sensors

MDPI

Article

A Video Based Fire Smoke Detection Using Robust AdaBoost

Xuehui Wu [1,2], Xiaobo Lu [1,2,*] and Henry Leung [3]

1 School of Automation, Southeast University, Nanjing 210096, China; xhwu@seu.edu.cn
2 Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education, Southeast University, Nanjing 210096, China
3 Department of Electrical and Computer Engineering, University of Calgary, 2500 University Dr N.W., Calgary, AB T2N 1N4, Canada; lenugh@ucalgary.ca
* Correspondence: xblu2013@126.com

Received: 26 September 2018; Accepted: 31 October 2018; Published: 5 November 2018

Abstract: This work considers using camera sensors to detect fire smoke. Static features including texture, wavelet, color, edge orientation histogram, irregularity, and dynamic features including motion direction, change of motion direction and motion speed, are extracted from fire smoke to train and test with different combinations. A robust AdaBoost (RAB) classifier is proposed to improve training and classification accuracy. Extensive experiments on well known challenging datasets and application for fire smoke detection demonstrate that the proposed fire smoke detector leads to a satisfactory performance.

Keywords: fire smoke detection; video based; feature extraction; robust AdaBoost; classifier

1. Introduction

Detection fire smoke at the early stage has drawn a lot of attentions recently due to its importance to social security and economic development. Conventional point fire smoke detector sensors are effective for indoor applications, but they have difficulties to detect smoke in large outdoor areas, it is because point fire smoke detector typically detect the presence of certain particles generated by smoke and fire by ionization [1], photometry [2], or smoke temperature [3,4]. They require a close proximity to fire and smoke, which are not effective for open spaces. For particles to reach these sensors to activate alarms, many sensors are needed to cover a large area which is not cost effective.

Video based fire smoke detection using cameras is of great interest in large and open spaces [5]. Closed circuit television (CCTV) surveillance systems are widely installed in many public areas to date. These systems can be used to provide early fire smoke detection if a reliable fire detection software is installed in the system. Gottuk et al. [6] test three commercially available video based fire detection systems against conventional spot systems in a shipboard scenario. Video based systems are found to be more effective in flame detection. These systems are economically viable as CCTV cameras are already available for traffic monitoring [7] and surveillance [8] applications. Braovic, M et al. [9] propose an expert system for fast segmentation and classification of regions on natural landscape images that is suitable for real-time automatic wildfire monitoring and surveillance systems. It is noted that smoke is always visible before fire in most outdoor scenarios. This motivates us to research on detecting smoke in the absence or presence of flame from a single frame of video.

There are some technical challenges in video based fire smoke detection. First, it is observed to be inferior to particle-sampling based detectors in terms of false alarm rate. It is mainly due to the variability in smoke density, scene illumination, interfering objects. Second smoke and fire are difficult to be modeled, most of the existing image processing methods do not characterize smoke

well [10]. Current fire detection algorithms are based on the use of static and motion information in video to detect flames [11–15]. Many efforts have been made to reduce the false alarm rate and missing detection rate. Table 1 shows some static and dynamic features used in these approaches. As shown, the most used features are color, texture, energy and dynamic features.

Table 1. Literatures about fire smoke feature extraction.

	Color	Contour	Texture	Energy	Irregularity	Other	Dynamic	Training
[16]	√						√	
[17]	√			√			√	
[18]	√	√		√				
[15]	√			√	√			
[19]	√						√	
[20]	√						√	
[21]						Dark channel		
[22]						Sparsity		
[23]						Chrominance	√	√
[24]	√	√						√
[25]			√	√			√	√
[26]			√				√	√
[27]			√		√		√	√
[28]	√		√		√		√	√
[29]	√		√	√			√	√
Total	9	2	5	5	3	3	9	7

On the other side, for classification, some researchers [15,16,18–22] use thresholds or parameters from analysis of extracted features, which is less time-consuming but not adaptive in some complex environments. Some other researchers [30,31] use well known AdaBoost to training models and classify. AdaBoost based component classifiers can solve the overfitting problem relatively with high precision, one weak classifier of component classifiers is a learner which can return a hypothesis that roughly outperform random guessing. For the component classifiers, the weights updates of them in every step mainly depend on the last errors,

$$\alpha_m = \frac{1}{2} ln \frac{1 - e_m}{e_m}, \tag{1}$$

α_m gives the weights of component classifiers, and e_m is the errors. Definitely, the weights α_m should be positive, so the error e_m is required to be less than 0.5, and AdaBoost also requires the error not much less than 0.5, so that the boosting function can make sense of these cascade classifiers. Therefore, we must select a series of proper base classifiers and set a best parameter for every base classifier prudently and it is also time-consuming.

Deep learning is also considered for smoke fire detection. In [32], a binary classifier is trained using annotated patches from scratch, second, learning and classification using cascade convolutional neural network (CNN) fire detector. Muhammad et al. [33] propose an adaptive prioritization mechanism for fire smoke detection. CNN and the internet of multimedia things (IoMT) for disaster management are used for early fire detection framework. Through deep learning, features can be learned automatically. However, these deep learning methods for fire smoke detection are still limited by learning static features. Wu et al. [34] combine deep learning method and conventional feature extraction method to recognize the fire smoke areas. CNN is used in Caffe framework to achieve a Caffemodel with static features. This deep learning approach can work well to certain extent, but the dynamic feature is not trained directly for most situations.

As for fire smoke classification, thresholds or parameters from extracted features are usually used in classification which is less time-consuming. However, it is not adaptive to deal with complex environments [15,16,18–22]. SVM and AdaBoost have been considered for classifier in [23–29].

SVM can achieve good accuracy with a sufficiently large training set. Some kernel functions can make it possible to solve the nonlinear problem in high dimensional space [35–38]. One popular kernel used in SVM is RBF (RBFSVM). RBFSVM determines the number and location of the centers and the weight values automatically [39]. It uses a regularization parameter, C, to balance the model complexity and accuracy. Another parameter known as Gaussian width, σ, can be used to avoid the over-fitting problem. However, it is difficult to select proper values of C and σ. AdaBoost based component classifiers can solve the overfitting problem with relatively high precision. One weak classifier of component classifiers is a learner which can return a hypothesis that roughly outperform random guessing.

In this paper, static features, include texture, wavelet, color, hog, irregularity, and dynamic features, include motion direction, change of motion direction and motion speed, are extracted and trained with different combinations. A robust AdaBoost (RAB) classifier is designed for detection. It can overcome the weights update problem and solve the dilemma between high accuracy and over-fitting. RBFSVM (Radial Basic Function—Support Vector Machine) [35] is chosen as the base classifier with some improvements made to guarantee the validity and accuracy of the boosting function. An effective algorithm for locating the original fire position is introduced in this paper for practical fire smoke detection.

The remainder of this article is organized as follows: Video based fire smoke detection using camera are described in Section 2, including static feature in Section 2.1 and dynamic feature extraction in Section 2.2. The proposed Robust AdaBoost classifier for fire smoke detection is presented in Section 3. Performance of the proposed method are evaluated by extensive experiments in Section 4. The paper is concluded in Section 5.

2. Features Extraction

Fire and smoke training data usually come from pre-collected image datasets [40]. Due to the difficulty in collecting this type of data, these sets are limited and most of them include other random information besides fire smoke as shown in Figure 1. In [29,34], image data sets are used for static feature extraction and set threshold values for classifying dynamic features in real-time video detection. Although this approach can work well to some extent, it is difficult to set a single threshold value that works for situation.

(**a**) (**b**) (**c**) (**d**)

Figure 1. (**a**–**d**) show the fire and smoke image data sets with none-smoke objects.

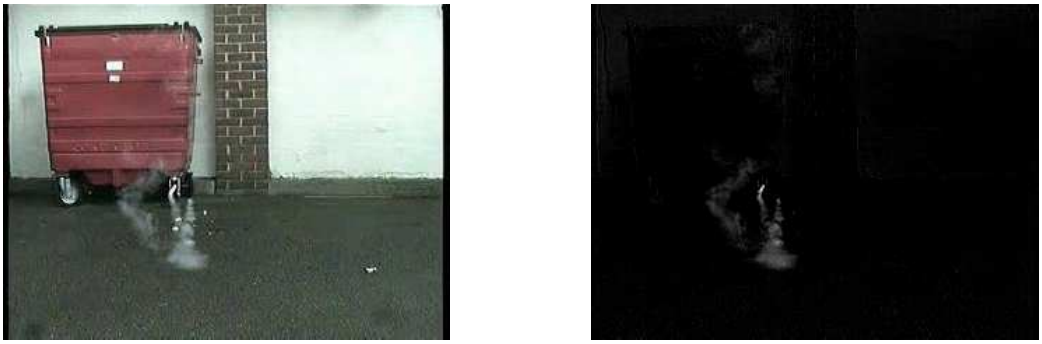
In this paper, different smoke and non-smoke videos taken from camera are used as training samples. Motion regions of all video sequences are used for static feature extraction, and dynamic features are extracted depending on the correlations among successive frames. These training videos are collected and preprocessed so that the videos contain only fire smoke moving objects.

Robust Orthonormal Subspace Learning (ROSL) [41] method is used to segment foregrounds from images.

$$\min_{D,\alpha,E} \|\alpha\|_{row-1} + \lambda\|E\|_1, s.t\ D\alpha + E = M, A = D\alpha, D^T D = I_k, \forall i, \tag{2}$$

where M denotes the observed matrix of video, A is the low-rank background matrix, and M is the foreground matrix. This method represents A under the ordinary orthonormal subspace

$D = [D_1, D_2, ..., D_k] \in \mathbb{R}^{m \times n}$, where coefficients $\alpha = [\alpha_1; \alpha_2; ...; \alpha_k] \in \mathbb{R}^{k \times n}$. The dimension k of the subspace is set as $k = \beta r$ (β is a constant and $\beta > 1$). E is the sparse matrix which represents foregrounds. Foreground segmentation results are shown in Figure 2. Motion regions are selected frame by frame. In order to choose the exact sample data sets that we need, tiny motion regions which may be created by minute jitter or too big ones should be eliminated.



Original image Foreground

Figure 2. Foreground segmentation.

2.1. Static Features Extraction

Color, texture and energy in wavelet domain are three commonly used static features as shown in Table 1. Here we consider three other static features including Edge Orientation Histogram (EOH) [42], irregularity and sparsity, and three dynamic features including motion direction, change of motion direction and motion speed.

2.1.1. Color, Texture and Energy

Color moments are measures that can be used to differentiate images. Stricker and Orengo [43] propose three central moments of an image color distribution such as mean, standard deviation and skewness. Here we use HSV (Hue, Saturation, and Value) to define three moments for each three color channels: mean, standard deviation and skewness of the ith color channel at the jth image pixel p_{ij}:

$$E_i = \frac{1}{N} \sum_{j=1}^{N} p_{ij} \tag{3}$$

$$\sigma_i = \sqrt{\left(\frac{1}{N} \sum_{j=1}^{N} (p_{ij} - E_i)^2 \right)} \tag{4}$$

$$s_i = \sqrt[3]{\left(\frac{1}{N} \sum_{j=1}^{N} (p_{ij} - E_i)^3 \right)} \tag{5}$$

LBP feature has been used to capture spatial characteristics of an image as reported in [22,44]. The LBP features are extracted as texture descriptor of smoke in this study. When smoke is found in a region, the edge of background will be blurred so that the high frequency energy of this region will decrease. Gabor wavelet is used to get three high frequency components in horizontal (HL), vertical (LH) and diagonal (HH) directions. The energy of image in region R_i is defined as

$$EN_i = \frac{\sum_{(x,y) \in R_i} w(x,y)}{\sum_{(x,y) \in R_i} W(x,y)}, \tag{6}$$

where EN_i is the energy of image in region R_i and $w(x,y)$ is high frequency wavelet coefficients of pixel at (x,y) in region R_i and $w(x,y)$ is the entire wavelet coefficients of the same pixel. We have

$$w(x,y) = |HL(x,y)|^2 + |LH(x,y)|^2 + |HH(x,y)|^2, \tag{7}$$

$$W(x,y) = |HL(x,y)|^2 + |LH(x,y)|^2 + |HH(x,y)|^2 + |LL(x,y)|^2. \tag{8}$$

2.1.2. Edge Orientation Histogram

Gradients of an image plays an important role in object detection. The Scale Invariant Feature Transform (SIFT) [45] achieves an impressive performance on image registration and object detection. As a variant of SIFT, Histograms of Oriented Gradients (HOG) [46] are used for human detection. HOG can capture local gradient-orientation structures within an image but it is computed on a dense grid of uniform space over an image. Therefore, a high dimensional feature vector is produced to describe each detection window, which is not suitable for real-time application like fire smoke detection. Levi and Weiss [42] propose the Edge Orientation Histogram (EOH) method which measures the response of linear edge detectors at different subareas of the input image.

In this paper, we use Canny operator to generate image edge and the gradient which contains horizontal component G_x and vertical component G_y. The edge gradient $[G_x, G_y]$ is then converted into polar coordinate. That is

$$m(x,y) = \sqrt{G_x^2 + G_y^2}, \tag{9}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right), \tag{10}$$

where m is the magnitude and θ denotes the orientation.

The edge orientations are mapped to the range $[-180^\circ, 180^\circ]$ with an interval of 10°. The numbers of edge orientation in these 36 angle ranges are counted using two angle threshold matrices as $T_1 = [-170^\circ, -160^\circ, ..., 160^\circ, 170^\circ, 180^\circ]$ and $T_2 = [-180^\circ, -170^\circ, ..., 150^\circ, 160^\circ, 170^\circ]$,

$$h(k) \begin{cases} +1, \theta(x,y) \in [T_2(k)\, T_1(k)) \\ +0, \text{otherwise} \end{cases}, k = 1, 2, ..., 36. \tag{11}$$

For each angle range $[T_2(k), T_1(k))$, the EOH is obtained by summing all the gradient magnitudes whose orientations belongs to this range,

$$E(k) = \sum_{\theta(x,y) \in [T_2(k), T_1(k)]} m(x,y) \tag{12}$$

The EOH features can be extracted with two different methods- Dominant Orientation Features (DOF) and Symmetry Features (SF) [42]. When we try to find the dominant edge orientation in a specific area rather than the ratio between two different orientations, we define a slightly different set of features which measures the ratio between a single orientation and the others. That is

$$E_{k,dof} = \frac{E(k) + \varepsilon}{\sum_{k=1}^{36} E(k) + \varepsilon}, k = 1, 2, ..., 36, \tag{13}$$

where $E_{k,dof}$ records the ratio of every single orientation from $E_{k,dof}$ to $T_1(k)$, ε is a tiny positive value to avoid division by zero. According to Equation (7), the domain orientation is located in $[T_2(k_d), T_1(k_d))$.

According to [42], we define the symmetry features in $[T_2(k), T_1(k))$ as

$$E_{k,sf} = \begin{cases} \frac{|R_1(k) - R_2(k)|}{h(k)}, h(k) \neq 0 \\ \max\left(E_{k,sf}\right), h(k) = 0 \end{cases}, \tag{14}$$

To normalize this feature, we reset it as

$$E_{k,sf} = \frac{E_{k,sf}}{\max\left(E_{k,sf}\right)}, \tag{15}$$

where $(R_1(k), R_2(k)) = \begin{cases} (E(k), E(18+k)), k \in [1,18] \\ (E(k), E(k-18)), k \in [19,36] \end{cases}$, R_1 and R_2 are rectangles of the same size and are positioned at opposite sides of the symmetry axes. Not only the symmetry features can be used to find symmetry, but it can also find places where symmetry is absent. From Equation (9), a small $E_{k,sf}$ means that the image orients more in $[T_2(k), T_1(k)]$ compared to its opposite side range.

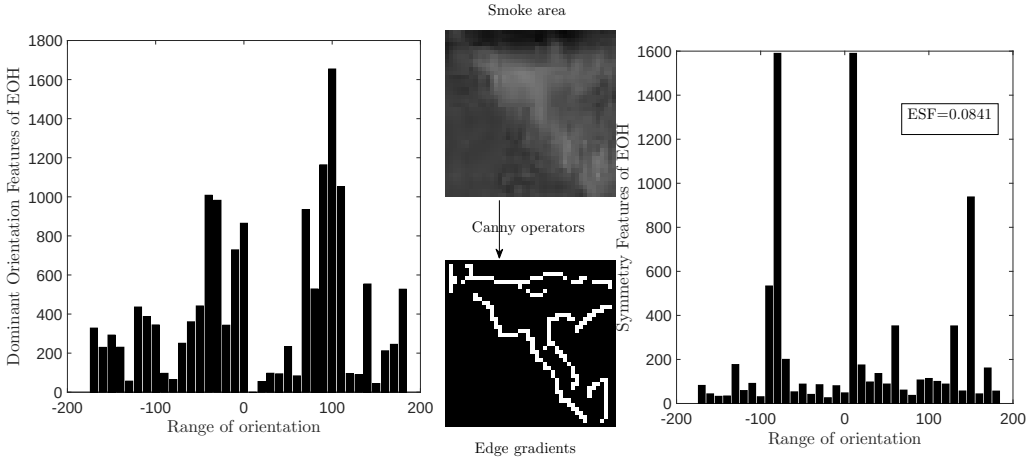
The symmetry value of the entire detected region is computed as

$$E_{SF} = \frac{\left|2\sum_{k=k_s}^{17+k_s} E_{k,sf} - \sum_{k=1}^{36} E_{k,sf}\right| + \varepsilon}{\sum_{k=1}^{36} E_{k,sf} + \varepsilon}, \tag{16}$$

$$\begin{aligned} k_s &= \arg\min_{k^*} \left|\sum_{k=k^*}^{17+k^*} E_{k,sf} - \left(\sum_{k=1}^{36} E_{k,sf} - \sum_{k=k^*}^{17+k^*} E_{k,sf}\right)\right| \\ &= \arg\min_{k^*} \left|2\sum_{k=k^*}^{17+k^*} E_{k,sf} - \sum_{k=1}^{36} E_{k,sf}\right|, k^* \in [1,18] \end{aligned} \tag{17}$$

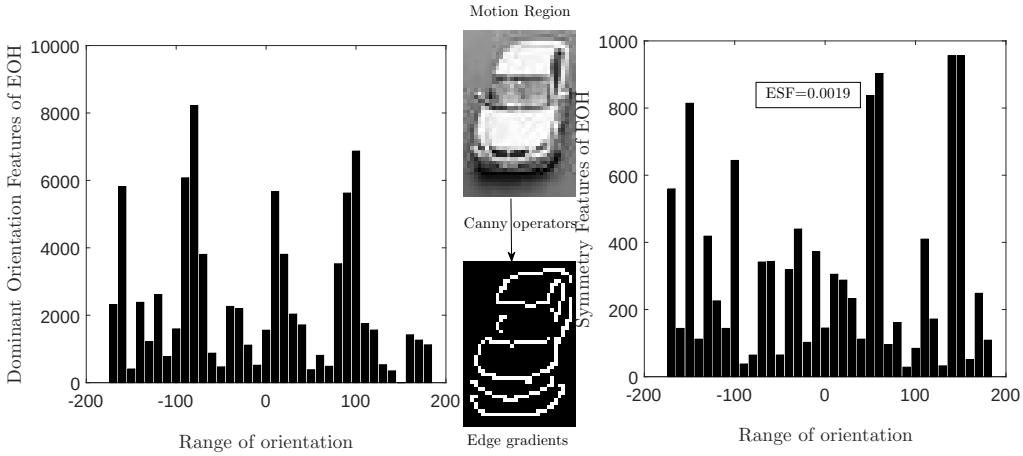
E_{sf} is small when the detected motion region is symmetric.

Figure 3 shows the two different EOH features (E_{dof} and E_{sf}) of smoke and car images. In this paper, we concatenate these two EOH features to characterize the edge features.



(**a**)

Figure 3. *Cont.*



(**b**)

Figure 3. Histogram of Edge Orientation: Left figure of (**a**,**b**): Dominant Orientation Features. Right figure of (**a**,**b**): Symmetry Features.

2.1.3. Irregularity and Sparsity

Fire smoke have no regular shapes in diffusion, this irregularity feature is defined as

$$IR = \frac{c^2}{4\pi s} \tag{18}$$

where IR is the irregularity, c is the edge perimeter of detected region and s is the area.

For the sparsity feature, it is mainly for light smoke as described in [22]. We extract the sparse foreground with ROSL [41], and E in Equation (2) is the foreground matrix. Figure 4 shows the extracted sparse smoke without backgrounds.

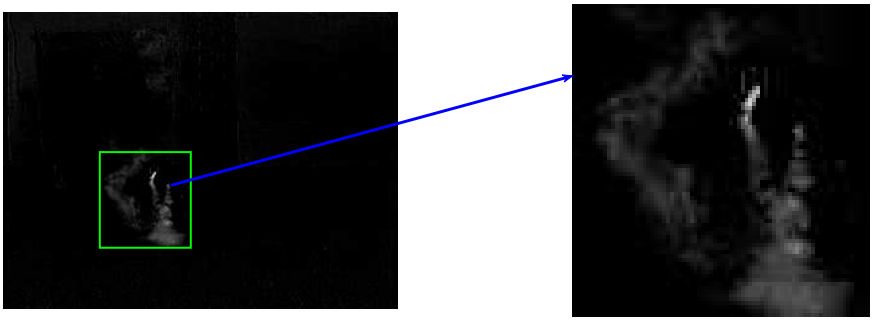


Figure 4. Sparsity of foreground.

The sparsity value is calculated as

$$SP = \frac{\|F_M\|_0}{S_{F_M}} \tag{19}$$

where F_M is the matrix of region M, S_{F_M} is the size of F_M. Figure 4 shows the foreground motion region. The $\ell 1$ norm counts the numbers of non-zero pixels in foreground.

2.2. Dynamic Features Extraction

Fire smoke has special dynamic features that provides information for recognition. Motion direction, change of direction and motion speed are extracted as three dynamic features. Before extracting these features, we need to find the locations of the detected motion region in

consecutive frames. The centroid of current motion region is first calculated. Because a moving object in two adjacent frames has similar location and shape, these two motion regions have maximum overlap ratio and the shortest distance. Figure 5 shows the mechanism of finding the locations of these two corresponding motion regions. In Figure 5, A, B and C are the centroid of three successive frames, A1 and C1 are the two overlapping regions, region A has the maximum overlap ratio computed with current detected region B for the last frame while region C has the maximum overlapping ratio computed with current detected region B for the next frame, the overlapping ratio is calculated as

$$OVR_{Last} = \frac{S_{A_1}}{S_A + S_B - S_{A_1}}, OVR_{next} = \frac{S_{C_1}}{S_A + S_C - S_{C_1}} \tag{20}$$

where $S.$ represents the area of each region shown in the figure.

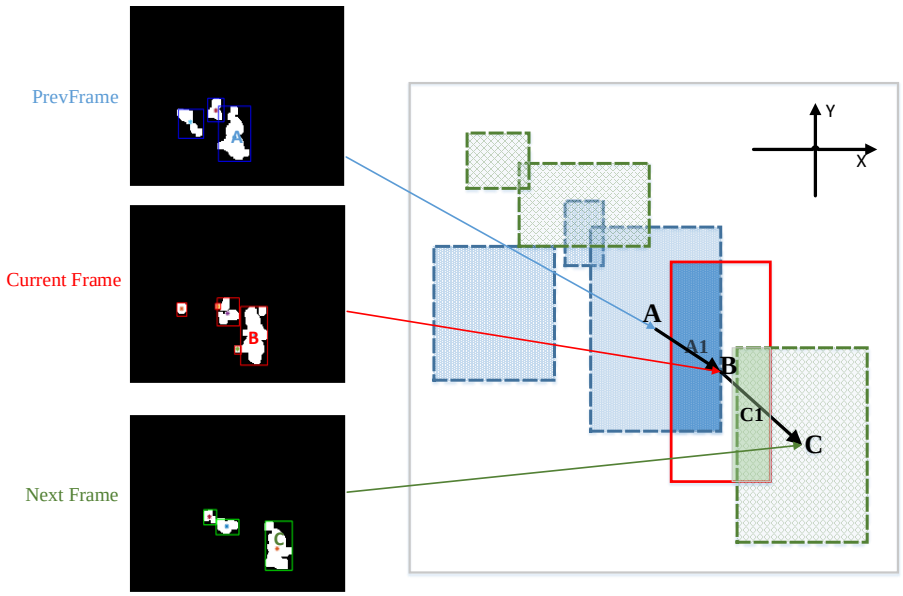


Figure 5. Overlapping ratio computing.

From Figure 5, motion region B is detected moving from A to C in three consecutive frames. Hence, its motion direction can be calculated with the orientation of $\overrightarrow{AB}$ and $\overrightarrow{BC}$. Define the horizontal component of $\overrightarrow{AB}$ as f_{X-AB} and the vertical component of $\overrightarrow{AB}$ as f_{X-AB}, the motion direction of region B is calculated as

$$md_A = \tan^{-1}\left(f_{X-AB}/f_{Y-AB}\right), \tag{21}$$

As smoke moves slowly and its motion is not obvious in adjacent two frames, we use the statistic measurement to extract the motion features by calculating the mean values of corresponding motion region directions from t_1 and t_2 instead of that in adjacent two frames. The final motion direction feature of one motion region in frame t_1 is defined by

$$MD_{t1} = \frac{\sum_{i=t_1}^{t_2} md_i}{t_2 - t_1}. \tag{22}$$

We set the frame interval as $t_2 - t_1 = 5$. Each motion region can be calculated by following the scheme above (MD in the first four frames and last four frames cannot be calculated), such that we can also obtain one motion direction change in frame t_1

$$mdc_{t1} = md_{t1} - md_{t2}, \tag{23}$$

Another dynamic feature is called motion speed, while we can easily achieve the motion speed with the distance of two corresponding motion regions in consecutive two frames as

$$ms_A = \left|\overrightarrow{AB}\right| \times R_f. \tag{24}$$

where E_f is the frame rate, the motion speed from t_1 to t_2 is calculated as

$$MS_{t1} = \frac{\sum_{i=t_1}^{t_2} ms_i}{t_2 - t_1}. \tag{25}$$

Combining different features can enhance the robustness of features. According to [24], features can be concatenated together to form a feature vector in cascade fusion as follows:

$$FV_i = \left\{F_i^1, F_i^2, ..., F_i^n, \right\} \tag{26}$$

where F_i^n represents every single features of one image.

Figure 6 shows the framework of fire smoke detection in this paper including motion region detection, feature extraction and samples training and fire smoke classification. As for the classifier, we propose a robust AdaBoost method in the following section.

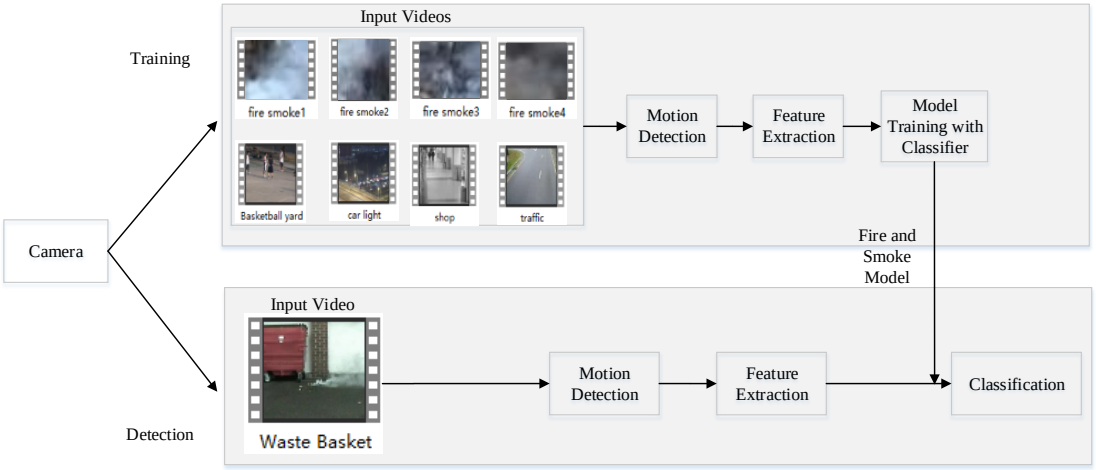


Figure 6. Framework of fire smoke detector.

3. The Proposed Robust AdaBoost Classifier for Fire Smoke Detection

3.1. AdaBoost

Given a set of samples $s = \{(x_i, y_i)\}_{i=1}^N$, $x \in \chi$, χ represents the input space, and $\chi \subset \mathbb{R}^N$, $y \in \{-1, +1\}$ is the label. The goal of AdaBoost is to train a series of weak classifiers to be a stronger one. Boosting algorithms improve the performance of a learning algorithm by calling a weak learner in a succession of cycles, in each cycle, it maintains a weight distribution D_m over the weak learner with input set S and returns a hypothesis h_m. The weight distribution is initially set uniform as D_1,

$$D_1 = (w_{11}, w_{12}, ..., w_{1i}, ..., w_{1N}), w_{1i} = \frac{1}{N}, i = 1, 2, ..., N, \tag{27}$$

We use the distribution D_m to train the base classifier $G_m(x)$ with training dataset and get errors e_m,

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(h_{mi} \neq y_i) \in [0,1]. \tag{28}$$

The weight of weak classifier α_m can be achieved with error e_m by $\alpha_m = \frac{1}{2} ln \frac{1-e_m}{e_m}$, when $e_m \leq \frac{1}{2}$, $\alpha_m \geq 0$, and α_m increases with the decrease of e_m. That is, the smaller the classification error is, the more important the role of this weak classifier is of the component classifiers. For the next cycle, the weights of samples are updated as

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2}, ..., w_{m+1,i}, ..., w_{m+1,N}), \tag{29}$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} e^{(-\alpha_m y_i G_m(x_i))}, \left(w_{0,i} = \frac{1}{N}\right) \tag{30}$$

$$Z_m = \sum_{i=1}^{N} w_{mi} e^{(-\alpha_m y_i G_m(x_i))}, \tag{31}$$

where Z_m is the normalization factor. This distribution D_{m+1} shows that the weights of the misclassified samples will be increased and the weight of the correctly classified samples will be reduced. In this way, AdaBoost will be forced to pay its attention to the samples that have been misclassified in the next cycle.

Combine all component weak classifiers,

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x), \tag{32}$$

where M is the total number of cycles. The final classifier is defined as

$$G(x) = sign(f(x)), \tag{33}$$

AdaBoost calls this algorithm repeatedly in a series cycles and can get a lower training error with a higher accuracy.

3.2. Proposed Robust AdaBoost

One important theoretical property of AdaBoost is that component classifiers need to be only slightly better than random guessing, the weight α_m will be negative if $e_m > 0.5$ and be larger than 1 when $e_m < \frac{1}{e^2+1}$, that means we should spend time on selecting moderate parameter so that the error created by this weak classifier is slightly less than 0.5, otherwise, the classifier with classification error slightly more than 0.5 will be discarded. When the weak classifier error is less than 0.1, its weight will be relatively high, it can just take control of errors in a tiny limited range. However, AdaBoost treat this limited controlled errors as the only key for updating weights. Either too strong or too weak a classifier will not satisfy the boosting requirements. The weight calculation of base classifier is modified as

$$\alpha_m{}^* = \left(\ln \frac{2e}{1+e^{e_m}}\right)^n, n \geq 1. \tag{34}$$

As $e_m \in [0,1]$,$\alpha_m{}^* \in \left[\left(\ln \frac{2e}{1+e}\right)^n, 1\right]$, $\ln \frac{2e}{1+e} \approx 0.38$. Figure 7 shows these two weight computing methods of conventional AdaBoost and robust AdaBoost in this paper. $\alpha_m{}^*$ also decrease with an increasing e_m. No matter how weak or strong the base classifier is, it can achieve its corresponding weight. The stronger the base classifier is, the more important it will be. While n is large enough, the weight of the base classifier that slightly outperforms random guessing ($e_m \approx 0.5$) will almost equal

to zero. When e_m approaches to zero, the weight $\alpha_m{}^*$ approaches to zero. Therefore, our proposed method is consistent with boosting theory.

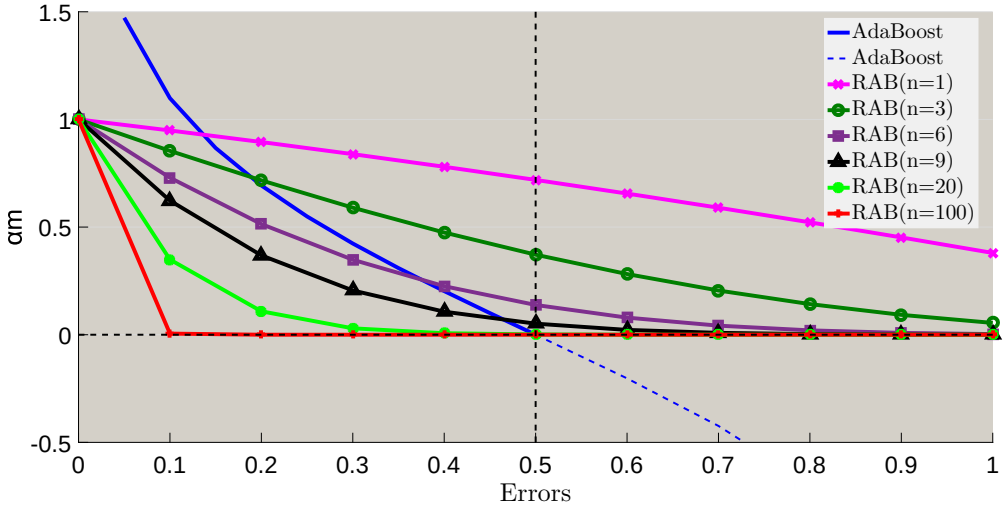


Figure 7. Base classifier weights update.

AdaBoost can decrease its training errors through its learning. Its error upper bound is given by

$$E = \frac{1}{N}\sum_{i=1}^{N} I\left(G\left(x_i\right) \neq y_i\right) \leq \frac{1}{N}\sum_{i=1}^{N} \exp\left(-y_i f\left(x_i\right)\right), \tag{35}$$

when $G\left(x_i\right) \neq y_i$,$y_i f\left(x_i\right) \leq 0$, $\exp\left(-y_i f\left(x_i\right)\right) \geq 1$. According to Equations (30), (31) and (35), the error upper bound can be calculated as

$$\begin{aligned}
E &= \tfrac{1}{N}\sum_{i=1}^{N} \exp\left(-y_i f\left(x_i\right)\right) = \tfrac{1}{N}\sum_{i=1}^{N} \exp\left(-\sum_{m=1}^{M} \alpha_m y_i G_m\left(x_i\right)\right) \\
&= w_{1i}\sum_{i=1}^{N} \exp\left(-\sum_{m=1}^{M} \alpha_m y_i G_m\left(x_i\right)\right) \\
&= w_{1i}\prod_{m=1}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right) \\
&= Z_1\sum_{i=1}^{N} w_{2i}\prod_{m=2}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right) \\
&= Z_1 Z_2\sum_{i=1}^{N} w_{3i}\prod_{m=3}^{M} \exp\left(-\alpha_m y_i G_m\left(x_i\right)\right) \\
&= Z_1 Z_2 \cdots Z_{M-1}\sum_{i=1}^{N} w_{Mi}\left(-\alpha_M y_i G_M\left(x_i\right)\right) \\
&= \prod_{m=1}^{M} Z_m
\end{aligned} \tag{36}$$

The error upper bound of the component classifiers is $\prod_{m=1}^{M} Z_m$. It means that we can minimize Z_m by selecting moderate base classifier in every cycle, and make the training errors decrease moderately.

When it comes to binary classification, Equation (32) can be written as

$$\begin{aligned}
Z_m &= \sum_{y_i = G_m(x)} w_{mi} e^{-\alpha_m} + \sum_{y_i \neq G_m(x)} w_{mi} e^{\alpha_m} \\
&= \left(1 - e_m\right) e^{-\alpha_m} + e_m e^{\alpha_m} \\
&= 2\sqrt{e_m\left(1 - e_m\right)}
\end{aligned} \tag{37}$$

For the proposed Robust AdaBoost,

$$\begin{aligned} Z_m{}^* &= \sum_{y_i=G_m(x)} w_{mi} e^{-\alpha_m{}^*} + \sum_{y_i \neq G_m(x)} w_{mi} e^{\alpha_m{}^*} \\ &= (1-e_m)\, e^{-\alpha_m{}^*} + e_m e^{\alpha_m{}^*} \\ &= (1-e_m)\, e^{-\left(\ln \frac{2e}{1+e^e m}\right)^n} + e_m e^{\left(\ln \frac{2e}{1+e^e m}\right)^n} \\ &= (1-e_m)\, e^{-\left(\ln \frac{1+e^e m}{2e}\right)^n} + e_m e^{\left(\ln \frac{2e}{1+e^e m}\right)^n} \end{aligned} \quad . \tag{38}$$

Figure 8 shows the different Z_m changes with different errors of conventional AdaBoost and the proposed robust AdaBoost (n = 1, 3, 6, 9, 20, 100). Definitely, the goal of boosting is to achieve a series base classifiers with lower errors. When we focus on the Z_m with error less than 0.5, it is clear that the proposed robust AdaBoost is consistent with the boosting function, Furthermore, it is also appropriate for the entire domain of definition [0,1], which is more robust than conventional AdaBoost.

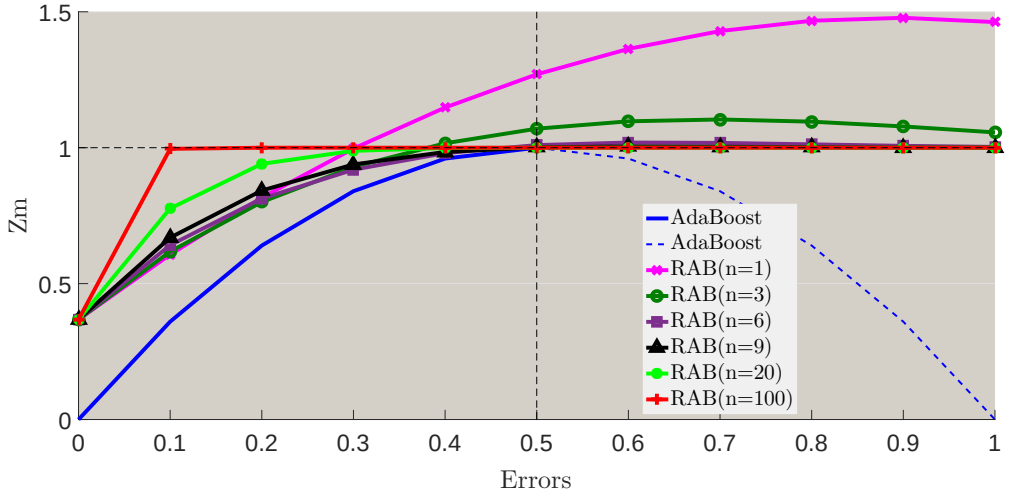


Figure 8. Weights normalization values of AdaBoost.

3.3. Feasibility Analysis for Choosing RBFSVM as Base Classifier

This paper aims at applying RBFSVM as component classifier in boosting. There are two vital parameters, Gaussian width σ and regularization parameter C, that are relevant to classification performance. The classification performance of RBFSVM can mainly depend on σ value with a roughly suitable C [47]. We can adjust RBFSVM to be an appropriate base classifier of boosting component classifiers by simply changing σ over a large range of C.

