energies

MDPI

Article

AdViSED: Advanced Video SmokE Detection for Real-Time Measurements in Antifire Indoor and Outdoor Systems

Alessio Gagliardi and Sergio Saponara *

Department of Information Engineering, University of Pisa, Via G. Caruso 16, 56127 Pisa, Italy; alessio.gagliardi@phd.unipi.it

* Correspondence: sergio.saponara@unipi.it

Received: 16 March 2020; Accepted: 20 April 2020; Published: 23 April 2020

Abstract: This paper proposes a video-based smoke detection technique for early warning in antifire surveillance systems. The algorithm is developed to detect the smoke behavior in a restricted video surveillance environment, both indoor (e.g., railway carriage, bus wagon, industrial plant, or home/office) or outdoor (e.g., storage area or parking area). The proposed technique exploits a Kalman estimator, color analysis, image segmentation, blob labeling, geometrical features analysis, and M of N decisor, in order to extract an alarm signal within a strict real-time deadline. This new technique requires just a few seconds to detect fire smoke, and it is 15 times faster compared to the requirements of fire-alarm standards for industrial or transport systems, e.g., the EN50155 standard for onboard train fire-alarm systems. Indeed, the EN50155 considers a response time of at least 60 s for onboard systems. The proposed technique has been tested and compared with state-of-art systems using the open access Firesense dataset developed as an output of a European FP7 project, including several fire/smoke indoor and outdoor scenes. There is an improvement of all the detection metrics (recall, accuracy, F1 score, precision, etc.) when comparing Advanced Video SmokE Detection (AdViSED) with other video-based antifire works recently proposed in literature. The proposed technique is flexible in terms of input camera type and frame size and rate and has been implemented on a low-cost embedded platform to develop a distributed antifire system accessible via web browser.

Keywords: IoT (Internet of Things); distributed smoke/fire alarm systems; embedded video processing; Kalman filter; industrial antifire system; mobility antifire system

1. Introduction

Recent reports of the NFPA (National Fire Protection Association) show that the average number of fires per year is about 1.3 million in the US alone, with a high cost in terms of lives (more than 3000 civilian fire fatalities) and economic losses—the cost for fire losses is estimated to be about 55 billion US Dollars (USD) per year [1]. Hence, with the advent of the Internet of Things (IoT) and the growing interest about safety in public places, an early fire-smoke detection system should be implemented for the benefit of all citizens. To this end, a video-based approach based on the recognition of fire smoke is a promising method. Indeed, the video signal is a wide monitor of the area under investigation and often closed-circuit television (CCTV) systems for surveillance purpose are already installed in smart buildings, in public places in smart cities, or onboard passenger vehicles of public transport systems. Exploiting an already existing video infrastructure allows the reduction of purchase/installation costs of additional add-on products, increasing only the complexity of the algorithm used to detect the smoke.

The smoke is the first event of a fire hazard since it appears before the flames. Standard smoke sensors, based on chemical, temperature, or PIR (Passive InfraRed) detectors [2–7], trigger an alarm

within several minutes when combustion is already producing flames and increasing the environmental temperature. In [8,9], a photoelectric smoke detector with an actual smoke chamber was mixed with smoke temperature. Current standards such as EN50155 for onboard train safety set a delay between the start of fire and its detection of 1 min. This is why the commercial solution in [10] for onboard train antifire systems uses point-based optical and temperature smoke detectors. These sensors aim at revealing the presence of a fire with a time delay within 1 min. The system in [10] exploits the fact that hot air produced by a fire moves from the bottom to the top of the train wagon, and hence the smoke moves toward the sensor placed on the roof. In [11], flames were identified by measuring absolute temperature and its gradient through a smart IR sensor. The detection systems presented have the drawback of reacting slowly. Moreover, they need active fans or air-conditioning to speed-up the smoke/fire detection process, thus avoiding a too high measuring latency. Instead, the aim of this work is to develop an innovative video-based smoke detection technique able to trigger the alarm within a few seconds. Such an algorithm has been already implemented into several IoT embedded devices to develop a distributed antifire system accessible via web browser and able to signal a fire alarm from different camera nodes, discussed in [12]. With this paper, the authors intend to extend that discussion by focusing more on the smoke detection algorithm aspects.

Hereafter, the paper is organized as follows. Section 2 deals with a detailed review of video-based smoke/fire measurements. Section 3 presents the new Advanced Video SmokE Detection (AdViSED) technique discussing the global architecture and then each of the algorithms used in the multiple video processing steps. Section 4 discusses the AdViSED thresholds configuration. Section 5 presents the evaluation of the complexity of the algorithm and the performance compared with the state of art systems. Implementation results of AdViSED in x86-based and ARM (Advanced RISC Machine)-based platforms are analyzed in Section 6. Conclusions are drawn in Section 7.

2. State of the Art Review of Video-Based Smoke Detection Algorithm

Video-based fire measuring and warning systems may help to reduce the detection time compared to other available sensors (smoke chemicals or temperature sensors) in both indoor and outdoor scenarios. A video camera can monitor "volumes" without transport delays that traditional "point" sensors suffer from. Particularly, CCTV cameras are suitable for fire detection within 100 m, which is the case of fire onboard passenger vehicles or in houses, offices, factories, or cities. Instead, for wildfire in forests or rural environments [13], which is out of the scope of this work, other techniques must be used, optimized for scenes observed at distances of several km. Many studies have been recently proposed in literature for video smoke detection [14–25].

In [24], a commercial solution was proposed exploiting both a 160 × 120-pixel, 9 fps, long wave infrared camera for thermal imaging, and a 1920 × 1080-pixel color CMOS (Complementary Metal Oxide Semiconductor) camera to exploit the thermal energy emitted from the objects and to analyze their dynamic features. Hence, absolute and gradient temperatures plus size and motion of the objects were analyzed by the system in [24]. However, the purchase and installation of such a solution is too expensive for exhaustive use in smart cities, homes/offices, and intelligent transport systems. In [25], a CCD (charge-coupled device) camera was used but only as a temperature sensor. The system in [25] was used to estimate the edges of a combustion flame and not as a fast fire/smoke detector. The authors in [13] implemented a background subtraction using frame differencing and working with the smoke features to discover the smoke areas. In the same direction is the work of O. Barnich and M. Van Droogenbroeck, which in [14] generates the background subtraction using Visual Background Estimation (ViBe). The ViBe technique was used also in [15] in combination with the dissipation function to estimate the background. Another smoke recognition was studied in [16], combining fuzzy logic, Gaussian Mixed Model (GMM), and Support Vector Machine (SVM) for recognition. In [19], the background and the foreground were estimated with the use of the Kalman filter. The work in [20] proposed an early fire detection system using Internet Protocol (IP) camera technology with Motion JPEG codec with the Discrete Cosine Transform (DCT) operating in the frequency domain for

smoke features. Instead, the work in [21] focused the algorithm on the appearance-based method and then a background subtraction was applied. In [26], edge detection image processing was used to reveal the edge of flames, rather than implementing a fire warning algorithm. In [27], a multiexpert decision algorithm was used for fire detection: combining the results of three expert systems exploiting motion, shape, and color information, which achieved a detection accuracy higher than 90% over the testing dataset. However, the complexity of the proposed technique in [27] limits the real-time algorithmic execution on a Raspberry Pi embedded system to only 3 fps and using only low-resolution images, 320 × 240 pixels. A convolutional neural networks (CNN) approach for video-based antifire surveillance systems was proposed in [18,28], but also in this case the implementation complexity, although reduced compared to other CNN-based techniques, is still too high. For example, [28] required a system equipped with a NVidia GeForce GTX TITAN X with 12 GB onboard memory and deep learning framework and Intel Core i5 CPU with 64 GB RAM. Such a platform has a cost in the order of hundreds of USD with a power consumption of hundreds of Watts. A hybrid computing architecture combining a GPU and a general-purpose processor has also been adopted in [29]. Here, the algorithms exploited motion estimation based on background subtraction and color probability analysis to detect candidate smoke blobs. Some of the techniques seen above [14–17,21,29–31] perform basically a background subtraction, so they just want to recognize the foreground. Other papers, like [19], implement more sophisticated video processing techniques, like Kalman based motion estimation, but with fixed parameters not aware of the specific camera configuration. Techniques such as in [18,28], based on CNNs, are difficult to implement in real scenarios due to the lack of a large dataset for the training videos. This work also outperforms a preliminary algorithm, studied and presented by the authors in [22,23], which was using SAD (sum of absolute differences) block matching as motion estimation, plus other image processing tasks, such as morphological filter, bounding boxes extraction, features extraction, correct bounding boxes selection, and space and time analysis.

The new approach, discussed in Sections 3 and 4, refines the motion estimation method by replacing it with a Kalman estimator, properly tuned for smoke detection. After color analysis, image segmentation, blob labeling, features analysis, and M out of N decisor, an alarm signal is generated. When comparing the new algorithm with that in [22,23], it can be observed that there is an increase of all the evaluation metrics, such as recall, accuracy, F1 score, precision, and MCC (Matthews correlation coefficient) and a reduction of the algorithm implementation complexity that allows implementing the new measuring system in real-time and with low power.

3. AdViSED Fast Measuring System

Figure 1 shows the logical flow of the proposed video smoke detection algorithm, which is based on motion detection, segmentation, features extraction, and elaboration of the video frames. The algorithm is designed as a chain of image and video processing tasks. The sequence of elaborations ends with computation of a real-time alarm signal.

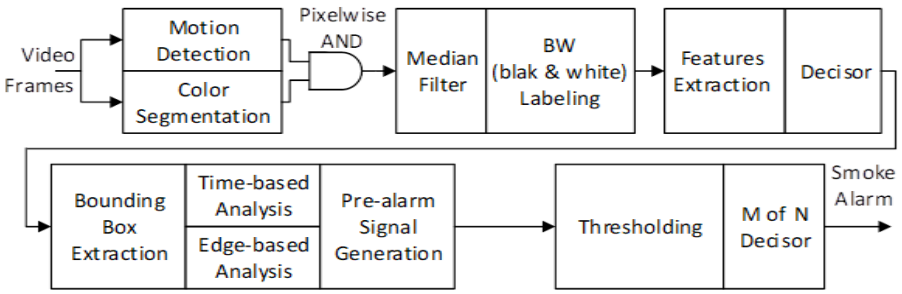


Figure 1. Workflow of the Advanced Video SmokE Detection (AdViSED) smoke detection algorithm.

A binary decision-maker for the fire alarm is implemented using a thresholding technique combined with the M of N classifier over the real-time alarm signal. When implementing the signal

processing chain of Figure 1 in a computing platform, the tasks motion-detection/color-segmentation and the tasks time/edge-based analysis can be parallelized, while the rest of the functions have to be calculated according to a sequential flow. Hereafter, the main video processing steps of the workflow in Figure 1 are detailed.

3.1. Motion Detection

A Kalman filter is used and a simplification of the theory regarding this filter is reported, where the information is obtained through an estimate and the prediction of the background. The background prediction is given by Equation (1), where $\widetilde{BG}_k$ is the background prediction of the current frame I, $\hat{BG}_{k-1}$ is the background estimation at the previous frame, and $a = A/(1-\beta)$ with $\beta = 1/\left(1+\tau_\beta \cdot fr\right)$. The fr coefficient represents the frame rate in fps of the processed video, τ_β is a time constant set to 10s, and A is a constant set to 0.618.

$$\begin{bmatrix} \widetilde{BG}_k \\ \dot{\widetilde{BG}}_k \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & a \end{bmatrix} \cdot \begin{bmatrix} \hat{BG}_{k-1} \\ \dot{\hat{BG}}_{k-1} \end{bmatrix} \tag{1}$$

The background estimation $\hat{BG}_k$ of the frame I is obtained from Equation (2), where $\widetilde{BG}_k$ is known in Equation (1), and K_1 and K_2 are defined in Equation (3).

$$\begin{bmatrix} \hat{BG}_k \\ \dot{\hat{BG}}_k \end{bmatrix} = \begin{bmatrix} \widetilde{BG}_k \\ \dot{\widetilde{BG}}_k \end{bmatrix} + \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} \cdot \left(I - \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \widetilde{BG}_k \\ \dot{\widetilde{BG}}_k \end{bmatrix} \right) \tag{2}$$

$$K_1 = K_2 = \Lambda \cdot FG_k + \beta \cdot (1 - FG_k) \tag{3}$$

$$FG_k = \left| I - \hat{BG}_{k-1} \right| \tag{4}$$

The above formulas work at pixel level. During the initialization, we set $\widetilde{BG}_k$ equal to initial frame I and $\dot{\widetilde{BG}}_k$ equal to zero. Obviously, the initialization is performed the first time when the first frame is received. According to Figure 1, the pixel-wise logic AND with the color segmentation is activated by the motion-estimation block only when the pixel of the foreground FG_k is higher than a proper threshold THR_{foreg}. In the above equations, FG_k is the foreground of the frame I, $\Lambda = 1/(1+\tau_\alpha \cdot fr)$, where τ_α is a time constant set to 16 s. The empirical threshold THR_{foreg} is set to 0.08. It is noted that the Kalman estimator proposed in [19] has the variable $a = 0.7$ fixed, while our variable $a = A/(1-\beta)$ depends on β, which in turn depends on constants A and τ_β and on the frame rate fr. In the proposed algorithm, the frame rate is extracted from the input processed video, so the variable a and hence the matrix in Equation (1) depend on the value of the video frame rate. It is noted that in the test videos considered in this work, fr ranges from 10 to 60 fps. The coefficient $K_1 = K_2$ depends on the coefficients Λ and β, selected to allow filtering out quickly in the background the objects that are faster than the smoke (like people in movements) and to filter out in the background static objects that are slower than the smoke. The value of A = 0.618 has been found, in case of high frame rate values (with $fr = 60$, then $\beta \sim 0$ and $a \sim 0.618$), starting from the value of 0.7 proposed in [19] and refining it to maximize the accuracy performance with the test video set considered in Sections 4 and 5. The values of τ_α and τ_β influence the values of Λ and β and determine the observation times of foreground and background scenes. Because we aim at an early detection of smoke, their value is limited in the order of few seconds (i.e., 300 frames observed with a typical $fr = 30$ fps). Empirical measurements with the test video set considered in Sections 4 and 5 prove that the detection accuracy is maximized with $\tau_\alpha = 1.6\ \tau_\beta$ and hence $\tau_\alpha = 10s$ and $\tau_\beta = 16s$. The proposed values for time constants τ_α and τ_β, to filter out objects moving faster than smoke or objects that are static or moving slower than smoked, are also in-line with experiments carried out in literature [32].

3.2. Color Segmentation and Blob Labeling

In this phase, the RGB color frames are converted in gray scale and next in HSV (hue, saturation, value) scale, where H is the hue, S is the saturation, and V is the value. We use only the saturation of the frames and through a saturation threshold, THR_{sat} (set to 0.2 in the range from 0 to 1), we select only those parts of the scene with $frame_{sat} < THR_{sat}$. We choose this parameter because the smoke changes color according to the background, so a good way is to use the saturation variable. The value of the threshold 0.2 is found empirically as the value maximizing the estimation accuracy using the dataset detailed in Sections 4 and 5.

The output pixels from motion detection (the foreground pixels) and color segmentation go inside a logic AND (pixel-wise). After this phase, the output logic mask is filtered with a median filter to remove the isolated pixels, considered as noisy pixels. We also do a labeling of the agglomerations of the pixels, called "blob". This labeling is used to extract the region properties in order to decide in the next sub-block which blobs have the right characteristics to pass the thresholds.

3.3. Color Segmentation and Blob Labeling

After motion estimation, color segmentation, and blob labeling, the following features of the blobs are extracted inside the feature-extraction block of Figure 1:

- Area is the number of pixels in the region studied.
- Turbulence is calculated as $Perimeter^2_{blob}/Area_{blob}$.
- Extent is the ratio between the number of pixels in the region and in the bounding box.
- Convex area is the region inside a bounding box delimited from a polygon without concaved angles; the corners are defined from the most external pixels.
- Eccentricity is the eccentricity of the ellipse of the bounding box.

