Multimed Tools Appl (2015) 74:689–705
DOI 10.1007/s11042-014-2106-z

Towards a solid solution of real-time fire and flame detection

Bo Jiang · Yongyi Lu · Xiying Li · Liang Lin

Received: 30 September 2013 / Revised: 7 March 2014 / Accepted: 14 April 2014 /
Published online: 4 July 2014

Abstract Although the object detection and recognition has received growing attention for decades, a robust fire and flame detection method is rarely explored. This paper presents an empirical study, towards a general and solid approach for fast detection of fire and flame in videos, with the applications in video surveillance and event retrieval. Our system consists of three cascaded steps: (1) candidate regions proposing by a background model, (2) fire region classifying with color-texture features and a dictionary of visual words, and (3) temporal verifying. The experimental evaluation and analysis are done for each step. We believe that it is a useful service to both academic research and real-world application. In addition, we release the software of the proposed system with the source code, as well as a public benchmark and data set, including 64 video clips covered both indoor and outdoor scenes under different conditions. We achieve an 82 % Recall with 93 % Precision on the data set, and greatly improve the performance by state-of-the-arts methods.

Keywords Fire detection · Empirical study · Video surveillance · Region classification

This work was supported by Fundamental Science and Technology Program of Ministry of Public Security (no. 2013GABJC013), Program of Guangzhou Zhujiang Star of Science and Technology (no. 2013J2200067), Guangdong Science and Technology Program (no. 2012B031500006), Guangdong Natural Science Foundation (no. S2013050014548), Special Project on Integration of Industry, Education and Research of Guangdong Province (no. 2012B091000101) and Fundamental Research Funds for the Central Universities (no. 13lgjc26).

B. Jiang · Y. Lu · X. Li (✉) · L. Lin
School of Engineering, Sun Yat-sen University, Guangzhou, China
e-mail: stslxy@mail.sysu.edu.cn

L. Lin
e-mail: linlng@mail.sysu.edu.cn

L. Lin
SYSU-CMU Shunde International Joint Research Institute, Shunde, China

1 Introduction

Fire catastrophes always cause big lost to human kind. For example, Dhaka fire on June 2010 caused over 117 people died and over 100 injured. Forest fires in Russia in summer 2010 caused over 10 billion US dollars lost. Therefore, robust early-warning fire alarm systems are always in need to protect people's safety and properties. A fast and accurate fire detection method is the key technology needed in such systems. As a result, fire or flame detection has received great attention by research communities for decades [16, 32]. In recent years, fire detection also find applications in event retrieval on fire in digital image or video archives [3].

Existing fire detection methods can be classified into two categories: sensor-based methods and vision-based methods. Sensor-based methods employed information captured by special instruments such as infra-red sensors and smoke sensors, which are expensive and not easy to get [27]. Moreover, those sensors typically detect the presence of certain particles generated by fire ionisation or photometry rather than the combustion itself. Therefore, fire-alarm systems based on those sensors usually have time delay and result in high false rates [29]. In contrast, vision-based methods using the normal R-G-B images or videos have several advantages [13, 16, 32]. Firstly, cameras are becoming more functional with steady dropping price. Thus images and videos can provide more detailed visual information for fire detection cheaply. Moreover, surveillance cameras already installed in public places can also be used for fire detection. Secondly, the response time can be faster than traditional sensors, as cameras do not need to wait for the smoke or heat to diffuse. Finally, compared to traditional point sensors, cameras can monitor a broader area, creating a higher possibility of fire detection at early stage [16]. Our system presented in this paper belongs to the latter category, and aims to detect fire in video clips.

Despite of the growing needs and interests in fire detection, there is still not a large number of work on fire detection in the computer vision literature [3]. Building a robust fire detection system is challenging in the following two aspects: (1) fire or flame is flexible in shape and intensity, and it has no fixed structure or appearance. Though color is a relative distinct feature widely used for fire detection, the fire color appeared in images and videos is affected by the camera quality (e.g. resolution, sharpness) and settings (e.g. white balance). The fire also appears widely various in scale. Figure 1 shows six fire images with various

Fig. 1 Various appearances of fire

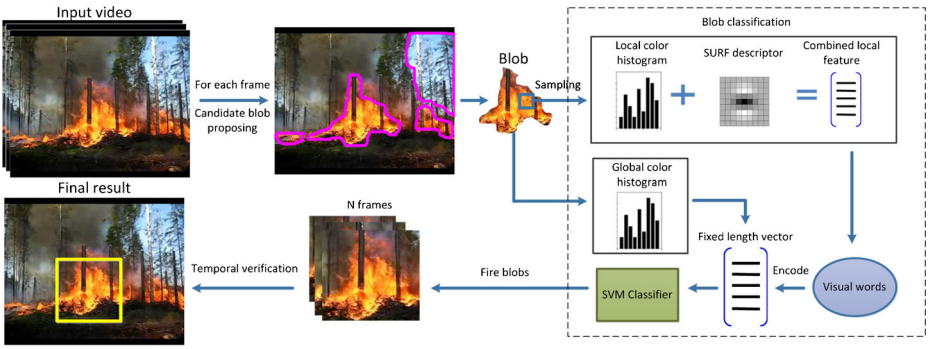


Fig. 2 Framework of proposed fire detection system

shapes, intensities, colors, and scales. Thus, it is hard to build a solid and generic model for fire. And (2) real application requires a fast fire detection method that can work in real-time. Complicated methods or models can not find a position in real applications. Some fire detection methods use filter banks [32], frequency transforms [23], and motion tracking techniques, thus need more computational processing time and are not suitable for real-time applications [3].

There are mainly four kinds of limitations in existing fire detection systems: (1) merely based on color, or color plus motion information. Such as [13] and [4] only used color to detect fire. Limited clues may lead to high false alarm. (2) merely worked for some special situations, such as tunnel fire [18]. (3) used many heuristic fixed thresholds, such as [7], [34] and [4], which restricted their applications. And (4) tested on limited data set, such as Cho et al. [8] tested their method on six fire video clips, and Ko et al. [16] performed their experiments on twelve video clips. Though they showed good results on the small data set, their systems may be impractical for other untested situation. Furthermore, as far as we know, there are no standard fire data sets.

Above mention four limitations lead to the big gap in real-world fire detection applications. The method we proposed in this paper is expected to narrow the gap. We have randomly downloaded 64 fire video clips from public video sharing web sites.[1] This data set is much larger than those used by other researchers. We carried out a comprehensive empirical study and experiments on this data set. Based on the experiment results, we adopt three most effective fire features, including a local and a global color descriptor in CIE Lab space, and a SURF (Speeded Up Robust Features [2]) texture descriptor. Then a Support Vector Machine (SVM) is employed to find fire regions. Based on our experiment, the RBF(Radial Basis Function) SVM kernel works the best for fire detection. Different with most existing fire detection methods which used only local features on pixel level, we find that our proposed global color feature and local SURF texture feature on regions made a good contribution for fire detection. Finally, a temporal verification is applied to reduce the false alarm. The flowchart of our system is shown in Fig. 2. Our system does not use any complicated features or tools, thus it is fast and can detect fire in real-time. Compared to Toreyin and Cetin's method proposed in [32], ours has higher recall and precision in fire detection on our large testing data set. The remainder of this paper is organized as follows.

[1]Project page. The data set and software of the proposed system can be downloaded from the web page: http://vision.sysu.edu.cn/systems/fire-detection/

Section 2 discusses the detailed implementation of our system based on empirical study. Section 3 introduces the framework of our fire detection system. Section 5 presents our experiment results. Comparisons with Toreyin and Cetin's method proposed in [32] are also given in this section. Finally Section 6 summarizes this paper and draws the conclusion.

2 Empirical study and feature selection

Our objective is to propose a robust real-time fire detection method for real-world application. However, as we discussed in Section 1, to build a solid and general model for fire is hard. Instead, we aim to implement the system based on a comprehensive empirical study. Since there are little open data set available for fire detection, we firstly collected a bundle of fire videos, and then carried out comprehensive experiments on it to figure out the empirically best feature and classifier kernel for fire detection. According to our survey, neither theoretical analysis nor empirical comparisons on fire feature selection has been done before.

Color and texture are the most widely used features for general object detection and recognition. Both features can be computed efficiently, hence suitable for real-time systems. Almost all fire detection methods use color as a distinct feature of fire, but explore it in different color space. Such as methods in [3, 32, 34] use RGB color information, and method in [14] in used HSI color space. There are also other color spaces such as YUV and LAB. However, there is no trial on theoretical analysis and empirical comparison on color feature selection.

2.1 Global color feature

We experiment with four color spaces including RGB, YUV, HSV, and LAB. The global color histogram of different color space consists of 96 bins, which is the concatenation

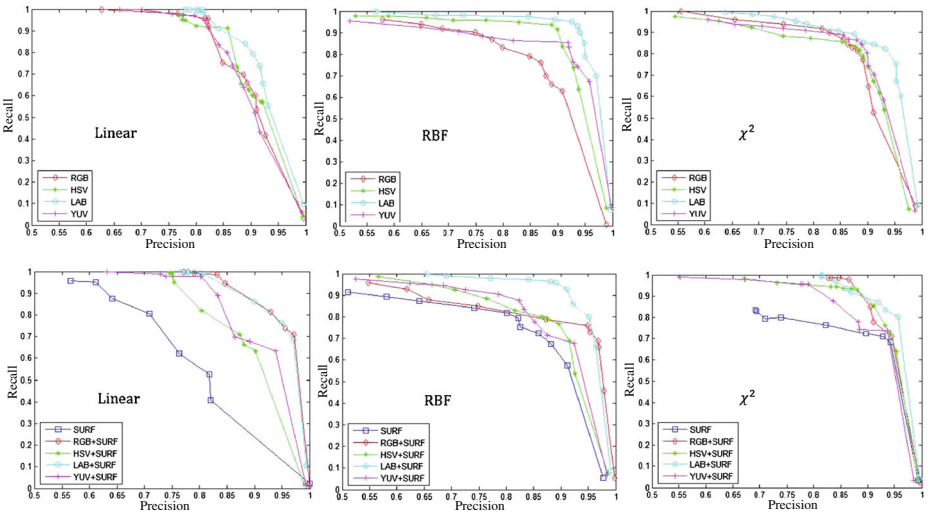


Fig. 3 Precision-recall curves of different features with 3 kinds of SVM kernels

of three 32-bin histograms for each channel. The precision-recall curves are shown on the first row in Fig. 3. As the result indicates, RGB histogram gives the poorest result, YUV, and HSV are better, and LAB performs the best in each SVM kernel setting. The highest performance is obtained by LAB histogram with RBF kernel.

2.2 Local color-texture feature

Texture is another distinctive feature besides color. Many texture descriptors like SIFT [26], and HOG [10] have been used widely and successfully on various kinds of object and scene recognition. However, the high computational complexity limits their application in video surveillance [20, 24, 25] and retrieval [11, 17]. To ensure that our system can run in real-time, we adopt the SURF descriptor [2]. It has similar performance compared to SIFT but much faster for computation. Moreover, SURF computing is parallelizable, so that it can benefit from systems with multi-core CPUs and GPUs.

Actually, both SIFT and SURF operate on gray scale images only. There are several methods to boost color saliency of SIFT descriptor [19, 21], like Hue-SIFT [35] and CSIFT [1]. The former is a concatenation of hue histogram with SIFT descriptor, and the latter computes SIFT descriptor on R, G, and B channels respectively. Other methods and their performance discussed in [30] have proved that local color information can improve the robustness and distinctiveness of SIFT descriptor by a good extent. Methods like CSIFT needs to compute 3 times over a single point and generates a descriptor with very high dimension, which slows down the matching process. In our method, we attach a local color histogram, which describes the color statistics within a SURF kernel, to a SURF descriptor.

The SURF descriptors are computed over the keypoints detected by its Fast Hessian Detector, yielding many 64 dimensional vectors. Because the quality and sharpness of video frames are relatively low, the threshold of the detector is set to 100 in order to detect more keypoints. The local color histograms are computed within the scope of each SURF kernel. Its size is 24 bin, with 8 bins for each channel. They are combined to form an 88-dimensional feature vector.

We need to obtain a fixed length vector representing the whole image, thus we use a visual vocabulary, which is also known as “codebook”, to quantize all local features sampled from the image.

To construct the codebook, we first sample, generate and collect local features from all fire and non-fire training image patches. Then we use k-means clustering algorithm

Fig. 4 The *left* image illustrates keypoint sampling and the *right* image illustrates dense sampling. *Blue dot* is the location of sampling point and the diameter of *cyan circle* represents the size of SURF kernel

Table 1 Precision comparisons of two sampling strategies

	Dense sampling (%)	Keypoint sampling (%)
SURF	**84.85**	76.43
RGB + SURF	**86.20**	79.12
HSV + SURF	84.18	**85.11**
LAB + SURF	**92.26**	84.85
YUV + SURF	**85.19**	80.47

Bold emphasis means that result is better

to produce 500 centers after 50 iterations. The centers are the elements in a codebook, since they are the most representative local features appear in our training data. The feature searching, matching and encoding steps are the same as that in our system.

The precision-recall curves of five different local features are shown on the second row in Fig. 3. As can be seen from the figure, including color information can greatly improve the system performance. Overall, LAB histogram + SURF + RBF kernel achieves the best performance.

2.3 Keypoint sampling VS. dense sampling

All previous experiments we conduct used keypoint sampling strategy in both codebook construction and classification. Recent researches like [15] and [28] indicate that dense sampling gives better results than keypoint sampling in local feature based object classification. We now sample over the whole image using SURF kernels with several predetermined intervals and scales, in both codebook construction and classification stage. The difference between keypoint sampling and dense sampling is shown in Fig. 4.

Result of comparison between both sampling strategies is shown in Table 1. We find that dense sampling outperforms keypoint sampling in most cases,which is consistent with the conclusion in [15]. In smooth surfaces like walls and ceilings, Fast Hessian detector may find few keypoints, which is an impact on classification results. In contrast, dense sampling can gather sufficient information of an image, hence it is more robust.

2.4 SURF VS. SURF-128

SURF-128 is an extended SURF descriptor that has 128 dimensions. It calculates the sum of positive and negative Laplacian separately, and the resulting signs in the descriptor distinguish positive gradient change from negative one. In our problem, it may be able to distinguish bright flame from dark background. Although SURF-128 is generally more

Table 2 Accuracy comparison of SURF and SURF-128

	64-dimension (%)	128-dimension (%)
SURF	84.85	**85.52**
RGB + SURF	**86.20**	85.52
HSV + SURF	84.18	**86.87**
LAB + SURF	92.26	**92.59**
YUV + SURF	85.19	**86.53**

Bold emphasis means that result is better

Fig. 5 Surveillance video example with still background. The background modeling result is shown in (**a**), and the mask of candidate regions is shown in (**c**)

distinctive than standard 64-dimensional SURF, it will significantly slow down the matching step. We compare SURF-128 with standard SURF to find out whether it deserves the extra cost. As shown in Table 2, on average, SURF-128 is only slightly better. In the worst case, SURF-128 is worse than standard SURF. Therefore, we abandon SURF-128 for our real-time fire detection.

3 System framework

As shown in Fig. 2, our fire detection system consists of three cascaded parts: (1) candidate fire regions proposing by a background model, (2) fire region classifying with color-texture features and a dictionary of visual words, and (3) temporal verifying. We will discuss each part in detail in the following three sections respectively.

3.1 Candidate fire region proposing

Most of the fire detection systems detect fire regions or pixels directly. Based on observations and statistic experiments, we find that fire regions are relatively bright, i.e. their pixel intensities are above some certain value. We can use this feature to find fire region candidates and fasten the further fire detection process. According to our survey, only Toreyin et al. [32] tried to eliminate the non-fire background. However, they used a fix heuristic threshold, which may fail when the background is bright or when there are fire reflections on white wall. To deal with such challenging circumstances, we adopt a multi-level threshold based on empirical study. The system can adjust the threshold automatically according

Fig. 6 Video examples with moving background and different illumination. The candidate fire region mask shown in (**b**) and (**d**) is found by a multi-level threshold according to the background intensity values

to the background intensity statistics. In our experiment, we manually divided the background intensity values into three levels, i.e. [0, 85), [85, 130), [130, 255], and set the threshold values of foreground and background intensity difference as 140, 70, 30, according to the various environment of our training data. Higher background intensity leads to lower threshold, and vice versa. In this way, the system is able to adapt itself to variations in the characteristics of the incoming data.

For a surveillance video clip, which is usually captured by a fixed camera, the background will be relatively static. We adopted the widely used Gaussian Mixture Models (GMM) [31] to further reduce the fire region candidates. Examples are shown in Figs. 5 and 6. Figure 5 illustrates a fire example taken by a surveillance camera. The first image shows the result of background modeling, and the right most white mask shows the proposed candidate fire regions. Figure 6 shows an example fire video with moving background. Since background modeling is not reliable in moving camera scenes, we use the multi-level threshold to find the fire region candidates, as shown in the binary masks.

3.2 Feature extraction and region classification

At the initialization stage of classification, we use the pre-generated codebook to build a K-D tree, which allows fast feature indexing and searching. For each incoming blob, we densely sample LAB+SURF descriptors from its region with interval of 9 pixels and SURF kernels of 9×9 scale. This step produces N descriptors.

For each descriptor d_j ,$j = 1, 2, ..., N$, we query its distance $D(d_j, v_i)$ to m nearest neighbors $v_i | i = 1, 2, ..., m$ from the K-D tree. In order to prevent drawbacks of single matching, d_j can be matched with all m nearest neighbors. Theoretically, the larger m is set, the better classification result will get, and the higher computational cost is needed. The weight of matching d_j with v_i is calculated as follows:

$$w_{ij} = \frac{K_\sigma(D(d_j, v_i))}{\sum_{k=1}^{m} K_\sigma(D(d_j, v_k))}, i \in [1, m] \tag{1}$$

Where $K_\sigma(x)$ is a Gaussian kernel function defined as:

$$K_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{1}{2}\frac{x^2}{\sigma^2}\right) \tag{2}$$

While using this matching method, we assume that the distance between d_j to v_i satisfies Gaussian distribution due to the effect of clustering, i.e. v_i is the cluster centers of the sampled features. This kernel converts smaller distance into higher probability, which is widely used in measuring similarity of two feature descriptors [15].

We use a 500-bin histogram h to encode all local features in a fire candidate blob. The values of each bin is calculated as follows:

$$h_i = \sum_{j=1}^{N} w_{ij}, i \in [1, 500] \tag{3}$$

After normalization, each bin represents the probability of occurrence of the correspondent element in the codebook and the histogram. Finally, we compute the 96-bin normalized global LAB histogram for the fire candidate blob. The concatenation of both histograms is the final representation of the blob, and is used as the input to the SVM classifier.

3.3 Temporal verification

Objects like street lamps and car lights have great similarity with fire in appearance, and they are hard to be excluded based on a single frame. Thus, blobs classified as fire in the previous step should be further verified using statistic of temporal variation (Fig. 7).

Fire mainly exhibits shape variation but not color or texture. When a new blob emerges, we use three simple parameters: perimeter, area, and spatial distribution to estimate the stableness of the region over consecutive 25 frames, without having to figure out its exact shape representation like Fourier descriptors used in [23]. Perimeter and area are very intuitive, and $\mu_p, \mu_a, \sigma_p, \sigma_a$ denote their mean and standard deviation respectively. To calculate spatial distribution, we divide the bounding rectangle of a blob into 4 equal sub rectangles, and calculate the number of pixels within each sub rectangle that are labeled as "1" in binary mask, namely d_1, d_2, d_3, d_4. The standard deviation of spatial distribution is defined as follows:

$$\sigma_d = \sigma_{d_1} + \sigma_{d_2} + \sigma_{d_3} + \sigma_{d_4} \tag{4}$$

We define that a blob is stable if the parameters satisfy the following condition:

$$\sigma_p < t_1\mu_p \quad and \quad \sigma_a < t_1\mu_a \quad and \quad \sigma_d < t_1\mu_a \tag{5}$$

And a blob is unstable if the following condition holds:

$$\sigma_p > t_2\mu_p \quad or \quad \sigma_a > t_2\mu_a \quad or \quad \sigma_d > t_2\mu_a \tag{6}$$

Where t_1 and t_2 are thresholds based on environmental settings. Indoor surveillance can use lower values. In outdoor scenarios, where fire may be greatly influenced by air flow, the thresholds are higher. False alarm rate can be further pruned by these two conditions, i.e. Satisfying equation (5) and (6) should not be proposed as fire.

Apart from color, texture and temporal characteristics of fire, we can adopt more prior knowledge to further remove the false alarm. For example, if we use blob tracking algorithm such as MeanShift [9] and find that a newly emerged blob vanishes from its original

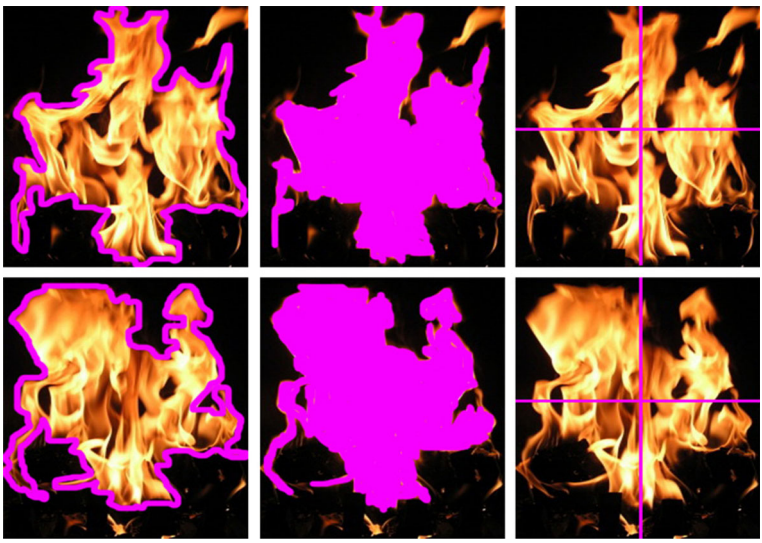


Fig. 7 The *first row* and the *second row* are two different frame of the same video respectively. From *left* to *right*: illustration of perimeter, area and spatial distribution

location after several frames, we can assert that it is not fire because fire does not move by itself. There may be other prior knowledge which we can take advantage of, but they are application specific and out of the scope of this paper.

4 Implementation details

We capture key frames from videos and manually crop out fire and non-fire patches as samples for training. We collected 726 fire patches varied in colors, shapes, and intensities, and 637 non-fire samples involving a variety of objects including human and cars, which are full of visual details, and plain objects such as ceilings, walls, and skies, which are simple in color and texture. We randomly select 4/5 of the patches for training and 1/5 for testing.

We exploit Support Vector Machine (SVM) to do fire classification, since SVM is one of the latest and proved the most successful statistical pattern classifier. The LIBSVM [6] is used in our system. The weights of positive and negative sample are balanced with respect to their number. We search for optimized parameters using 5-fold cross validation with a parameter range of 2^{-8} to 2^{8}. We test the system with three different SVM kernels: linear, RBF, and χ^2 [30].

All the parameters are fine-tuned and fixed according to the experiment results. Theoretically, the larger the value of m, i.e., the number of the nearest neighbors to be matched in the codebook, the more accurate is the resulting matching. We set m=10 to achieve the balance between computational cost and performance. The temporal verification thresholds t1, t2 in Section 3.3 are set to be 0.02, 0.6 indoors, and 0.04, 0.8 outdoors according to the training data, which shows that fire may be greatly influenced by air flow in outdoor scenarios. Note that if t1 is significantly low, more stable candidates like the bulbs are recognized as fire, with the increase of false alarm rate. Likewise, if t2 is significantly high, more fluctuating ones like the blinking or flickering light might also be accepted.

5 Results and discussion

5.1 Data set collection

There are some small data set for fire detection available on internet [36], such as the data set published by Cetin et al. includes 13 video clips [5]. But these data sets are usually small and neither of them covers most of the real world scenarios. To test the performance of our fire detection system, we introduce *FireCollect*, a larger fire and flame detection data set. We downloaded the small public data sets and randomly downloaded fire video clips from public video sharing web sites, such as youtube.com and youku.com. We also captured some fire videos by ourselves to expand the data set. Finally we have 64 video clips in total for the proposed data set. The average length of each clip is 2 minutes. We tried to cover most of the real world application scenarios including fire indoor (20 clips), urban outdoor (21 clips), and forest fire(9 clips) with both static and moving background. Potential false alarms cases (14 clips) in the data set include car lights, human wearing fire-color clothes, illumination changing, and so on. Sample image frames from the *FireCollect* data set are showed in Fig. 8.

We use our newly collected data set to benchmark the performance of our method, and we also test Toreyin et al.'s method [32], which is one of the state-of-the-art methods using in video based fire detection. Since both methods make a decision at an interval of several

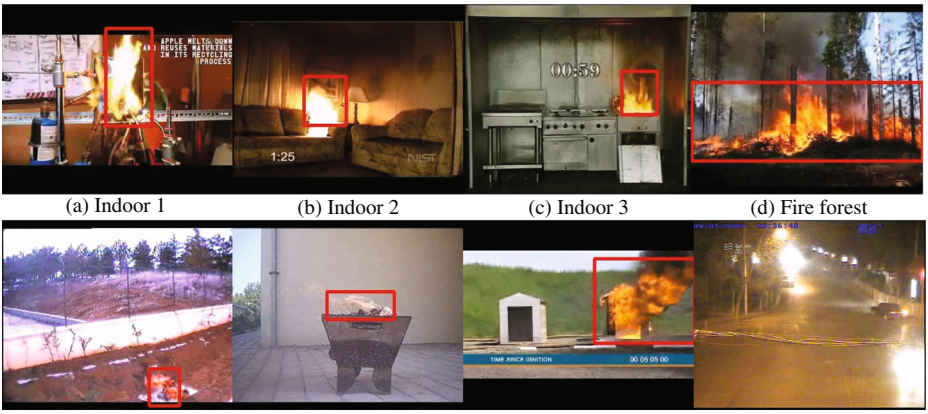
(a) Indoor (b) Outdoor (c) Forest fire (d) No fire

Fig. 8 Sample image frames of different scenarios from the collected data set

frames, we do not use a frame-by-frame evaluation. We manually split the videos into individual sections of 200 frames, and label them as either "contains fire" or "does not contain fire". As a result, we have 774 sections as testing units. We adopt the Precision and Recall rate for measurement. To demonstrate the efficiency of our framework, the average processing times as well as time delay of issuing a correct alarm are reported. In Fig. 9, there are sample detection results of our method in different scenarios. It can be seen that our method is robust in both indoor and outdoor scenes, while being insensitive to fire-like objects such as moving car light. The comparison result is shown in Table 3.

Overall, our method has similar precision rate comparing to Toreyin's method. However, we significantly outperform Toreyin's method in recall rate. We should be aware that miss in fire detection is much more critical than false alarm. Therefore, our method is more preferable for real-life applications [22]. Besides, our framework is computationally efficient compared to the state-of-the-art, with an average processing time of 25 fps when image frames of size 352×288 are used, which is real-time and can be used in a real world scenario.

We also highlight the comparison results of our method and Toreyin's method for challenging instances picked out from the data set, as illustrated in Table 4. The typical scenarios are described briefly in the right column as video content description. We can see our result is much better than Toreyin's method. However, when inspecting the result, we find some typical failure cases. Both our method and Toreyin's fail to detect the fire on initial combustion, because the size of the flame is very small. Nevertheless, our method is still more sensitive to fire with regarded to response time (see Table 3). Both methods issue some false



(a) Indoor 1 (b) Indoor 2 (c) Indoor 3 (d) Fire forest

(e) Outdoor 1 (f) Outdoor 2 (g) Outdoor 3 (h) No fire

Fig. 9 Detection results from the test video clips

Table 3 Overall benchmark performance

	Our method	Toreyin et al. [32]
True positive	361	311
True negative	305	308
False positive	27	24
False negative	81	131
Precision rate	**93.04 %**	92.83 %
Recall rate	**81.67 %**	70.36 %
Avg. processing time	**25 fps**	20 fps
Avg. response time	**within 2s**	within 4s

Bold emphasis means that result is better

alarms on flash lamp, since the lamp is in bright yellow tint and flicking in video, which is similar to the characteristics of fire (see Fig. 10b).

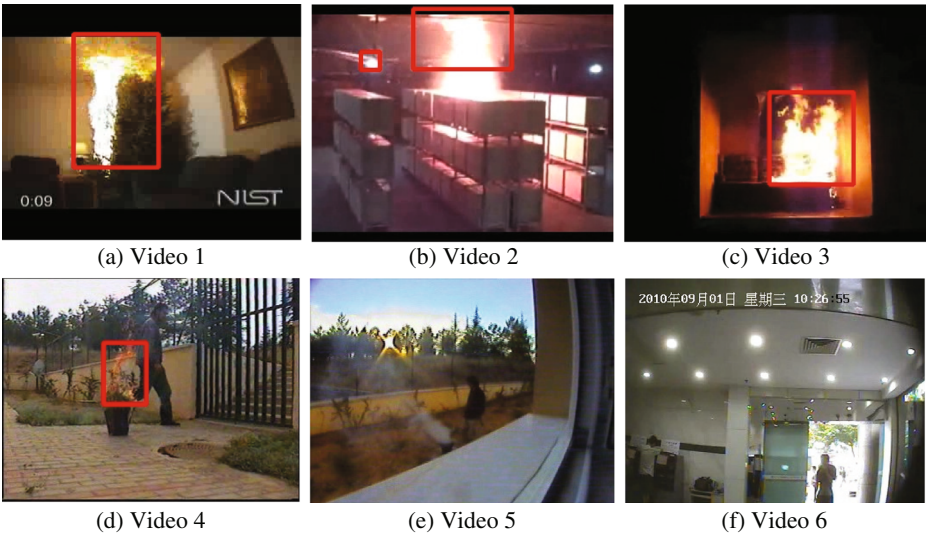
Due to the naïve thresholding in pixel color, Toreyin's method always fail to recognize fire with bright intensity, which is the main reason of its low recall rate. Our method, however, overcomes this problem and gains great advantage. Toreyin's method also issues false alarms on human skin and yellow shirts. In contrast, our method withstands the challenge.

To demonstrate the superior performance of the proposed method on other datasets, we further test our algorithm on the widely used dataset published by Cetin et al. [5], and compare with two competitors which we called *Method 1* and *Method 2*, respectively. *Method 1* [12] newly proposed by Cetin et. al. also utilizes color and temporal information for detection, without using texture features. *Method 2* [33] adopts wavelet transformation to capture the motion information of fire. The dataset we test contains 13 fire videos with a total number of 8093 frames. As the last two clips are not publicly available, we test with the first 11 clips and report the true detection rates in Table 5. For fair comparisons, the calculation of metrics follows the standard process introduce in [12].

It is clearly shown that our algorithm outperforms [12] in 9 scenarios out of 11 clips, with a highest detection rate of 92.6 %, compared to 89.8 % in [12] and 51.7 % in [33].

Table 4 Detection results for typical instances from the data set

	Our method		Toreyin et al. [32]		
Input	False positive	False negative	False positive	False negative	Video Content Description
Video 1	0	0	0	6	A Christmas tree is burning in a bright room
Video 2	4	0	9	1	Flash lamp and dazzling flame in a warehouse
Video 3	0	0	0	9	A man doing fire experiment in a dark place
Video 4	0	0	0	5	White flame burning in a back yard
Video 5	0	0	1	0	A man is walking around
Video 6	0	0	4	0	Bank surveillance video



(a) Video 1 (b) Video 2 (c) Video 3

(d) Video 4 (e) Video 5 (f) Video 6

Fig. 10 Sample results of the typical instances corresponding to Table 4. Note (**b**) shows a false alarm due to the tint and flicker of the lamp

Table 5 Comparison of our method with two state-of-the-arts [12, 33] in terms of true detection rates in dataset [5]

	True detection rates			
Video name	Our method	Method1 [12]	Method2 [33]	Description
Video 1	266/293 (90.8 %)	161/293 (54.9 %)	0/293 (0.0 %)	A truck is burning
Video 2	487/510 (95.5 %)	413/510 (81.0 %)	0/510 (0.0 %)	A woman lit a fire in the kitchen
Video 3	350/381 (91.9 %)	310/381 (81.4 %)	0/381 (0.0 %)	Branches are on fire
Video 4	1643/1655 (99.3 %)	1643/1655 (99.3 %)	627/1655 (37.9 %)	Flame in a dark place 1
Video 5	2394/2406 (99.5 %)	2394/2406 (99.5 %)	2348/2406 (97.6 %)	Flame in a dark place 2
Video 6	246/258 (95.3 %)	35/258 (13.6 %)	0/258 (0.0 %)	A man lit a fire
Video 7	535/547 (97.8 %)	495/547 (90.5 %)	404/547 (73.9 %)	Waste paper catches fire outside
Video 8	255/513 (49.7 %)	501/513 (97.7 %)	0/513 (0.0 %)	Flame in a dark place 3
Video 9	634/663 (95.6 %)	651/663 (98.2 %)	64/663 (9.7 %)	A fire on a pot
Video 10	223/235 (94.9 %)	223/235 (94.9 %)	181/235 (77.0 %)	Forest fire 1
Video 11	43/178 (24.7 %)	35/178 (19.7 %)	23/178 (12.9 %)	Forest fire 2
Total	**92.6 %**	89.8 %	51.7 %	

Note some of the videos are recorded with hand-held moving cameras. Since [33] assumes a stationary camera, it fails to detect most of the positive fire regions in the clips.

5.2 Efficiency

The proposed framework is computationally efficient. Our implementation is coded in C++ on an Intel I7 3.4 GHz processor with 4GB memory. On average, video clips are processed 25 fps when image frames of size 352 × 288 are used. It is sufficiently fast for real-time applications due to the usage of simple features, the key point sampling strategy and the fast indexing and searching tree structure. The method issues a correct alarm within 2 second on the average.

6 Summary

In this paper, we present a robust real-time fire detection method based on empirical study. To carry out the empirical experiments, we collect so far the largest open data set for fire detection in video. Experiments show that, overall, LAB histogram plus SURF texture descriptor, plus the RBF SVM kernel, lead to the best performance of an 82 % recall with 93 % precision on the proposed larger data set, as well as the highest true detection rate of 92.6 % on a popular small data set, which outperform the performances by state-of-the-arts methods.

For future works, we can study more detailed fire characteristics to effectively detect the fire on initial combustion and exclude the fire-like objects such as car light in night video.

References

1. Abdel-Hakim A, Farag A (2006) Csift: A sift descriptor with color invariant characteristics. IEEE computer society conference on computer vision and pattern recognition, 2006, pp 1978–1983
2. Bay H, Ess A, Tuytelaars T, Gool LV (2008) Surf: Speeded up robust features. Comput Vision Image Underst (CVIU) 110(3):346–359
3. Borges PVK, Izquierdo E (2010) A probabilistic aprroach for vision-based fire detection in videos. IEEE Trans Circ Syst Vi Technol 20(5):721–731
4. Celik T, Demirel H, Ozkaramanli H, Uyguroglu M (2007) Fire detection using statistical color model in video sequences. J Visual Commun Image Represent 18(2):176–185
5. Cetin AE (2007) Computer vision based fire detection software. http://signal.ee.bilkent.edu.tr/VisiFire
6. Chang CC, Lin CJ (2001) LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/cjlin/libsvm
7. Chen TH, Wu PH, Chiou YC (2004) An early fire-detection method based on image processing. IEEE international conference on image processing (ICIP'04), pp 1707–1710
8. Cho BH, Bae JW, Jung SH (2008) Image processing-based fire detection system using statistic color model. International conference on advanced language processing and Web information technology (ALPIT'08), pp 65–76
9. Comaniciu D, Ramesh V, Meer P (2000) Real-time tracking of non-rigid objects using mean shift. IEEE computer society conference on computer vision and pattern recognition (CVPR'00), pp 2142–2142
10. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. IEEE computer society conference on Computer Vision and Pattern Recognition (CVPR, 2005), vol 1, pp 886–893
11. Duan X, Lin L, Chao H (2013) Discovering video shot categories by unsupervised stochastic graph partition. IEEE Trans Multimed 15(1):167–180
12. Habiboglu YH, Gnay O, Cetin AE (2012) Covariance matrix-based fire and flame detection method in video. Mach Vis Appl 1103–1113

13. Healey G, Slater D, Lin T, Drda B, