithm achieves an F1-score of 97.9 % and processes HD video streams at 25 fps. The use of spatiotemporal analysis and a combination of visual smoke features reduces the error rate to an average of 2 %, which is lower than that of comparable methods [2].

Table 1 presents the evaluation criteria for smoke detection methods, and the last row presents the results of testing the proposed algorithm. The compared metrics include AC, P, false-positive rate (FPR), false-negative rate (FNR) and F1-score [1]. Additionally, the table specifies the processed frame size, processing performance, and the SBC used.

Table 1

Comparison of smoke detection algorithms in edge-based systems

Method	Hard-ware	Frame size, px	FPS	Metrics, %					Source
				AC	FPR	FNR	F1-score	mAP	
MobileNetV2	RPi5	224 × 224	1–2	97.9	–	–	98.3	–	[5]
AdaBoost+	RPi4B	320 × 240	25	94.7	1.1	3.6	–	–	[6]
Rule-based+	RPi3	320 × 240	9	87.7	24.7	4.7	90.6	–	[7]
	JN	320 × 240	55						
YOLOv2	JN	128 × 128	21	96.8	3.4	2.9	95.4	–	[8]
YOLOv8n+	JN	640 × 640	20	89.0	6.5	2	88.0	–	[11]
YOLOv8	T41	672 × 416	47	–	–	–	–	79.4	[9]
YOLOv5	RPi4B	320 × 240	9	–	–	–	–	92.5	[10]
Proposed algorithm	RPi4B	1280 × 720	25	97.9	0.26	1.77	97.5	–	This work

Figure 3 shows examples of detection on video frames from a prepared dataset, where detected smoke regions are indicated by bounding boxes.

Fig. 3. Smoke detection results on video frames produced by the proposed algorithm

Performance evaluation of the camera tampering detection. The camera tampering detection module was evaluated using a custom dataset comprising video sequences of four scene types: daytime street, nighttime street, illuminated indoor environment, and semi-dark indoor environment (fig. 4).

The proposed method was compared with two other approaches based on image entropy estimation: the Shannon entropy calculation algorithm, which uses the distribution of pixel intensities represented as a histogram, and entropy estimation based on the compression ratio of the Lempel – Ziv – Markov chain algorithm (further – LZMA). The LZMA is deterministic, has efficient implementations in programming languages and is used in the 7z archive format. The LZMA compression ratio, defined as the ratio of the original data size to the compressed size, is inversely proportional to the entropy of the observed scene. Therefore, it provides a convenient and reproducible means for entropy estimation [19].

Four camera tampering scenarios were tested: covering the camera lens with a light-coloured object, covering the camera lens with a dark object, covering the camera lens with a translucent object, and illuminating the lens with a bright light source (fig. 5).

Fig. 4. Scene types for tampering detection evaluation:
a – daytime street; *b* – nighttime street;
c – illuminated indoor scene; *d* – semi-dark indoor scene