The set of thresholds used to select the correct blobs is reported hereafter. We chose these thresholds by doing a trade-off between the video characteristics.

$$THR_{area} = W{\cdot}H/2000; \quad THR_{turb} = 55; \quad THR_{ext\_min} = 0.2$$
$$THR_{ext\_max} = 0.95; \quad THR_{eccentr} = 0.99; \quad THR_{convArea} = 0.5$$

Now after setting the thresholds, for each blob we set the ranges in order to decide which blob can survive or not to the next analysis, see the following rules:

$$Area_{blob} > THR_{area}; \quad Turbulence_{blob} < THR_{turb};$$
$$THR_{ext\_min} < Ext_{pixels} < THR_{ext\_max};$$
$$ConvArea_{blob} > THR_{convArea}; \quad Eccent_{blob} < THR_{eccentr}$$

The turbulence factor we introduce is an amplification of the Boundary Area Roughness (BAR) factor used in other works in literature [32]. The turbulence index is determined by relating the perimeter of the region to the square root of the area, and hence it is the squared version of the BAR index. For example, the turbulence index is 4π for a circle, 16 for a square, and about 20 for a triangle with sides L, L, and L·sqrt (2). Hence, the heuristic value of $THR_{turb} = 55$ used in this work for smoke blobs is also theoretically justified by being 3 to 4 times higher than that of "regular" figures like a circle. The area threshold THR_{area} is calculated so that for a VGA frame (W = 640 × H = 480) its value is about 150 pixels, which is in line with the size of about 100 pixels (10 × 10) considered as typical in literature [32] for video-based detection systems with an observed area within a distance of 100 m.

It is worth noting that some papers listed in the introduction use some static smoke features. For example, [14] used the distance between the pixels and the clusters, [17] used the thermal turbulence and diffuse and the complexity of the edges, and [20] used the expansion direction of smoke regions. Different from state-of-art systems, the algorithm proposed in this work does not use just a single

feature, but it uses together several geometric features like area, turbulence, extent, convex area, and eccentricity. This way, considering more parameters, the proposed AdViSED technique can reach better performance in terms of accuracy of the alarm prediction; see Section 5.

3.4. Bounding Box Extraction and Time/Edge Analysis

After the feature extraction and first decision step discussed above, the survivor blobs are analyzed dynamically (time-based analysis step in Figure 1) considering the characteristic of the bounding box that contains them. The bounding box table is composed by the lists of all bounding boxes and for each of these we have the dimension and origins x and y and a counter called kill time. The kill time is initialized to the same value for each box (i.e., $kill\_time = frame\_rate$). The counter is decreased by one for each frame thereafter analyzed. When the counter is zero, the bounding box is deleted. This means that each bounding box has a lifetime of one second by setting the value of $kill\_time = 30$ and considering a $frame\_rate$ of 30 fps.

There is also a test in parallel to the count to kill the smoke blobs without marked edges (edge-based analysis step in Figure 1). Therefore, we perform an edge test that deletes the bounding boxes that do not respect the constraint without waiting for the end of the count of the kill time.

To implement this task, we use the Sobel edge detection method in Equation (5). With I being the source image, G_x and G_y represent the vertical and horizontal derivative approximation, where $*$ denotes the two-dimensional signal processing convolution operation. The resulting gradient approximation can be combined to give the gradient magnitude using Equation (6).

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * I; \; G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * I; \tag{5}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{6}$$

3.5. Pre-Alarm Signal and Smoke Alarm

At runtime, if the smoke is present in the scene caught on camera, it is natural to think that multiple boxes are overlapped. Therefore, we want to count the number of overlaps. The algorithm counts for each pixel how many bounding boxes it belongs to. An index called overlap takes the maximum of these values, and this overlap index is used to generate the final smoke alarm.

A threshold is used to generate a binary signal from the prealarm, also known as overlap index. We chose $THR_{overlap} = 7$, so we need at least seven bounding boxes overlapped to generate a logical one in this comparison. To activate the smoke alarm, we want the signal to be greater than the threshold for some seconds in a sliding temporal window. The selected time window has a size of 3 s, and the overlap index must exceed the threshold for at least 1.5 s inside the window. If this happens, the smoke alarm is generated. The implementation of the latter is done implementing the M of N decision algorithm, where M is the number of detected smoke frames in 1.5 s and N is the number of frames in 3 s.

3.6. Pre-Alarm Signal and Smoke Alarm

The proposed technique has been tested with a large set of test videos from 320×240 to 1920×1440 pixels/frame, while the frame rate is between 10 fps and 60 fps (including all intermediate formats usually present in the state-of-art systems). Hereafter, we show the pictures of the output processed videos for three outdoor cases (Figures 2i and 3a,c) and one indoor case (Figure 3b) representing both the case of true positive (Figures 2i and 3b) and negative (Figure 3a,c) conditions. Purple boxes in the figures refer to detected blobs. The alarm is only generated when the red circle appears on the top left of the image (Figures 2i and 3b); otherwise the circle displayed is green.

Figure 2 is an example application of a smart city antifire system showing the output mask of each processing step already described. Figure 3a is related to an intelligent transportation system with antifire safety features. Figure 3b is an example of an active antifire home alarm. Figure 3c is an example of a forest fire prevention.

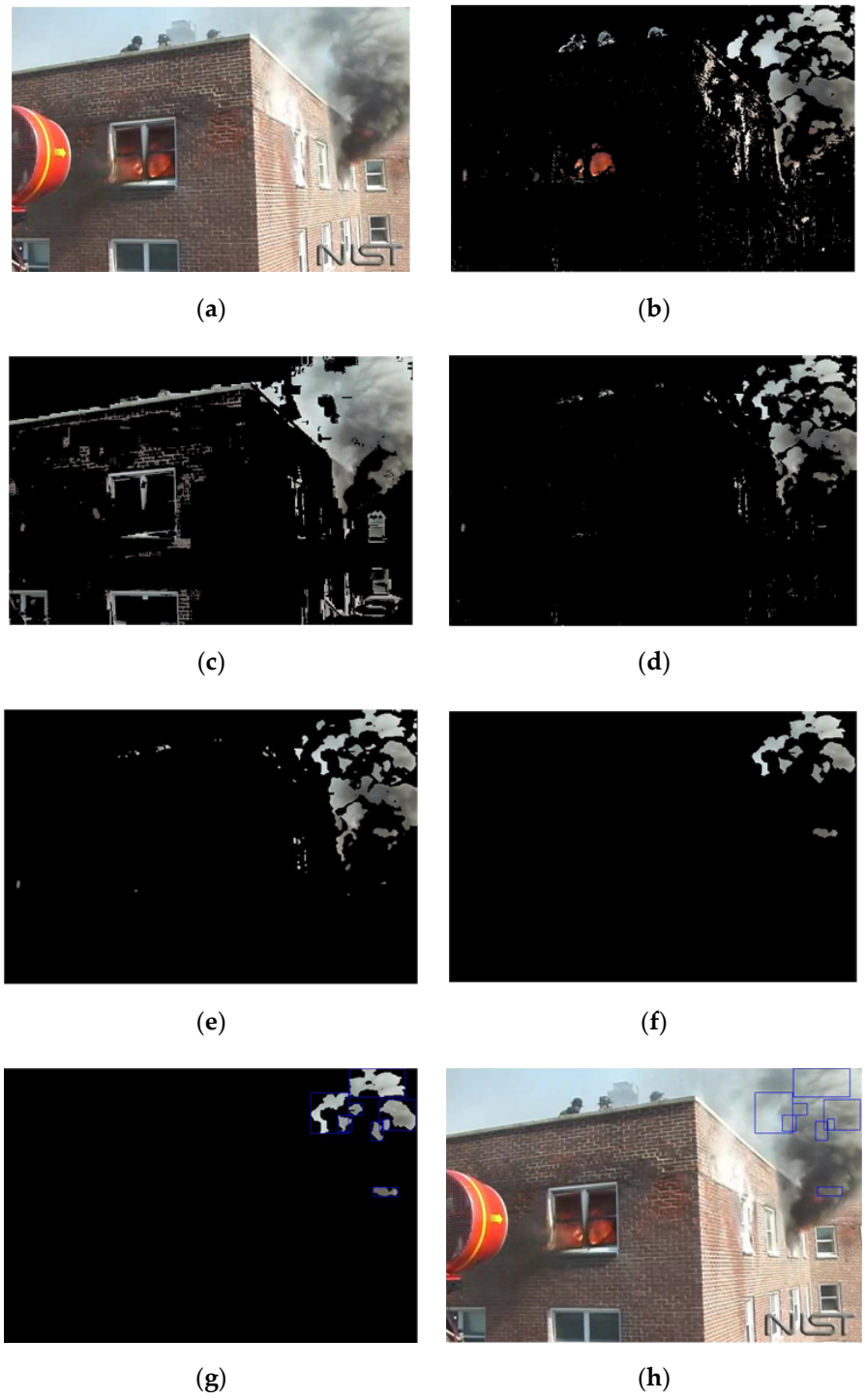
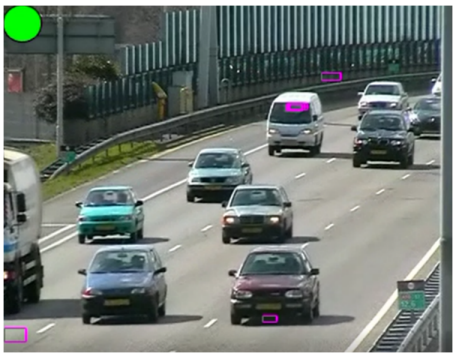


Figure 2. *Cont.*

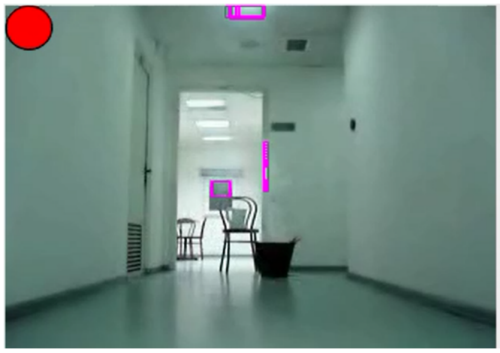
(**i**)

Figure 2. Detection of smoke on a test video. (**a**) Current Frame n; (**b**) motion detection; (**c**) color segmentation; (**d**) logic AND pixelwise; (**e**) median filter; (**f**) feature extraction; (**g**) bounding box extractor; (**h**) bounding boxes applied on the current frame n; and (**i**) final result at runtime with many overlaps and smoke alarm activated.

In the elaboration of Figures 2i and 3b, the smoke alarm is activated while in Figure 3a,c the smoke alarm is not activated. This algorithm, different from [14,16–18,20,21], surrounds the smoke blobs and examines their bounding boxes evolution. This way, AdViSED can reach better performance in terms of accuracy of the alarm prediction.



(**a**)



(**b**)

(**c**)

Figure 3. (**a**) Outdoor transportation scenarios with few bounding box overlaps; (**b**) indoor scenarios with smoke alarm activated; and (**c**) outdoor scenarios with smoke alarm not activated.

4. AdViSED Thresholds Analysis

To evaluate the performance of the algorithm we run it over a selection of videos available in two datasets, one from [33,34] and the other developed in the FP7 EU project Firesense and largely used in literature [35]. The final dataset is the union of test videos in [33–35], since some scenes are present both in [33,34] and in [35] plus other videos, which are not public available, provided by the Engineering of *Trenitalia*, the Italian national railway company. Such videos have been acquired during specific fire/smoke tests on railway wagons, done at the Trenitalia testing facility in Osmannoro, Italy. The final dataset is composed of both videos with smoke presence and videos with no smoke presence. In both cases, we have outdoor and indoor videos of different quality (i.e., different levels of noise) and different frame rates, from 10 to 60 fps. Most of the used videos are adopted often in literature to compare the smoke/fire measurements and detection performance of state-of-art techniques. For the metrics computation, we need to define the true/false and positive/negative terms. We define a positive video from the dataset as a video with smoke presence, while a negative video is without it. In Table 1, if the algorithm identifies the smoke in a positive video, then we mark the result as true positive (TP). If the algorithm does not identify the smoke in a positive video, then we mark the result as false positive (FP). If the algorithm identifies the smoke in a negative video, then we mark the result as true negative (FN). If the algorithm does not identify the no smoke video in a negative video, then we mark the result as a true positive (TN).

Table 1. Confusion matrix.

	Smoke Presence	**No Smoke Presence**
Alarm	TP	FP
No Alarm	FP	TN

The metrics chosen to evaluate the goodness of the algorithm are recall, precision, accuracy, F1 score, and MCC; see formulas in Equations (7)–(11). The recall means how many relevant items are selected. The precision indicates how many selected items are relevant. The accuracy shows how close you are to the true value. The F1 score is the harmonic mean between precision and recall. The MCC parameter has a range that starts at −1, when there is a complete misalignment between predicted and true value, and ends at +1, when there is a complete alignment between predicted and true value. When the MCC is equal to 0 it means that the prediction is random compared to the true values.

$$Recall = TP/(TP + FN) \tag{7}$$

$$Accuracy = (TP + TN)/(TP + FN + TN + FP) \tag{8}$$

$$F1\ score = 2{\cdot}TP/(2{\cdot}TP + FP + FN) \tag{9}$$

$$Precision = TP/(TP + FP) \tag{10}$$

$$MCC = \frac{(TP{\cdot}TN - FP{\cdot}FN)}{\sqrt{(TP + FP){\cdot}(TP + FN){\cdot}(TN + FP){\cdot}(TN + FN)}} \tag{11}$$

In this section, the analysis of the thresholds is reported. For each threshold type, many tests are performed until discovering the best values. As an example of the tests that have been done, we report in Figure 4 the metrics of some not optimized threshold sets comparing the achieved results to the set n. 6, which is the final selected one optimizing all metrics. The results in Figures 5 and 6 are obtained as an average over the whole testing videos. Each set of thresholds is detailed in Table 2.

Table 2. Different threshold set considered for testing, for all τ_α =10 s, τ_β = 16 s, and $N = 2M$ = 3 s.

Set	Sat	Foreig.	Area	Turb.	Overlap	Ecc.	ConvArea	Ext min	Ext max
1A	0.2	0.8	2000	55	14	0.99	0.5	0.2	0.95
1B	0.2	0.8	2000	55	4	0.99	0.5	0.2	0.95
2A	0.5	0.8	3000	70	7	0.99	0.5	0.2	0.95
2B	0.2	0.16	1000	40	7	0.99	0.5	0.2	0.95
3A	0.2	0.04	2000	55	7	0.99	0.5	0.2	0.95
3B	0.2	0.8	2000	55	7	0.99	0.5	0.2	0.95
4A	0.6	0.8	2000	55	7	0.99	0.5	0.2	0.95
4B	0.1	0.8	2000	55	7	0.99	0.5	0.2	0.95
5A	0.2	0.8	2000	55	7	1	0.9	0.1	0.99
5B	0.2	0.8	2000	55	7	0.7	0.1	0.4	0.80
6	0.2	0.8	2000	55	7	0.99	0.5	0.2	0.95

The THR_{sat} value is set using the real smoke saturation color. The THR_{area}, THR_{turb}, $THR_{eccentr}$, THR_{ext_min}, THR_{ext_max}, and $THR_{convArea}$ values are set using the real features of the blob smoke in the frame. The $kill_{time}$ is set to delete the bounding box within a reasonable time. The $THR_{overlap}$ is set considering a reasonable number of overlapping bounding boxes in a real smoke scene. The THR_{foreg} is set considering the motion detection operation result $\left|I - \hat{BG}_{k-1}\right|$ and filtering this value with the threshold according to the best background subtraction. After these considerations, we adjust the thresholds to find the best values to maximize the metrics. In the thresholds set in Table 2, just a single threshold type is changed, while the other threshold values are kept fixed. The set $1A$ and the set $1B$ are developed using $THR_{overlap} = 14$ *and* 4; the set $4A$ and the set $4B$ are performed using $THR_{sat} = 0.6$ *and* 0.1; the set $3A$ and the set $3B$ are obtained using $THR_{foreg} = 0.16$ *and* 0.04; the set $2A$ and the set $2B$ adopt $THR_{area} = Area/3000$ and $Area/1000$ and $THR_{turb} = 70$ *and* 40; and the set $5A$ and the set $5B$ are developed using $THR_{eccentr} = 1$ *and* 0.7, $THR_{convArea} = 0.9$ *and* 0.1, $THR_{ext\_min} = 0.1$ *and* 0.9, and $THR_{ext\_max} = 0.1$ *and* 0.4; the set 6 in Table 2 is the optimal one.

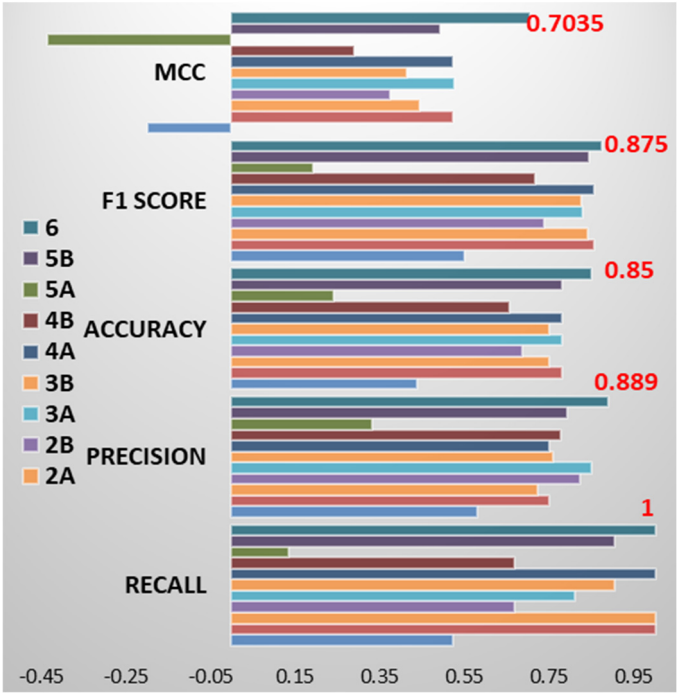


Figure 4. Metrics comparison between different threshold sets.

5. Performance and Complexity Results

Figure 5 shows a comparison in terms of estimation parameters between the algorithm proposed in [22,23] and AdViSED. Another comparison in terms of computational complexity is provided in Figure 6. The latter is evaluated as normalized execution time when running the algorithms via SW on an Intel Core i3-4170 CPU, with Intel HD Graphics 4400 GPU, 8GB DDR3 RAM, equipped with Windows 10 Pro x64 operating system. Results in Figures 5 and 6 represent the average of the results obtained for the whole test videos.

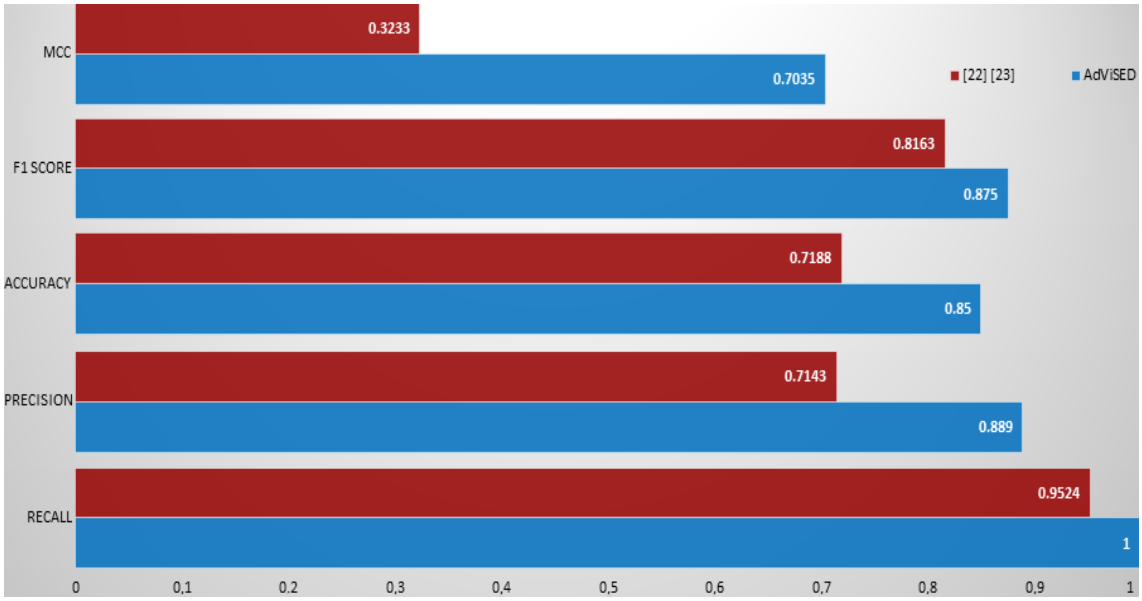


Figure 5. Performance metrics comparison of [22,23] and AdViSED.

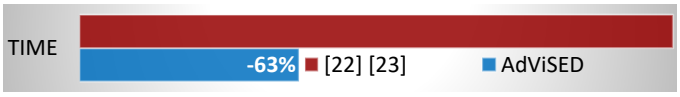


Figure 6. Computation time comparison of [22,23] and AdViSED.

The results in Figure 5 show an improvement in terms of all parameters, MCC, F1 score, accuracy, precision, and recall, of AdViSED compared to the state-of-art techniques [22,23]. This is mainly due to an improved Kalman-based motion estimator in Section 3.1, by the fact that the decisor in Section 3.3 takes into account multiple geometrical parameters (together with the color parameters). The new edge and time-based analysis in Section 3.4 permits also an improvement respect [22,23]. The results in Figure 6 show a relevant improvement of the computational complexity. Indeed, using the same dataset and the same hardware condition, the total amount of time to get the results is less than a third. The improvement is justified by the fact that a more accurate Kalman motion estimation avoids the generation of too many smoke blobs that must be iteratively processed in the next steps.

Figure 7 shows how the computational cost is shared between the different functions of the AdViSED workflow, averaging the results achieved on the considered test video set. From Figure 7 it is clear that the most computing intensive tasks are the feature extraction and alarm decisor steps, since they are applied iteratively on the preselected blobs.

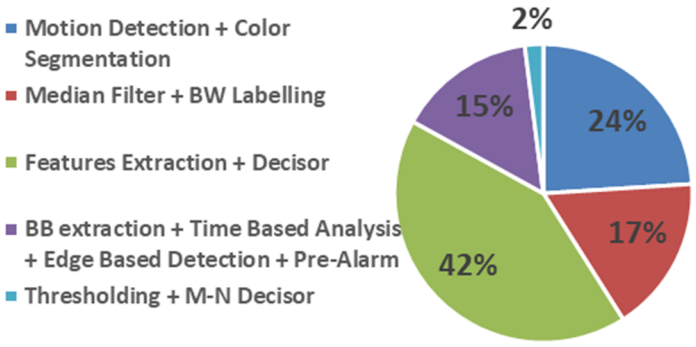


Figure 7. Computation time comparison of [22,23] and AdViSED.

Tables 3 and 4 compare the proposed smoke detection technique to the state of the art [20,21,36–44] using the set of test videos, which is the one showed in Figure 8. The results in Table 3 show that AdViSED has improved recognition capabilities in terms of correctly detected frames and has no false alarms. Table 4 shows instead a comparison with Millan-Garcia's [20], Yu's [42], and Toreyin's [36,40,41] methods in terms of delay (calculated in number of frames) to reveal the presence of smoke/fire. Table 4 shows that AdViSED is faster than the other video-based algorithms reported in the table, except for video n.5 due to the fence in front of the smoke. Even in the worst case of Table 4, an early-alarm is generated in 130 frames, i.e., about 4 s at 30 fps, x15 times faster than EN50155-standardized techniques that react in 60 s.

Table 3. Comparison of correct detected frame and false alarm with respect to state-of-the-art techniques.

Video in Figure 8	Total Frames	Smoke Frames	Toreyin [36,40,41]		Wang [21]		AdViSED	
			Correct Detected	False Alarm	Correct Detected	False Alarm	Correct Detected	False Alarm
n.1	900	805	805	2	836	0	900	0
n.2	900	651	651	4	711	2	900	0
n.3	483	408	235	0	337	0	483	0

Table 4. Comparison of detection delay alarm with respect to state-of-the-art techniques.

Video in Figure 8	Total Frames	Millan-Garcia [20]	Yu [42]	Toreyin [36,40,41]	AdViSED
		Delay in Smoke Detection (in n. frames)			
n.1	900	805	86	98	9
n.2	900	651	121	127	19
n.3	483	408	118	132	120

It is noted that we used different reference works in Tables 3 and 5, since in literature the same set of test videos in Figure 8 is used but the results are reported in different ways. For example, performance metrics are reported as number of correct frames and false alarms by Wang and Toreyin in Table 3, while the works in Table 4 report, as performance metrics, the delay in smoke detection measured in number of frames.

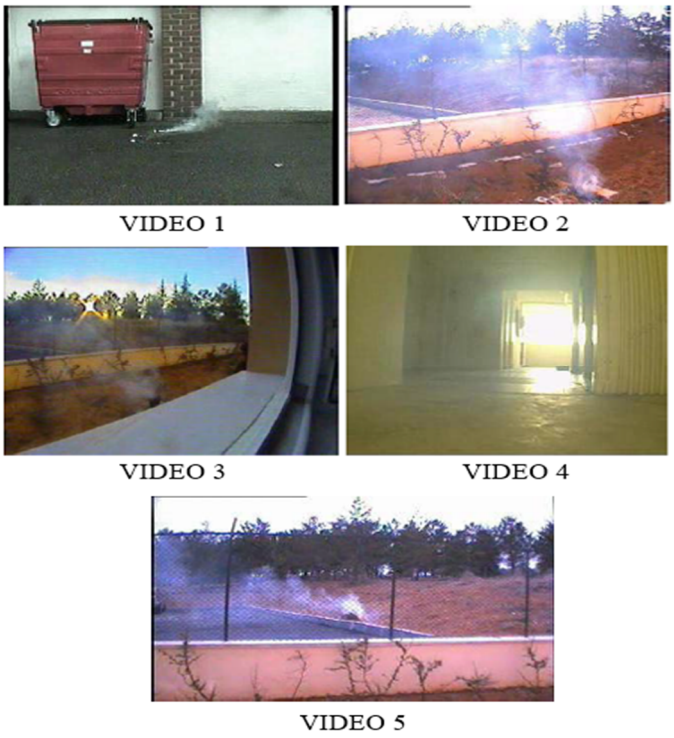


Figure 8. Test video used for comparison with state-of-art smoke detection techniques.

6. Real-Time Embedded Platform Implementation

The AdViSED video-based measuring technique has been implemented on a single board embedded computer, a Raspberry Pi 3 model B (RPI), to test its performance in a real-world scenario. Raspberry Pi in Figure 9 is small, powerful, and low cost at about 30 USD per unit. To this end, it is a perfect platform for applications like distributed and networked measuring notes in smart city or intelligent transport systems scenarios. The board is equipped with a Broadcom BCM2837, a System on Chip (SoC) including a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, with a cache L2 of 512 KB and 1GB of DDR2 RAM, Video Core IV GPU, 4 USB 2.0 ports, onboard WiFi @2.4 GHz 802.11n, Bluetooth 4.1 Low Energy, 40 GPIO pins, and many other features [45]. It runs on Raspbian OS, a Debian-based Linux distribution for download [46]. The camera module used in the implementation is an RPI camera board v1.3 that plugs directly into the CSI (camera serial interface) connector on Raspberry Pi board. The module is able to deliver a 5 MP resolution image or 1080p HD video recording at 30 fps [47]. MATLAB "Run on Hardware" support package is adopted to generate C code from MATLAB algorithm and run it on hardware as a stand-alone application.

Unfortunately, only a subset of MATLAB built-in function and toolboxes are supported for efficient C/C++ code generation on the embedded Broadcom SoC. Therefore, the algorithm was examined and modified introducing consideration for low-level C implementation. Hereafter, we report a list of main modifications implemented in the algorithm description to embed it in the Broadcom SoC.

For example, the original algorithmic description in Section 3 adopts variable-size data structure for the storage of the candidate blobs (that are then eliminated during the time/edge-based analysis in Section 3.4) and for the storage of overlapping structures in Section 3.5. Instead, variable size data structures are critical for embedded systems. To implement fixed size data structure on the stack of the RPI, a statistical analysis of the memory required by AdViSED was carried out using different test videos. During these tests, we observed that the maximum number of bounding boxes, including blobs, displayed together in the whole dataset was not exceeding the number of 200. So that, a fixed size for the data structure containing the coordinate of the boxes and kill frames has been used. It means that for each single frame we can track information for a maximum of 200 boxes.

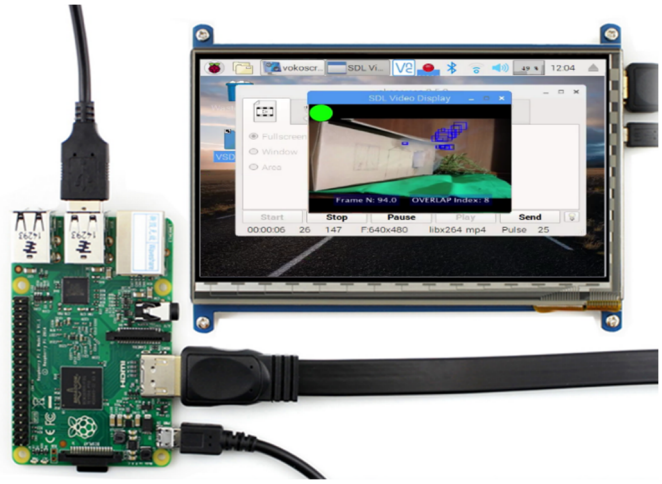


Figure 9. Raspberry Pi 3 model B final implementation at runtime.

Moreover, the array element in the sliding windows were declared as the product of the camera frame rate times the width of the window in seconds (parameter "N" in Section 3), instead of declaring a variable size buffer as in the original algorithmic description. The new data structures are now to be considered as a circular buffer to avoid any type of error at runtime. Size of matrix BG, $\widetilde{BG}$, etc. are obviously dependent of the input frame, and thus, they are declared statically at the beginning in relation to the resolution's camera.

Furthermore, adopted functions in the original AdViSED description, such as "mat2gray" and the feature "convex area" of "regionprops" function are not supported yet for C/C++ code generation on ARM cores. The issue of missing "mat2gray" absence was solved just scaling the value of the matrix from the range 0–255 to the range 0–1. For the "convex area" feature instead, it was not possible to determine any direct workaround; therefore, this function was deleted when implementing AdViSED in the embedded system. For that reason, the $THR_{overlap}$ factor was increased from 7 to 8. The new ready-to-deploy code was tested again on the whole dataset, to check possible differences with respect to the previous version of the code. The same metrics already showed in Sections 4 and 5 have been obtained.

We tested the final implementation using different video resolutions and frame rates and with respect to a porting of AdViSED both on the Broadcom SoC with Raspbian OS and on an x86-based 64b general purpose processor (GPP) using Windows10 OS, as shown in Table 5.

Table 5. Real-time processing and output visualization in fps for different platforms.

Platform	Frame Size				
	320 × 480	**640 × 480**	**1280 × 720**	**1920 × 1080**	**1920 × 1440**
X86 64b PC	19.1 fps	9.32 fps	4.23 fps	N/A	1.54 fps
RPi 3 model B	10.3 fps	2.33 fps	0.64 fps	0.27 fps	N/A

The PC runs at 3.70 GHz, thanks to the Intel Core i3-4170 and 8 GB of RAM. To acquire video frames, we used a simple GitUp Git2 action camera connected via USB on the PC. The performance was compared for both platforms in terms of average frames per second (fps), elaborated for the whole test video set introduced in Section 4. In real-time, we obtain 19 fps for the x86 64b GPP, while the RPI reaches 10.3 fps for the lowest resolution (320 × 480). For 320 × 240 frame size, the maximum frame rate processed in real-time is about 36 fps for the x86 64b GPP and about 19 fps for the RPI platform. The data presented in Table 5 are related to the worst case in which each frame is displayed during the processing on a screen connected to the GPP and on the RPI LCD display. Visualizing the output images requires a lot of computation time; therefore, the application can be sped-up by just closing the

visualization windows. That is why we made other tests, running the application without showing the processed frames and therefore reducing the total overhead. In any case, it was always possible to retrieve information of alarm, overlap index, and processing time on the terminal window. The results in real-time fps without output image visualization are shown in Table 6.

Table 6. Real-time processing in fps for different platforms, without displaying processed output frames.

Platform	Frame Size				
	320 × 480	**640 × 480**	**1280 × 720**	**1920 × 1080**	**1920 × 1440**
X86 64b PC	29.3 fps	17.5 fps	6.07 fps	N/A	1.81 fps
RPi 3 model B	13.4 fps	4.47 fps	01.38 fps	0.87 fps	N/A

Table 6 shows an increasing of the performance in both platforms for all the resolutions. In real-time, we obtain about 30 fps for the x86 64b GPP, while the RPI reaches 13.4 fps for low resolution (320 × 480). For 320 × 240 frame size the maximum frame rate processed in real-time is about 47 fps for the x86 64b GPP and about 25 fps for the RPI platform. The latter value is eight times faster than the result achieved in the state-of-art technique by [27], where an implementation on a Raspberry platform of a video-based smoke measuring system was limited at maximum 3 fps for the same input video resolution.

In Table 6, the implementation on the x86-based 64b GPP ensures, of course, the best performance in terms of maximum frame rate processed in real-time, but the advantage of the RPI over a desktop computer is still the lower cost—30 USD instead of hundreds of USD—lower power consumption, and portability, considering also its relatively good performance. Figure 10 shows power consumption measurements of the Raspberry Pi platform using different configurations and frame sizes. During the tests, a 5V power supply bench is used and measurements are made tracking the current absorption in mA.

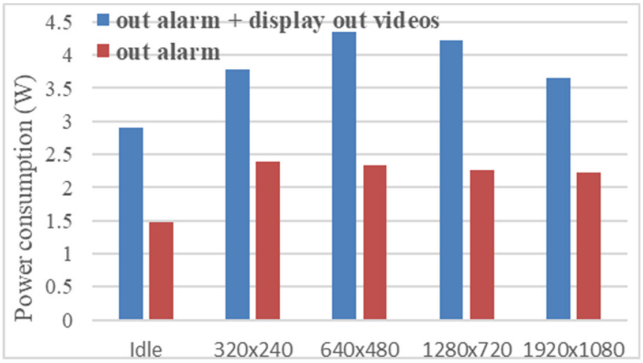


Figure 10. Power consumption in different configurations (each frame size is processed at the max. fps achievable in real-time from Tables 5 and 6).

The board was disconnected from any peripherals, such as keyboard and mouse, except for the camera. In Figure 10 with the term "display", we refer to the configuration test made using a 5-inch display attached to the RPI board (that of Table 5). The other one refers to the version that sends on console data about smoke alarms (that of Table 6). From Figure 10, we can observe that in idle state, the mean power consumption is around 2.9W with the display connected, while during the execution of the application the power consumption increases to maximum 4.36 W. The terminal version of the application (without displaying processed frames) has a power consumption below 1.5 W in idle mode and within 2.4 W in processing mode. Such values are orders of magnitude lower than typical GPP or GPU power costs. Such an embedded device has been also considered as a final device platform by implementing a distributed antifire and surveillance and exploiting an IoT architecture with several camera nodes [12].

7. Conclusions

The paper proposes AdViSED, a novel video smoke detection algorithm for antifire surveillance systems, considering both outdoor and indoor application scenarios. To reduce installations costs, the application scenario considers a fixed single camera, working in the visible spectral range, already installed in a close circuit television system for surveillance purpose. Thanks to the adoption of a Kalman-based motion detection technique, color analysis, image segmentation, blob labeling, time/edge-based bloc analysis, geometrical features analysis, and M out of N decisor, the measurement system is able to generate an alarm signal with improved estimation performance compared with the state-of-art techniques in terms of improved response latency and measurement metrics. The latter are calculated in terms of F1, accuracy precision, recall, and MCC metrics. For example, when compared to [22,23] with a set of video tests, available from the Firesense EU project, the accuracy and precision metrics are improved by about 20%, the MCC score doubles, and the recall increases up to 1. The computational complexity of the proposed technique is reduced by 63% compared with the work in [22,23], when considering the same hardware and software computing platform. With respect to the works in [20,36,40–42], AdViSED ensures a reduced response latency, while achieving equal or better measurement accuracy metrics. Several tests, carried out with different frame rate and frame size, have confirmed the scalability of the proposed measurement techniques to different input camera sensors. AdViSED has been implemented in platforms using both x86 64b GPP processors or embedded ones, based on ARM cores, such as the Raspberry Pi 3. The latter achieves in real-time a performance eight times better than state-of-art works targeting the same embedded unit [27]. Power measurements of the embedded implementation prove that its power cost is below 2.4 W. The low cost and power of the final implementation platform, which also includes Wi-Fi and Bluetooth connection capabilities, make it suitable for the implementation of a distributed measuring systems for smoke/fire surveillance in application scenarios like smart cities or intelligent transport systems. As a future step to improve the camera-based antifire surveillance system we will try adopting some lightweight deep learning networks like MobileNet, ShuffleNet, and SqueezeNet [32–35]. Although these deep neural nets are not directly related to smoke detection, a possible transfer learning solution could be used in order to improve the algorithm used as the base of the proposed distributed antifire and surveillance system [12].

Author Contributions: A.G. and S.S. conceived and designed the experiments, performed the experiments, analyzed the data and wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially supported by Crosslab IoT-Dipartimenti di Eccellenza project by University of Pisa/MIUR and by POR FSE EFEST project (Tuscany Region and Solari di Udine S.P.A.).

Acknowledgments: Discussions with G. Ciarpi for power measurements, M. Borraccino for embedded system porting, and F. Falaschi for EFEST support are gratefully acknowledged.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhuang, J.; Payyappalli, V.; Behrendt, A.; Lukasiewicz, K. *Total Cost of Fire in the United States*; NFPA National Fire Protection Association: Buffalo, NY, USA, 2017. Available online: https://www.nfpa.org/News-and-Research/Data-research-and-tools/US-Fire-Problem/Total-cost-of-fire-in-the-United-States (accessed on 21 April 2020).
2. Toreyin, B.U.; Soyer, E.B.; Urfalioglu, O.; Cetin, A.E. Flame detection using PIR sensors. In Proceedings of the IEEE 16th Signal Processing, Communication and Applications Conference, Aydin, Turkey, 20–22 April 2008; pp. 1–4.
3. Erden, F.; Töreyin, B.U.; Soyer, E.B.; Inac, I.; Günay, O.; Köse, K.; Çetin, A.E. Wavelet based flame detection using differential PIR sensors. In Proceedings of the IEEE 20th Signal Processing and Communications Applications Conference (SIU), Mugla, Turkey, 18–20 April 2012; pp. 1–4.

4. Ge, Q.; Wen, C.; Duan, S.A. Fire localization based on range-range-range model for limited interior space. *IEEE Trans. Instrum. Meas.* **2014**, *63*, 2223–2237. [CrossRef]
5. Aleksic, Z.J. The analysis of the transmission-type optical smoke detector threshold sensitivity to the high rate temperature variations. *IEEE Trans. Instrum. Meas.* **2004**, *53*, 80–85. [CrossRef]
6. Aleksic, Z.J. Minimization of the optical smoke detector false alarm probability by optimizing its frequency characteristic. *IEEE Trans. Instrum. Meas.* **2000**, *49*, 37–42. [CrossRef]
7. Aleksic, Z.J. Evaluation of the design requirements for the electrical part of transmission-type optical smoke detector to improve its threshold stability to slowly varying influences. *IEEE Trans. Instrum. Meas.* **2000**, *49*, 1057–1062. [CrossRef]
8. Amer, H.; Daoud, R. Fault-Secure Multidetector Fire Protection System for Trains. *IEEE Trans. Instrum. Meas.* **2007**, *56*, 770–777. [CrossRef]
9. Amer, H.H.; Daoud, R.M. Fault-Secure Multidetector Fire Protection System for Trains. In Proceedings of the Instrumentation and Measurement Technology Conference (IEEE I2MTC), Ottawa, ON, Canada, 17–19 May 2005; pp. 1664–1668.
10. Wabtec Corp. Available online: www.wabtec.com/products/8582/smir%E2%84%A2 (accessed on 10 January 2020).
11. Wabtec Corp. Available online: www.wabtec.com/products/8579/ifds-r (accessed on 10 January 2020).
12. Gagliardi, A.; Saponara, S. Distributed video antifire surveillance system based on iot embedded computing nodes. In *Applications in Electronics Pervading Industry, Environment and Society (ApplePies 2019)*; Springer LNEE: Cham, Switzerland, 2020; Volume 627, pp. 405–411.
13. Gunay, O.; Toreyin, B.U.; Kose, K.; Cetin, A.E. Entropy-functional-based online adaptive decision fusion framework with application to wildfire detection in video. *IEEE Trans. Image Process.* **2012**, *21*, 2853–2865. [CrossRef]
14. Vijayalakshmi, S.; Muruganand, S. Smoke detection in video images using background subtraction method for early fire alarm system. In Proceedings of the 2017 2nd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 19–20 October 2017; pp. 167–171.
15. Barnich, O.; Van Droogenbroeck, M. ViBE: A powerful random technique to estimate the background in video sequences. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, Taiwan, 19–24 April 2009; pp. 945–948.
16. Zhang, Q.; Liu, F.; Li, X.; Li, B. Dissipation Function and ViBe Based Smoke Detection in Video. In Proceedings of the 2nd International Conference on Multimedia and Image Processing (ICMIP), Wuhan, China, 17–19 March 2017; pp. 325–329.
17. Yuanbin, W. Smoke Recognition Based on Machine Vision. In Proceedings of the International Symposium on Computer, Consumer and Control (IS3C), Xi'an, China, 4–6 July 2016; pp. 668–671.
18. Tao, C.; Zhang, J.; Wang, P. Smoke Detection Based on Deep Convolutional Neural Networks. In Proceedings of the 2016 International Conference on Industrial Informatics—Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), Wuhan, China, 3–4 December 2016; pp. 150–153.
19. Ridder, C.; Munkelt, O.; Kirchner, H. Adaptive Background Estimation and Foreground Detection using Kalman-Filtering. In Proceedings of the International Conference on recent Advances in Mechatronics (ICRAM), Istanbul, Turkey, 14–16 August 1995; pp. 193–199.
20. Millan-Garcia, L.; Sanchez-Perez, G.; Nakano, M.; Toscano-Medina, K.; Perez-Meana, H.; Rojas-Cardenas, L. An early fire detection algorithm using IP cameras. *Sensors* **2012**, *12*, 5670–5686. [CrossRef]
21. Wang, Y.; Chua, T.W.; Chang, R.; Pham, N.T. Real-time smoke detection using texture and color features. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012; pp. 1727–1730.
22. Saponara, S.; Pilato, L.; Fanucci, L. Early Video Smoke Detection System to Improve Fire Protection in Rolling Stocks. *Proc. SPIE* **2014**, *9139*, 913903.
23. Saponara, S.; Pilato, L.; Fanucci, L. Exploiting CCTV camera system for advanced passenger services on-board trains. In Proceedings of the IEEE International Smart Cities Conference (ISC2), Trento, Italy, 12–15 September 2016; pp. 1–6.
24. Flir Corp. Available online: https://flir.netx.net/file/asset/8888/original (accessed on 10 January 2020).
25. Brisley, P.M.; Lu, G.; Yan, Y.; Cornwell, S. Three-dimensional temperature measurement of combustion flames using a single monochromatic CCD camera. *IEEE Trans. Instrum. Meas.* **2005**, *54*, 1417–1421. [CrossRef]

26. Qiu, T.; Yan, Y.; Lu, G. An autoadaptive edge-detection algorithm for flame and fire image processing. *IEEE Trans. Instrum. Meas.* **2011**, *61*, 1486–1493. [CrossRef]
27. Foggia, P.; Saggese, A.; Vento, M. Real-time fire detection for video surveillance applications using a combination of experts based on color, shape, and motion. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *25*, 1545–1556. [CrossRef]
28. Muhammad, K.; Ahmad, J.; Mehmood, I.; Rho, S.; Baik, S.W. Convolutional neural networks based fire detection in surveillance videos. *IEEE Access* **2018**, *6*, 18174–18183. [CrossRef]
29. Filonenko, A.; Hernández, D.C.; Jo, K.H. Fast smoke detection for video surveillance using CUDA. *IEEE Trans. Ind. Inform.* **2017**, *14*, 725–733. [CrossRef]
30. Zhao, C.; Sain, A.; Qu, Y.; Ge, Y.; Hu, H. Background Subtraction based on Integration of Alternative Cues in Freely Moving Camera. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *29*, 1933–1945. [CrossRef]
31. Wu, Y.; He, X.; Nguyen, T.Q. Moving object detection with a freely moving camera via background motion subtraction. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *27*, 236–248. [CrossRef]
32. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
33. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.
34. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
35. Iandola, F.; Keutzer, K. Small neural nets are beautiful: Enabling embedded systems with small deep-neural-network architectures. In Proceedings of the Twelfth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion, Seoul, Korea, 15–20 October 2017; p. 1.
36. Çetin, A.E.; Dimitropoulos, K.; Gouverneur, B.; Grammalidis, N.; Günay, O.; Habiboğlu, Y.H.; Verstockt, S. Video fire detection—Review. *Digit. Signal Process.* **2013**, *23*, 1827–1843. [CrossRef]
37. Sample Smoke Video Clips. Available online: http://signal.ee.bilkent.edu.tr/VisiFire/Demo/SampleClips.html (accessed on 10 January 2020).
38. Grammalidis, N.; Dimitropoulos, K.; Cetin, E. Firesense Database of Videos for Flame and Smoke Detection. Available online: https://zenodo.org/record/836749 (accessed on 10 January 2020).
39. Dimitropoulos, K.; Barmpoutis, P.; Grammalidis, N. Spatio-temporal flame modeling and dynamic texture analysis for automatic video-based fire detection. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *25*, 339–351. [CrossRef]
40. Toreyin, B.U.; Dedeoglu, Y.; Cetin, A.E. Contour based smoke detection in video using wavelets. In Proceedings of the 14th European Signal Processing Conference, Florence, Italy, 4–8 September 2006; pp. 1–5.
41. Töreyin, B.U.; Dedeoğlu, Y.; Cetin, A.E. Wavelet based real-time smoke detection in video. In Proceedings of the 13th European Signal Processing Conference, Antalya, Turkey, 4–8 September 2005; pp. 1–4.
42. Yu, C.; Fang, J.; Wang, J.; Zhang, Y. Video fire smoke detection using motion and color features. *Fire Technol.* **2010**, *46*, 651–663.
43. Yuan, F.; Fang, Z.; Wu, S.; Yang, Y.; Fang, Y. Real-time image smoke detection using staircase searching-based dual threshold AdaBoost and dynamic analysis. *IET Image Process.* **2015**, *9*, 849–856. [CrossRef]
44. Celik, T.; Demirel, H. Fire detection in video sequences using a generic color model. *Fire Saf. J.* **2009**, *44*, 147–158. [CrossRef]
45. Raspberry Pi Foundation. Available online: https://www.raspberrypi.org/ (accessed on 10 January 2020).
46. Raspbian OS. Available online: https://www.raspberrypi.org/downloads/raspbian/ (accessed on 10 January 2020).
47. RPI Camera Module v.1.3. Available online: https://www.raspberrypi.org/documentation/hardware/camera/ (accessed on 10 January 2020).