In [48], σ uses a large initial value σ_{ini} = 12 and decreases it slightly to increase the learning capacity of RBFSVM so that its error is less than 50%. In this paper, instead of adjusting errors to be less than 0.5, we focus on how to make these classifiers uncorrelated and improve the performance of every base classifier. Because of the robustness described in Section 3, our weight computing method can keep balance with errors in [0,1].

3.3.1. Diversity

In order to make the boosting function more efficient for the component classifiers, a roughly large σ is set initially. There is another important factor affecting the generation performance of ensemble methods called Diversity [49,50]. It is also appropriate in AdaBoost, so that these component classifiers have irrelevance and the boost function can make sense of boosting these same classifiers with different properties.

Similar to [48], the diversity is calculated as follows: If $h_{m,i}$ is the hypothesis of the mth SVM classifier on the sample x_i, and $f(x_i)$ is the combined prediction label of all the existing component classifiers, the diversity of the mth component classifier on the sample x_i is calculated as

$$d_{mi} = \begin{cases} 0, if I(h_{mi} = f(x_i)) = 1 \\ 1, if I(h_{mi} = f(x_i)) = 0 \end{cases}, \tag{39}$$

and the diversity value of RAB-RBFSVM in m th cycle for N samples is

$$DIV_m = \frac{1}{N} \sum_{i=1}^{N} d_{mi}. \tag{40}$$

Here we do not compare DIV_m with a preset threshold, as it is hard to find a single threshold for different training data. σ is decreased step by step to find a proper new σ^* that can minimize the error and maximize the diversity simultaneously. The step length of the decreasing σ is

$$\sigma_{step} = \frac{\sigma_{ini}}{50}. \tag{41}$$

3.3.2. Samples Processing: Denoising Initial Weights

With the update of σ, several RBFSVM classifiers using different σs are tested on the same samples in one cycle. From Equation (30), we know that the weights of misclassified samples will be increased. However, if a sample is misclassified too many times in one cycle, there is a large possibility that this sample is a noise of this training data, and its weight should be decreased relatively. The misclassified times of the sample should be inversely proportional to its weight, and the weights of misclassified samples as follows:

$$I_m = |I_{tr} - I_{tr,\max}|, \tag{42}$$

$$I_{m\_norm} = \frac{I_m}{\|I_m\|_2}, (I_{m\_norm} \in (0,1]) \tag{43}$$

$$w_{m,I_m} = \ln \frac{2e}{1 + e^{I_{m\_norm}}}, \left(w_{m,I_m} \in \left[\ln \frac{2e}{1+e}, 1\right)\right) \tag{44}$$

$$w_{m+1,i} = w_{m,I_m} \frac{w_{mi}}{Z_m} e^{(-\alpha_m^* y_i G_m(x_i))}. \tag{45}$$

where I_{tr} is the total times of misclassification in one cycle, and $I_{tr,\max}$ is the max value, Equations (42)–(44) calculate the possibility of classification of each sample w_{m,I_m}. $w_{m+1,i}$ is the final weight of samples in each cycle.

The conventional AdaBoost starts its boost with the same weights of samples, which means it does not take the balance of negative and positive samples into consideration in the first step. In RAB-RBFSVM+, the initial weights are calculated with different scales of negative and positive samples. For binary classification,

$$w_{1i}^n = \frac{1}{2N_n}, w_{1i}^p = \frac{1}{2N_p}, i = 1, 2, \cdots, N \tag{46}$$

N_n is the number of negative samples and N_p positive, w_{1i}^n and w_{1i}^p are the initial weights of negative and positive samples. Algorithm 1 shows the entire process of our proposed RAB-RBFSVM+.

Algorithm 1 Improved Robust AdaBoost classifier with RBFSVM: RAB-RBFSVM+.

Input: Training datasets $s = \{(x_i, y_i)\}_{i=1}^{m}$ with labels $y \in \{-1, +1\}$.

initialize: Normalize training datasets; Initialize weights of datasets via Equation (46), σ_{step} via Equation (41); $\sigma_{ini} \leftarrow 12$;$m \leftarrow 1$; $err \leftarrow -1$; $N \leftarrow 100$; $I_{0\_tr} \leftarrow I$;

while $m < N$ & $err \neq 0$ **do**

Train a RBFSVM base classifier ht on the weighted training datasets and get the misclassification flag of samples I_m.

Compute training errors using Equation (28)

Compute diversity value using Equations (39) and (40)

Compute misclassification times of samples: $I_{m\_tr} = I_{m-1\_tr} + I_m$

Compute possibility of noise samples via Equations (42)–(44)

Decrease σ by σ_m

if $\sigma > \sigma_{step}$ & $err \neq 0$ **then**

Catch the right σ which minimize the error: σ_m, and go to step 3

end if

$\sigma = \sigma_m$

Repeat step 3-7

Set the weight of component classifier via Equation (34)

Update the weight of samples via Equation (45)

end while

Output: Combine classifiers via Equations (32) and (33)

4. Experimental Results

In this section, we evaluate the performance of Robust AdaBoost (RAB) and the improved Robust AdaBoost RBFSVM (RAB-RBFSVM+). We compare the proposed algorithm with the conventional AdaBoost based component classifiers using other weak classifiers. The data sets from UCID Repository [51] are utilized to verify the effectiveness of the proposed robust AdaBoost classifier.

Different feature combinations are also compared using different algorithms we test above. The smoke and non-smoke videos are collected from public video clips (The related video clips can be downloaded from http://signal.ee.bilkent.edu.tr/VisiFire/Demo/SampleClips and http://imagelab.ing.unimore.it/visor). The smoke clips in this data sets cover indoor and outdoor with different illumination, short or long distance surveillance scenes. Seven smoke videos and ten non-smoke videos are used in the experiment. In the training datasets of smoke, we select or reprocess the smoke videos so that they only have smoke motion regions. In this paper, we extracted 5699 motion regions (include 2766 positive samples and 2933 negative samples) for training and test. Finally, we test three videos with smoke, non-smoke or mixed sequences.

4.1. Advantages of the Proposed Classifier

According Equation (34) and Figure 6, the weight parameter $\alpha_m{}^*$ has different domain of values with different n in $e_m \in [0, 1]$, as shown in Table 2.

Table 2. $\alpha_m{}^*$changes with different n.

n	$\alpha_m{}^* \in$	$\alpha_m{}^*$ $(e_m = 0.5)$
1	$[0.38, 1]$	0.7191
3	$[0.05, 1]$	0.3718
6	$[3.01 \times 10^{-3}, 1]$	0.1383
9	$[1.65 \times 10^{-4}, 1]$	0.0514
20	$[3.92 \times 10^{-5}, 1]$	0.0001
100	$[9.27 \times 10^{-43}, 1]$	4.77×10^{-15}
1000	$(0, 1]$	0

Theoretically, we expect that the well-performed classifier with lower errors could have higher weight while the poor one has roughly lower weight, so that the performance of AdaBoost could depend much more on these well performed classifiers, therefore, we appropriate a large n. However, we also prefer to have a robust algorithm for classifier with large errors rather than discard it directly, when n is too large, ${\alpha_m}^*$ will close to zero with large error and its computing complexity will increase too, it is unnecessary and not rational to use an extreme large n. Table 2 shows the range changes of ${\alpha_m}^*$ with respect to n, when $n = 1$, the ${\alpha_m}^*$ changes from a relatively large value to 1, and when $e_m = 0.5$, the value of ${\alpha_m}^*$ is also large, it is lack of effectiveness for boosting well-performed classifiers with $e_m \ll 0.5$. Additionally, when $n > 6$, classifier with large errors will be discarded directly since its weight is too small. These ns in Table 2 are tested and the results are shown in Figure 9. When the number of cycles is large, their differences are not very distinct. When n is less than 10 ($n = 1, 3, 6, 9$) in left subfigure, the right subfigure of Figure 9 demonstrates that the performance degrades when $n = 100$ or $n = 1000$. In this paper, we set $n = 3$.

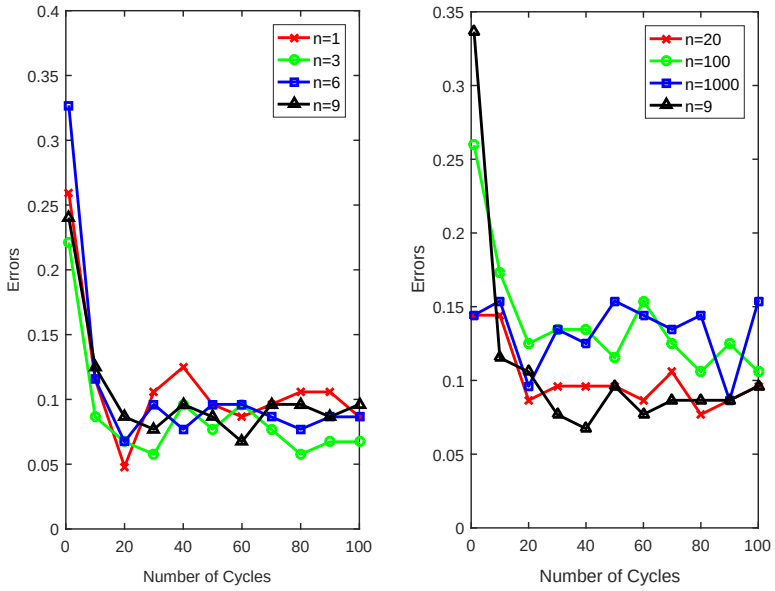


Figure 9. Classification errors using different ${\alpha_m}^*$ with different ns.

The base classifiers are listed as KNN (K-Nearest Neighbor), Fisher Linear Discriminant, Naive Bayes classifier, DTC(decision tree classifier) and SVM (Support Vector Machines), and RBFSVM, there are no further improved methods used except Robust AdaBoost and the initial σ is set as $\sigma_{ini} = 12$ for RBFSVM in this section. Figure 10 shows the performances of the two AdaBoost methods. AB represents AdaBoost method, and RAB represents Robust AdaBoost method. The results of the 50th, the 100th, the 150th, and the 200th cycles are listed in Table 3.

Table 3. Errors of AdaBoost and Robust AdaBoost with different component classifiers (Bold numbers show the lowest errors of AdaBoost, Robust AdaBoost and classifier without boosting).

Base Classifier	AB				RAB				No Boosting
	$N = 50$	$N = 100$	$N = 150$	$N = 200$	$N = 50$	$N = 100$	$N = 150$	$N = 200$	
KNN	0.1346	0.1442	0.1346	0.1538	**0.0865**	0.1442	0.1538	0.1538	**0.0865**
Fisher	0.1635	0.1827	0.1923	**0.1442**	0.2019	0.2019	0.1923	0.1538	0.2596
Naive Bayes	0.1923	0.1635	**0.1538**	**0.1538**	0.1635	0.1731	0.1635	0.1923	0.2692
Decision Tree	0.1538	0.1635	0.1538	0.1538	**0.1346**	0.1538	0.1538	0.1442	0.2500
Linear SVM	0.2596	0.2788	0.6154	**0.2019**	**0.2019**	0.2212	**0.2019**	**0.2019**	0.2596
RBFSVM	0.5962	0.5962	0.5962	0.2308	**0.1058**	0.1442	0.1250	0.1250	0.2019

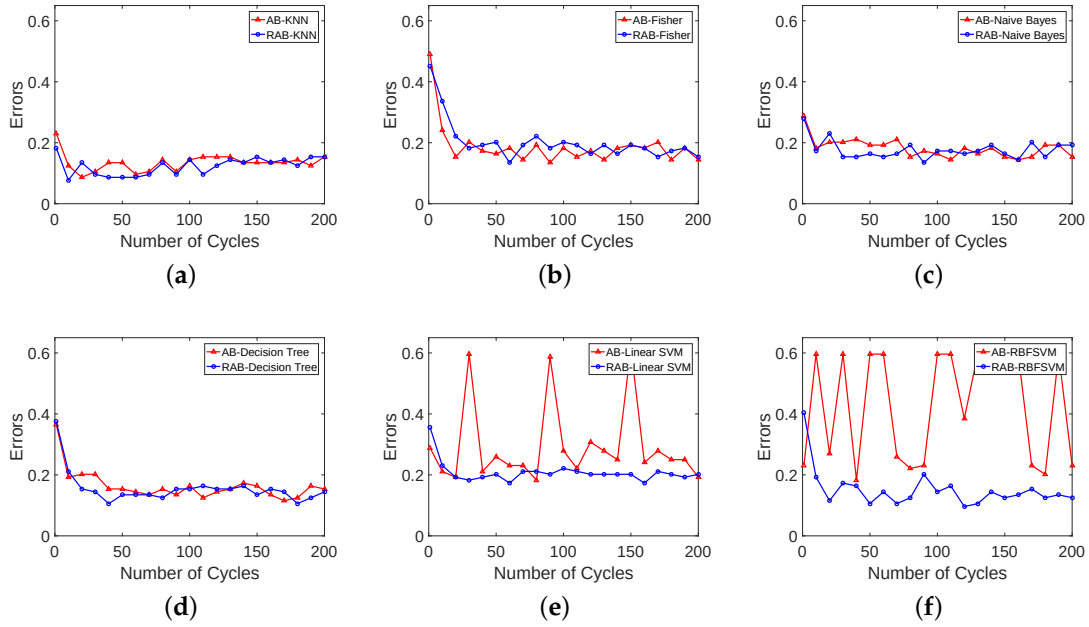


Figure 10. Performances of AdaBoost and Robust AdaBoost with different component classifiers. (**a**) Training errors of AB-KNN and RAB-KNN; (**b**) Training errors of AB-Fisher and RAB-Fisher; (**c**) Training errors of AB-Naive Bayes and RAB Naive Bayes; (**d**) Training errors of AB-Decision Tree and RAB-Decision Tree; (**e**) Training errors of AB-Linear SVM and RAB-Linear SVM; (**f**) Training errors of AB-RBFSVM and RAB-RBFSVM.

For Fisher Linear Discriminant, Robust AdaBoost performs as well as AdaBoost. For Naive Bayes, KNN, and Decision Tree classifiers, the Robust AdaBoost gives a little better boosting performance than AdaBoost, and both of them perform better than a single classifier. For Linear SVM and RBFSVM, the performance of Robust AdaBoost is much better than AdaBoost with lower errors and more stable changes. It is clear that the normal AdaBoost cannot boost SVM classifiers. The proposed RAB gives a better boosting result. Table 3 presents that Robust AdaBoost based RBFSVM component classifiers has the best result and it is also more efficient than the other classifiers without boosting.

Figure 11 shows the error changes of three different RAB-RBFSVM methods: RAB-RBFSVM without any improvements except the Robust AdaBoost; RAB-RBFSVM with improvements of diversity (RAB-RBFSVM−), and RAB-RBFSVM with both improvements of diversity and sample processing (RAB-RBFSVM+). The results 50th, 100th, 200th, 300th, 400th, 500th cycle are shown in Table 4.

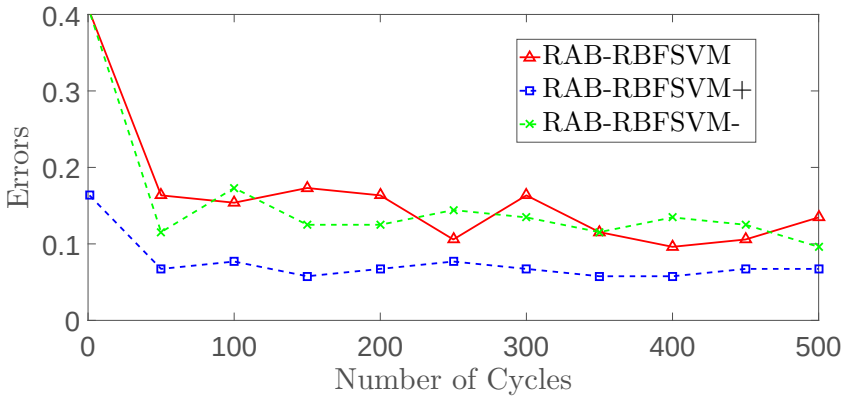


Figure 11. Performances of RAB-RBFSVM and Performances of RAB-RBFSVM, RAB-RBFSVM− and RAB-RBFSVM+

Table 4. Errors of RAB-RBFSVM and RAB-RBFSVM+ in different numbers of cycles.

Component Classifiers	RAB-RBFSVM	RAB-RBFSVM−	RAB-RBFSVM+
$N = 50$	0.1635	0.1154	0.0673
$N = 100$	0.1538	0.1731	0.0769
$N = 200$	0.1635	0.1250	0.0673
$N = 300$	0.1635	0.1346	0.0673
$N = 400$	0.0962	0.1346	0.0577
$N = 500$	0.1346	0.0962	0.0673

From Figure 10 and Table 4, RAB-RBFSVM− method with diversity performs better than RAB-RBFSVM, and our proposed method, RAB-RBFSVM+ with improvements of diversity and samples processing, achieves the best results with more stable changes and lower errors, which indicates the effectiveness of these improvements for RAB-RBFSVM.

In order to compare the proposed classifier with other different classifiers using common datasets from UCID, we test AdaBoost with KNN (K-Nearest Neighbor)component classifiers, (AB-KNN), AdaBoost with Fisher Linear Discriminant component classifiers, (AB-FLD), AdaBoost with Decision Tree component classifiers, (AB-DT), AdaBoost with Naive Bayes component classifiers, (AB-NB), AdaBoost with Linear SVM component classifiers, (AB-SVM), Diverse AdaBoostSVM, (DAB-SVM), our proposed Robust AdaBoost RBFSVM (RAB-RBFSVM+), in addition to these component classifiers, single RBFSVM classifier and linear SVM classifier are also compared. Table 5 shows the generation errors of different classifiers. Note that the proposed method generates lower errors on the datasets.

Table 5. Generation errors of different classifiers: AB-KNN, AB-FLD, AB-DT, AB-NB, AB-SVM, DAB-SVM, RAB-RBFSVM+ (For every single dataset, bold number gives the lowest error while underline number gives the second lowest error).

Datasets	AB-KNN	AB-FLD	AB-DT	AB-NB	AB-SVM	DAB-RBFSVM	RAB-RBFSVM+	RBFSVM	Linear SVM
Parkinsons	<u>0.0928</u>	0.2371	0.2680	0.3196	0.1753	0.1340	**0.0619**	0.2680	0.2577
Ionosphere	0.1657	0.2286	0.1314	<u>0.1086</u>	0.1771	0.1371	**0.0629**	0.1314	0.2000
Vote	0.0922	0.0737	0.0829	<u>0.0645</u>	0.0783	<u>0.0645</u>	**0.0599**	0.0737	**0.0599**
Sonar	0.1346	0.2212	<u>0.1731</u>	0.2212	0.2404	0.2019	**0.1346**	0.2019	0.2500
Wdbc	0.0704	0.0669	**0.0141**	0.0423	0.0634	0.0317	0.0282	<u>0.0211</u>	0.0317
Wpbc	0.4242	0.2626	0.2626	0.3232	**0.2424**	<u>0.2525</u>	0.2525	0.3232	**0.2424**
Mean	0.1400	0.1557	0.1332	0.1542	0.1396	<u>0.1174</u>	**0.0857**	0.1456	0.1488

4.2. Performance Advantages on Fire Smoke Detector

In this section, we will compare our proposed RAB-RBFSVM+ with other methods using our extracted fire smoke features in different combinations. Different static and dynamic features are extracted as described in Section 2. Extensive experiments are carried out to test these features in different combinations, and the effectiveness of our proposed method. For brevity, the static features are simply marked with their initials (Color-'C', Texture-'T', Wavelet energy-'W', Edge orientation histogram-'E', irregularity-'I' and sparsity-'S'), dynamic features are marked as 'MD' (Motion direction), 'MDC' (change of motion direction) and 'MS' (motion speed), 'Static', 'Dynamic' and 'All' represent all the static features, dynamic features and all features combinations separately. We divide them into six groups according to the combination rules.

The proposed RAB-RBFSVM+ method outperforms all other methods tested as shown in Figure 12. It generates lower errors along with more stable performance for all feature combinations. Moreover, the proposed classifier with single static features gives better performance than single dynamic features in Figure 12a.

For further comparison, we test feature combinations with the proposed RAB-RBFSVM+ as shown in Figure 12b–f to find out which feature combination can achieve more satisfied performance for fire smoke detection. We list the experimental statistics in Table 6. The proposed detector using CES+MD+MS, TCW+MDC and TCW+MDC+MS feature combinations generate the three lowest errors.

The feature combinations are chosen for application to fire smoke detection. Different videos with or without fire smoke objects are tested. For comparison, the following measures are considered,

$$\text{accuracy} = \frac{TP + TN}{N_{pos} + N_{neg}}, \tag{47}$$

$$\text{detection rate} = \frac{TP}{TP + FN}, \tag{48}$$

$$\text{false alarm rate} = \frac{FP}{FP + TN}, \tag{49}$$

where TP is the number of truth positives which gives the number of fire regions classified as fire, TN is the number of truth negatives, i.e., the number of non-fire regions which are classified as non-fire, FP is the number of false positives, i.e., the number of non-fire regions classified as fire, and FN is the number of false negatives, i.e., the number of fire regions classified as non-fire. N_pos and N_neg are the total number of positives and negatives respectively. We compare the proposed method with different feature combinations and other two methods, [29,34], the rates described above are listed in Table 7. Our proposed method can achieve higher accuracy and detection rate along with lower false alarm rate as shown in Table 7.

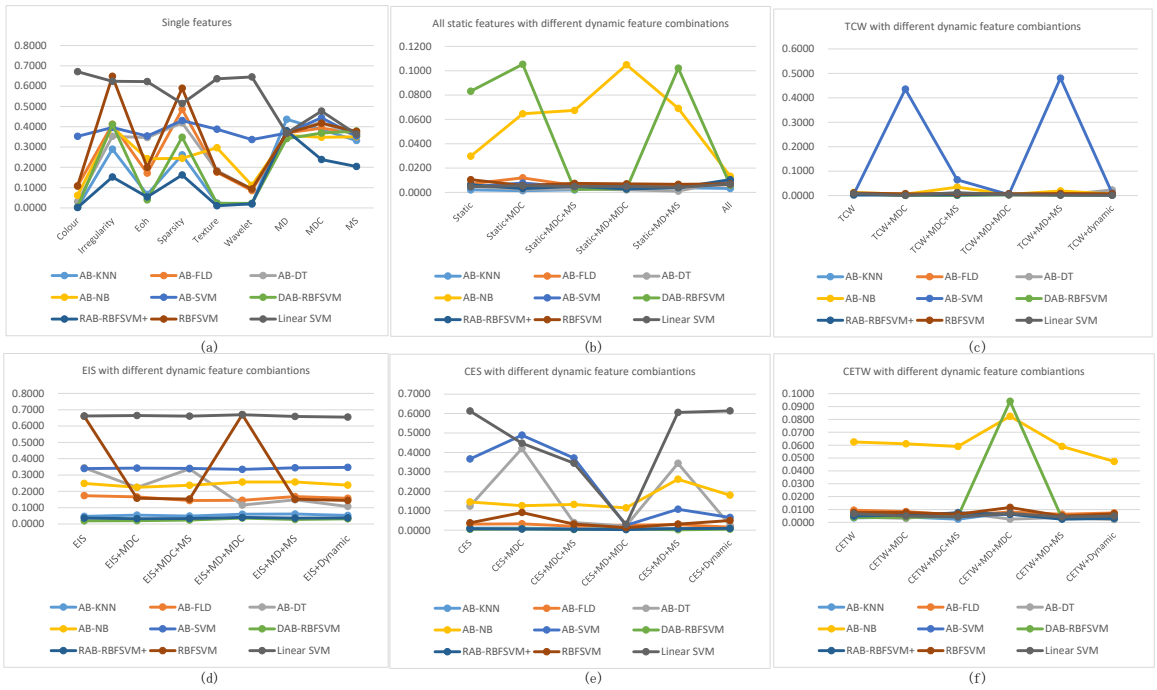


Figure 12. Generation errors of different classifiers with different feature datasets.

Table 6. Errors of RAB-RBFSVM+ with different fire smoke features.

	CES	EIS	Static	TCW	CETW
+[ ]	0.0039	0.0193	0.0832	0.0056	0.0039
+MDC	0.0067	0.0207	0.1053	0.0014	0.0039
+MDC+MS	0.0032	0.0242	0.0025	**0.0011**	0.0046
+MD+MDC	0.0067	0.0362	0.0028	0.0032	0.0941
cc +MD+MS	0.0017	0.0277	0.1021	0.0018	0.0032
+Dynamic	0.0046	0.0316	0.0060	0.0018	0.0032

Table 7. Accuracy of our proposed method with different feature combinations, [29,34] (%).

Methods	Accuracy	Detection Rate	False Alarm Rate
Cai et al. [29]	74.10	92.70	7.30
Wu et al. [34]	73.41	94.48	5.52
AB-RBFSVM+ (with CES+MD+MS)	88.28	99.52	0.48
AB-RBFSVM+ (withTCW+MDC)	**93.31**	99.30	0.70
AB-RBFSVM+ (with TCW+MDC+MS)	91.25	**99.69**	**0.31**

Figure 13 shows the performance of our proposed detection method with TCW+MDC+MS features. Red boxes give the detected smoke or fire regions while green boxes give the motion regions without smoke or fire. There is no false smoke window detected by the proposed method in waste basket video and cars video, and small number of false smoke windows enclosing shaking trees in the last video due to the small size of these objects and low image quality of the noisy video.

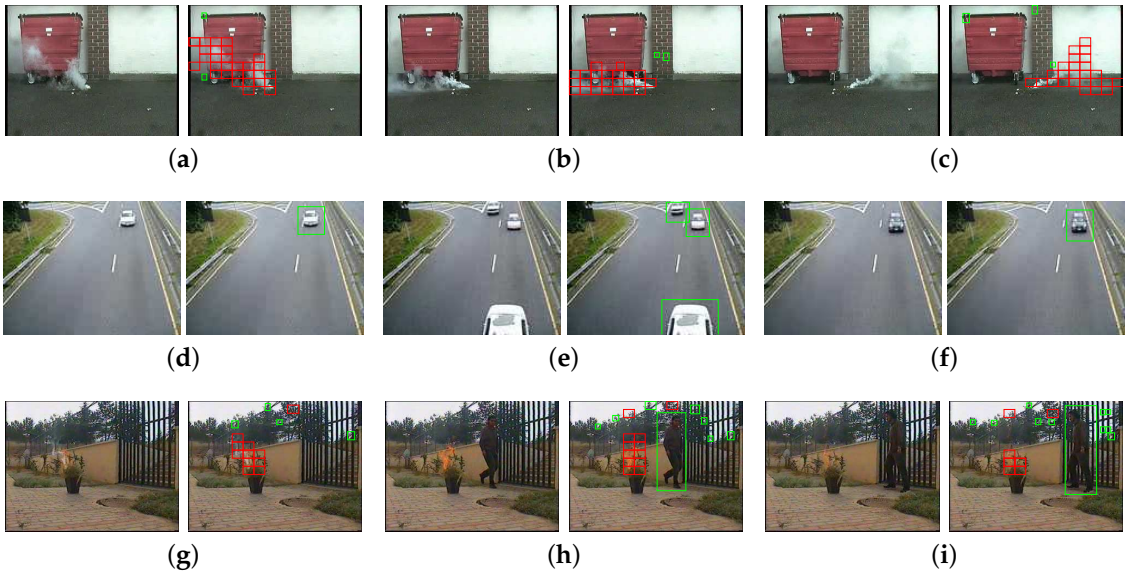


Figure 13. Detection results in three videos with or without smoke-fire. Smoke video with waste basket at frame (**a**) 121 (**b**) 166 and (**c**) 326. non-smoke video with cars on high way at frame (**d**) 11 (**e**) 26 and (**f**) 106. smoke-fire video with pedestrian at frame (**g**) 61 (**h**) 228 and (**i**) 244. Left image of each sub figure is the original frame image while the right one is the detected result.

5. Conclusions

This paper proposes a new fire smoke detector based on videos with Robust AdaBoost (RAB-RBFSVM) classifier. Features are extracted from different videos take by cameras. Static and dynamic features in different combinations are tested for fire smoke detection. The proposed classifier gives an efficient way of solving dilemma between errors and weights computing of AdaBoost. Further improvements including diversity and samples denoising for keeping balance of positive and negative training datasets are introduced to improve the performance of our detector. Tests with common datasets UCID and the extracted features from public video clips indicate that our proposed classifier outperforms the other classifiers. Extensive experiments on fire smoke detection are conducted and the results indicate effectiveness and performance advantages of our proposed fire smoke detector.

Author Contributions: All author have made significant contributions to this work. X.W. and X.L. proposed the structure of the paper. X.W. realized the algorithm and wrote the draft. H.L. gave important guiding advices to the implementation and the experiment.

Funding: This work was supported by the National Natural Science Foundation of China (No. 61871123), Key Research and Development Program in Jiangsu Province (No. BE2016739) and a Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

Acknowledgments: The authors would like to thank the editor and the anonymous reviewers for their valuable and helpful comments, as well as the important guiding significance to our researches.

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

References

1. Fonollosa, J.; Solórzano, A.; Marco, S. Chemical Sensor Systems and Associated Algorithms for Fire Detection: A Review. *Sensors* **2018**, *18*, 553. [CrossRef] [PubMed]
2. Adib, M.; Eckstein, R.; Hernandez-Sosa, G. SnO2 Nanowire-Based Aerosol Jet Printed Electronic Nose as Fire Detector. *IEEE Sens. J.* **2018**, *18*, 494–500. [CrossRef]
3. Cheon, J.; Lee, J.; Lee, I.; Chae, Y.; Yoo, Y.; Han, G. A Single-Chip CMOS Smoke and Temperature Sensor for an Intelligent Fire Detector. *IEEE Sens. J.* **2009**, *9*, 914–921. [CrossRef]
4. Soliman, H.; Sudan, K.; Mishra, A. A smart forest-fire early detection sensory system: Another approach of utilizing wireless sensor and neural networks. In Proceedings of the 2010 IEEE Sensors, Pittsburgh, PA, USA, 1–4 November 2010; pp. 1900–1904.
5. Millangarcia, L.; Sanchezperez, G.; Nakano, M.; Toscanomedina, K.; Perezmeana, H.; Rojascardenas, L. An Early Fire Detection Algorithm Using IP Cameras. *Sensors* **2012**, *12*, 5670–5686. [CrossRef] [PubMed]
6. Gottuk, D.T.; Lynch, J.A.; Rose-Pehrsson, S.L.; Owrutsky, J.; Williams, F.W. Video image fire detection for shipboard use. *Fire Saf. J.* **2006**, *41*, 321–326. [CrossRef]
7. Kastrinaki, V.; Zervakis, M.; Kalaitzakis, K. A survey of video processing techniques for traffic applications. *Image Vis. Comput.* **2003**, *21*, 359–381. [CrossRef]
8. Shah, M.; Javed, O.; Shafique, K. Automated visual surveillance in realistic scenarios. *IEEE MultiMedia* **2007**, *14*, 30–39. [CrossRef]
9. Braovic, M.; Stipanicev, D.; Krstinic, D. Cogent Confabulation based Expert System for Segmentation and Classification of Natural Landscape Images. *Adv. Electr. Comput. Eng.* **2017**, *17*, 85–94. [CrossRef]
10. Xiong, Z.; Caballero, R.; Wang, H.; Finn, A.M.; Lelic, M.A.; Peng, P.Y. Video-based smoke detection: Possibilities, techniques, and challenges. In Proceedings of the IFPA, Fire Suppression and Detection Research and Applications—A Technical Working Conference (SUPDET), Orlando, FL, USA, 5–8 March 2007.
11. Phillips Iii, W.; Shah, M.; da Vitoria Lobo, N. Flame recognition in video. *Pattern Recogn. Lett.* **2002**, *23*, 319–327. [CrossRef]
12. Straumann, W.; Rizzotti, D.; Schibli, N. Method and Device for Detecting Fires Based on Image Analysis. European Patent EP 1,364,351, 26 February 2002. Available online: https://patents.google.com/patent/EP1364351B1/en (accessed on 2 November 2018).
13. Chen, T.H.; Wu, P.H.; Chiou, Y.C. An early fire-detection method based on image processing. In Proceedings of the 2004 International Conference on Image Processing, ICIP'04, Singapore, 24–27 October 2004; Volume 3, pp. 1707–1710.
14. Toreyin, B.U.; Dedeoglu, Y.; Cetin, A.E. Flame detection in video using hidden markov models. In Proceedings of the IEEE International Conference on Image Processing, ICIP 2005, Genoa, Italy, 11–14 September 2005; Volume 2, p. II-1230.
15. Töreyin, B.U.; Dedeoğlu, Y.; Güdükbay, U.; Cetin, A.E. Computer vision based method for real-time fire and flame detection. *Pattern Recogn. Lett.* **2006**, *27*, 49–58. [CrossRef]
16. Chunyu, Y.; Jun, F.; Jinjun, W.; Yongming, Z. Video fire smoke detection using motion and color features. *Fire Technol.* **2010**, *46*, 651–663. [CrossRef]
17. Calderara, S.; Piccinini, P.; Cucchiara, R. Vision based smoke detection system using image energy and color information. *Mach. Vis. Appl.* **2011**, *22*, 705–719. [CrossRef]
18. Toreyin, B.U.; Dedeoglu, Y.; Cetin, A.E. Contour based smoke detection in video using wavelets. In Proceedings of the 2006 14th European Signal Processing Conference, Florence, Italy, 4–8 September 2006; pp. 1–5.

19. Çelik, T.; Özkaramanlı, H.; Demirel, H. Fire and smoke detection without sensors: Image processing based approach. In Proceedings of the 2007 15th European Signal Processing Conference, Poznan, Poland, 3–7 September 2007; pp. 1794–1798.
20. Surit, S.; Chatwiriya, W. Forest fire smoke detection in video based on digital image processing approach with static and dynamic characteristic analysis. In Proceedings of the 2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI), Jeju Island, Korea, 23–25 May 2011; pp. 35–39.
21. Long, C.; Zhao, J.; Han, S.; Xiong, L.; Yuan, Z.; Huang, J.; Gao, W. Transmission: A new feature for computer vision based smoke detection. In Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence, Sanya, China, 23–24 October 2010; pp. 389–396.
22. Tian, H.; Li, W.; Wang, L.; Ogunbona, P. Smoke detection in video: An image separation approach. *Int. J. Comput. Vis.* **2014**, *106*, 192–209. [CrossRef]
23. Yuan, F. A fast accumulative motion orientation model based on integral image for video smoke detection. *Pattern Recogn. Lett.* **2008**, *29*, 925–932. [CrossRef]
24. Yuan, F. A double mapping framework for extraction of shape-invariant features based on multi-scale partitions with AdaBoost for video smoke detection. *Pattern Recogn.* **2012**, *45*, 4326–4336. [CrossRef]
25. Zhao, Y.; Zhou, Z.; Xu, M. Forest fire smoke video detection using spatiotemporal and dynamic texture features. *J. Electr. Comput. Eng.* **2015**, *2015*, 40. [CrossRef]
26. Ye, W.; Zhao, J.; Wang, S.; Wang, Y.; Zhang, D.; Yuan, Z. Dynamic texture based smoke detection using Surfacelet transform and HMT model. *Fire Saf. J.* **2015**, *73*, 91–101. [CrossRef]
27. Tung, T.X.; Kim, J.M. An effective four-stage smoke-detection algorithm using video images for early fire-alarm systems. *Fire Saf. J.* **2011**, *46*, 276–282. [CrossRef]
28. Zhao, J.; Zhang, Z.; Han, S.; Qu, C.; Yuan, Z.; Zhang, D. SVM based forest fire detection using static and dynamic features. *Comput. Sci. Inf. Syst.* **2011**, *8*, 821–841. [CrossRef]
29. Cai, M.; Lu, X.; Wu, X.; Feng, Y. Intelligent video analysis-based forest fires smoke detection algorithms. In Proceedings of the 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, China, 13–15 August 2016; pp. 1504–1508.
30. Zhao, Y.; Li, Q.; Gu, Z. Early smoke detection of forest fire video using CS Adaboost algorithm. *Optik-Int. J. Light Electron Opt.* **2015**, *126*, 2121–2124. [CrossRef]
31. Yuan, F.; Fang, Z.; Wu, S.; Yang, Y. Real-time image smoke detection using staircase searching-based dual threshold AdaBoost and dynamic analysis. *Image Process. Iet* **2015**, *9*, 849–856. [CrossRef]
32. Zhang, Q.; Xu, J.; Xu, L.; Guo, H. Deep convolutional neural networks for forest fire detection. In Proceedings of the 2016 International Forum on Management, Education and Information Technology Application, Guangzhou, China, 30–31 January 2016.
33. Muhammad, K.; Ahmad, J.; Baik, S.W. Early Fire Detection using Convolutional Neural Networks during Surveillance for Effective Disaster Management. *Neurocomputing* **2017**, *288*, 30–42. [CrossRef]
34. Wu, X.; Lu, X.; Leung, H. An adaptive threshold deep learning method for fire and smoke detection. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 1954–1959.
35. Steinwart, I. Sparseness of support vector machines. *J. Mach. Learn. Res.* **2003**, *4*, 1071–1105.
36. Maji, S.; Berg, A.C.; Malik, J. Classification using intersection kernel support vector machines is efficient. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, CVPR 2008, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
37. Ring, M.; Eskofier, B.M. An approximation of the Gaussian RBF kernel for efficient classification with SVMs. *Pattern Recogn. Lett.* **2016**, *84*, 107–113. [CrossRef]
38. Zheng, S.; Liu, J.; Tian, J. An efficient star acquisition method based on SVM with mixtures of kernels. *Pattern Recogn. Lett.* **2005**, *26*, 147–165. [CrossRef]
39. Scholkopf, B.; Sung, K.K.; Burges, C.J.; Girosi, F.; Niyogi, P.; Poggio, T.; Vapnik, V. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Trans. Signal Process.* **1997**, *45*, 2758–2765. [CrossRef]
40. Bugarić, M.; Jakovčević, T.; Stipaničev, D. Adaptive estimation of visual smoke detection parameters based on spatial data and fire risk index. *Comput. Vis. Image Underst.* **2014**, *118*, 184–196. [CrossRef]

41. Shu, X.; Porikli, F.; Ahuja, N. Robust orthonormal subspace learning: Efficient recovery of corrupted low-rank matrices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 3874–3881.
42. Levi, K.; Weiss, Y. Learning object detection from a small number of examples: The importance of good features. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, Washington, DC, USA, 27 June–2 July 2004; Volume 2, p. II.
43. Stricker, M.A.; Orengo, M. Similarity of color images. In Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases III, Birmingham, UK, 11–14 September 1995; Volume 2420, pp. 381–392.
44. Tian, H.; Li, W.; Wang, L.; Ogunbona, P. A novel video-based smoke detection method using image separation. In Proceedings of the 2012 IEEE International Conference on Multimedia and Expo (Icme), Melbourne, Australia, 9–13 July 2012; pp. 532–537.
45. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
46. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893.
47. Valentini, G.; Dietterich, T.G. Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. *J. Mach. Learn. Res.* **2004**, *5*, 725–775.
48. Li, X.; Wang, L.; Sung, E. AdaBoost with SVM-based component classifiers. *Eng. Appl. Artif. Intell.* **2008**, *21*, 785–795. [CrossRef]
49. Melville, P.; Mooney, R.J. Creating diversity in ensembles using artificial data. *Inf. Fusion* **2005**, *6*, 99–111. [CrossRef]
50. Kuncheva, L.I.; Whitaker, C.J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach. Learn.* **2003**, *51*, 181–207. [CrossRef]
51. Lichman, M. *UCI Machine Learning Repository*; University of California: Oakland, CA, USA, 2013.