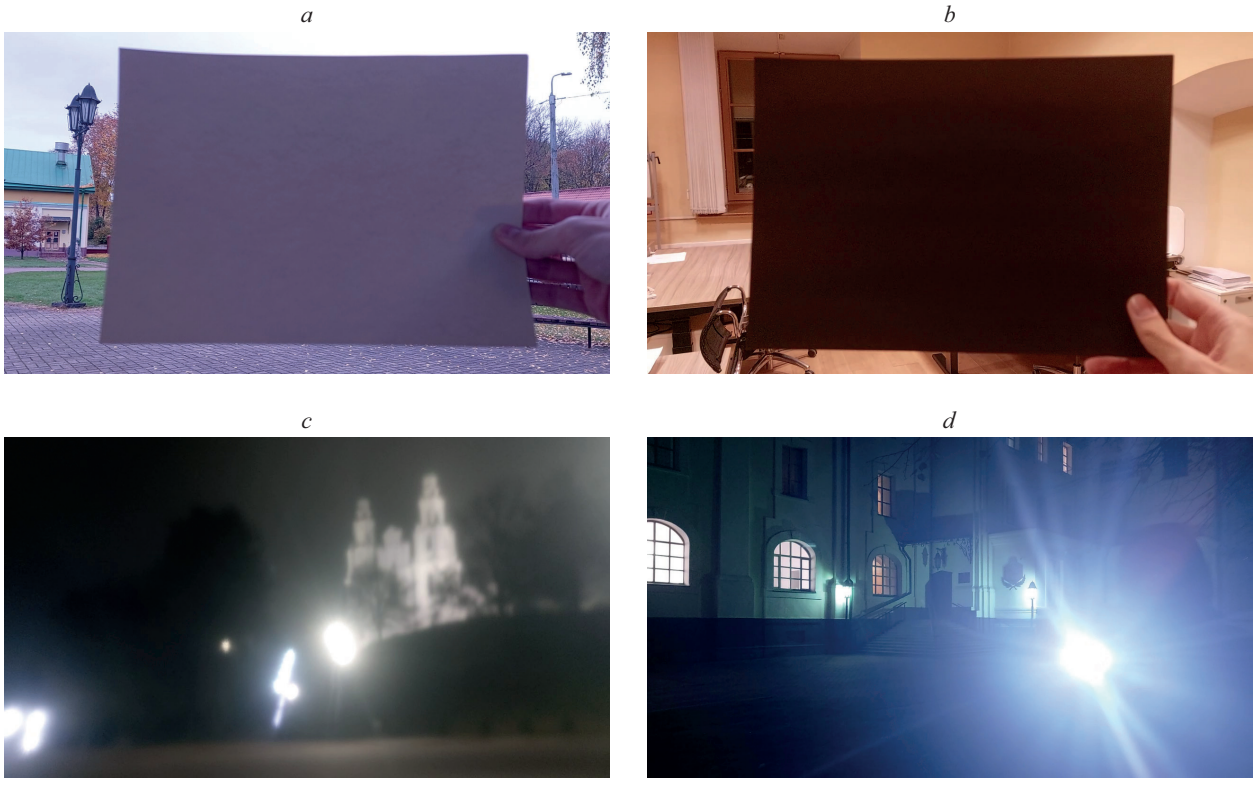

Fig. 5. Tampering scenarios for detector evaluation:
a – covering the camera lens with a light-coloured object;
b – covering the camera lens with a dark object;
c – covering the camera lens with a translucent object;
d – illuminating the lens with a bright light source

The obtained results are presented in table 2. The ranges of estimated values for individual test video recordings for different types of sabotage are shown.

Table 2

Estimated performance of camera tampering detection algorithms

Criterion	Scene (see fig. 4)	Reference estimated value	Estimated value in tampering scenarios (see fig. 5)			
			a	*b*	*c*	*d*
Sharpness assessment (σ)	*a*	39.20–57.70	22.50–24.70	19.90–24.20	17.10–19.20	–
	b	19.10–19.90	13.80–15.10	–	11.10–12.70	12.90–16.20
	c	25.0–26.6	16.20–17.60	15.70–16.60	13.0–14.0	17.20–20.50
	d	20.80–21.90	12.9–16.0	13.50–14.70	9.09–10.2	8.73–9.89
Shannon entropy estimate (*H*), bit/px	*a*	7.01–7.09	6.56–6.73	6.69–6.86	6.78–6.83	–
	b	5.91–5.93	3.96–4.28	–	6.50–6.55	7.52–7.71
	c	7.10–7.11	6.04–6.12	6.74–6.81	6.84–6.94	6.39–6.50
	d	5.51–5.54	5.57–5.66	2.74–3.32	5.79–5.97	6.80–7.26
LZMA-based entropy estimation (*k*)	*a*	2.46–2.57	4.68–4.94	5.02–5.57	8.28–8.85	–
	b	4.31–4.41	11.90–13.50	–	6.01–6.19	3.73–6.72
	c	4.67–4.83	8.41–9.12	5.28–5.74	8.38–9.94	5.51–8.31
	d	10.7–12.0	8.55–9.55	16.40–19.40	9.63–11.10	8.05–10.60

The reference estimated value in table 2 was obtained from video frames without features of camera tampering.

An analysis of table 2 shows that, for certain scenes, the outputs of algorithms 2 and 3 can be identical in the presence and absence of tampering. In contrast, the proposed tampering detection method, which is based on frame sharpness, enables the determination of a scene-specific threshold value σ_{th} for unambiguous classification of tampering presence or absence.

Thermal stability and real-time performance evaluation. The third and equally important stage of the research is to determine whether a SBC can operate within its thermal limits while maintaining real-time performance. The evaluation was performed under three load conditions of the smoke detection block.

1. No load (mode 1): the observed scene contains no smoke-like objects. Only the initial algorithm stages, namely motion detection and colour-based object detection, are active.

2. Partial load (mode 2): the observed scene contains smoke-like objects. Smoke regions are either absent or occupy a small portion of the frame for brief durations. All algorithm stages are periodically active, but detection events are infrequent and short-lived.

3. Full load (mode 3): the observed scene contains smoke that occupies a significant portion of the frame over an extended duration, representing the maximum computational load on all algorithm stages.

Video sequences corresponding to the described load conditions were selected from the dataset, and testing demonstrated that the system maintains a processing rate of 25 fps during extended operation.

The CPU temperature was recorded at 1-second intervals under ambient room temperature conditions, using a device housed in a case with a passive heatsink and no active cooling. The results were smoothed using a 60-sample moving average window to reduce noise and are presented in fig. 6.

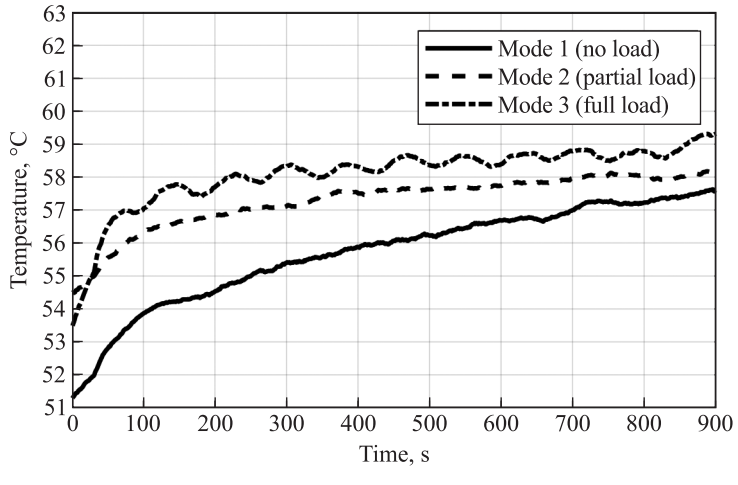

Fig. 6. RPi4B CPU temperature under various load conditions

Figure 6 shows that the initial CPU temperature is 50 °C. The peak temperatures observed during testing were 58.9 °C in mode 1, 59.4 °C in mode 2, and 60.4 °C in mode 3. The maximum permissible CPU temperature for the RPi4B is 80 °C, which is not exceeded in any load mode.

CPU core utilisation was also measured across all load modes with tampering detection disabled. The measurement data is presented in fig. 7 as histograms with core utilisation grouped into 10 % intervals. The height of each bar represents the number of measurements falling within the corresponding utilisation interval.

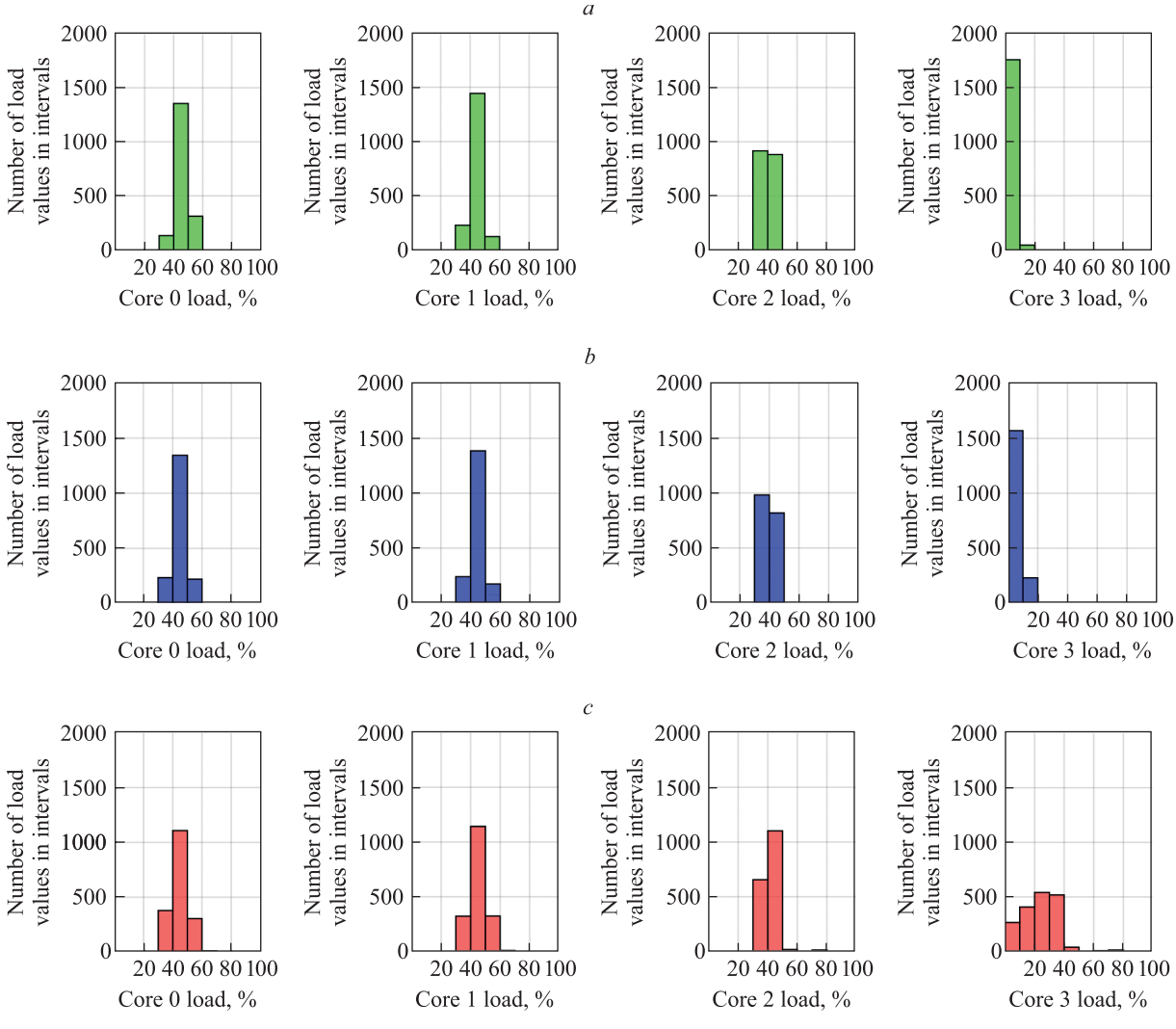

Fig. 7. RPi4B CPU core load: *a* – mode 1; *b* – mode 2; *c* – mode 3

A comparison of the data in fig. 7 shows that CPU cores 0 and 1 are consistently utilised at 30–60 %, regardless of the load mode, as they are not employed by the smoke detection algorithm. Core 2 is utilised at an average of 30–50 % and executes tampering and motion detection, which is the initial stage of the smoke detection algorithm and is applied to every frame. The utilisation of core 3 is the most mode-dependent, as it executes all subsequent stages of the smoke detection algorithm. Its load increases from 10–20 % in mode 1 to 40 % in mode 3. None of the cores reaches 100 % load, indicating a balanced distribution of tasks across the cores and ensuring stable system operation.

To assess the impact of the camera tampering detector on system performance, the CPU utilisation of core 2 was measured with the detector enabled and disabled. The results show that even in mode 3 (full load) core utilisation ranges from 31.2 to 48.6 %, which matches the range shown in fig. 7, *c*. Therefore, the tampering detector does not adversely affect the stability of the smoke detection algorithm or other system modules.

The experimental results demonstrate the effectiveness of the proposed smoke and camera tampering detection architecture. Analysis of the results confirms that both the algorithms and the overall system based on the proposed architecture, implemented on a SBC, perform robustly across a range of operating conditions.

Conclusions

This paper presents an edge-based architecture for a video surveillance system equipped with smoke and camera tampering detection capabilities, enabling real-time processing and analysis of high-resolution video streams on a RPi4B single-board computer. The system can be seamlessly integrated into existing video surveillance networks. By leveraging decentralised computing, the proposed approach significantly reduces the computational load on the central node, which is essential for scalable deployments involving a large number of cameras.

Experimental results confirm correct system operation, high detection AC for both smoke and tampering, and robust performance during continuous operation. The smoke detection algorithm achieves an AC of 97.9 % while processing high-quality video at 25 fps. The use of spatiotemporal analysis of smoke-related visual features reduces the total error rate to an average of 2.03 %, outperforming comparable state-of-the-art methods. Camera tampering detection based on image sharpness assessment via edge-based standard deviation enables reliable threshold selection without degrading the processing speed of the smoke detector or the overall system performance. Testing of the complete system demonstrated that the developed algorithms do not disrupt the thermal balance of the single-board computer, enabling deployment of the system in industrial applications.

References

1. Chaturvedi Sh, Khanna P, Ojha A. A survey on vision-based outdoor smoke detection techniques for environmental safety. *ISPRS Journal of Photogrammetry and Remote Sensing.* 2022;185:158–187. DOI: 10.1016/j.isprsjprs.2022.01.013.

2. Adamovskiy Y, Bohush R. Real-time algorithm for light gray smoke detection in video sequences. In: Shmaliy YS, editor. *8th International conference on computing, control and industrial engineering; 2024 June 21–22; Wuhan, China. Volume 1*. Singapore: Springer; 2024. p. 535–542 (Lecture notes in electrical engineering; volume 1252). DOI: 10.1007/978-981-97-6934-6_64.

3. Vennam P, Pramod TC, Thippeswamy BM, Kim Y-G, Kumar PBN. Attacks and preventive measures on video surveillance systems: a review. *Applied Sciences*. 2021;11(12):5571. DOI: 10.3390/app11125571.

4. Mahmoudi SA, Gloesener M, Benkedadra M, Lerat J-S. Edge AI system for real-time and explainable forest fire detection using compressed deep learning models. In: Bashford-Rogers T, Meneveaux D, Ammi M, Ziat M, Jänicke S, Purchase H, et al., editors. *Proceedings of the 20th International joint conference on computer vision, imaging and computer graphics theory and applications; 2025 February 26–28; Porto, Portugal. Volume 3.* Setubal: SciTePress; 2025. p. 847–854. DOI: 10.5220/0013382500003912.

5. Sharobiddinov D, Siddiqui HUR, Saleem AA, Mendez Mezquita G, Ramírez Vargas DL, de la Torre Díez I. Edge-based autonomous fire and smoke detection using MobileNetV2. *Sensors*. 2025;25(20):6419. DOI: 10.3390/s25206419.

6. Liu B, Sun B, Cheng P, Huang Y. An embedded portable lightweight platform for real-time early smoke detection. *Sensors*. 2022;22(12):4655. DOI: 10.3390/s22124655.

7. Gagliardi A, de Gioia F, Saponara S. A real-time video smoke detection algorithm based on Kalman filter and CNN. *Journal of Real-Time Image Processing.* 2021;18(6):2085–2095. DOI: 10.1007/s11554-021-01094-y.

8. Saponara S, Elhanashi A, Gagliardi A. Real-time video fire/smoke detection based on CNN in antifire surveillance systems. *Journal of Real-Time Image Processing.* 2021;18(3):889–900. DOI: 10.1007/s11554-020-01044-0.

9. Peng R, Cui C, Wu Y. Real-time fire detection algorithm on low-power endpoint device. *Journal of Real-Time Image Processing.* 2025;22(1):29. DOI: 10.1007/s11554-024-01605-7.

10. Ye J, Ioannou S, Nikolaou P, Raspopoulos M. CNN based real-time forest fire detection system for low-power embedded devices. In: Institute of Electrical and Electronics Engineers. *Proceedings of the 31st Mediterranean conference on control and automation; 2023 June 26–28; Limassol, Cyprus.* Piscataway: Institute of Electrical and Electronics Engineers; 2023. p. 137–143. DOI: 10.1109/MED59994.2023.10185692.

11. Shamta I, Demir BE. Development of a deep learning-based surveillance system for forest fire detection and monitoring using UAV. *PLoS ONE.* 2024;19(3):0299058. DOI: 10.1371/journal.pone.0299058.

12. Wang Y-K, Fan C-T, Cheng YK, Deng PS. Real-time camera anomaly detection for real-world video surveillance. In: Institute of Electrical and Electronics Engineers. *Proceedings of the 2011 International conference on machine learning and cybernetics; 2011 July 10–13; Guilin, China. Volume 4.* Piscataway: Institute of Electrical and Electronics Engineers; 2011. p. 1520–1525. DOI: 10.1109/ICMLC.2011.6017032.

13. Mantini P, Shah SK. UHCTD: a comprehensive dataset for camera tampering detection. In: Institute of Electrical and Electronics Engineers. *Proceedings of the 16th IEEE International conference on advanced video and signal based surveillance; 2019 September 18–21; Taipei, Taiwan.* Piscataway: Institute of Electrical and Electronics Engineers; 2019. p. 1–8. DOI: 10.1109/AVSS.2019.8909856.

14. Ye S, Adamovskiy Y, Chen H, Bohush R, Ablameyko S. Real-time smoke detection in video based on two-step selection of regions of interest and directional movement analysis. *Pattern Recognition and Image Analysis.* 2025;34:1323–1332. DOI: 10.1134/S1054661824701396.

15. Naimeng C, Wanjun Y, Xiaoyu W. Smoke detection for early forest fire in aerial photography based on GMM background and wavelet energy. In: Institute of Electrical and Electronics Engineers. *Proceedings of the 2021 IEEE International conference on cower electronics, computer applications; 2021 January 22–24; Shenyang, China.* Piscataway: Institute of Electrical and Electronics Engineers; 2021. p. 763–765. DOI: 10.1109/ICPECA51329.2021.9362647.

16. Fan X, Lei F, Yang K. Real-time detection of smoke and fire in the wild using unmanned aerial vehicle remote sensing imagery. *Forests.* 2025;16(2):201. DOI: 10.3390/f16020201.

17. Ye S, Bai Z, Chen H, Bohush R, Ablameyko S. An effective algorithm to detect both smoke and flame using color and wavelet analysis. *Pattern Recognition and Image Analysis*. 2017;27(1):131–138. DOI: 10.1134/S1054661817010138.

18. Wu Y, Chen M, Wo Y, Han G. Video smoke detection based on dense optical flow and convolutional neural network. *Multimedia Tools and Applications*. 2021;80(6):35887–35901. DOI: 10.1007/s11042-020-09870-x.

19. Challapalli K, Kothamasu V, Garikapati H, Charan RS, Thota M, Anamalamudi S. Analysing lossless image compression techniques for IoT devices. In: Institute of Electrical and Electronics Engineers. *Proceedings of the 2024 Third International conference on electrical, electronics, information and communication technologies; 2024 July 24–26; Trichirappalli, India.* Piscataway: Institute of Electrical and Electronics Engineers; 2024. p. 1–6. DOI: 10.1109/ICEEICT61591.2024.10718516.

Received 24.12.2025 / revised 19.01.2026 / accepted 29.01.2026.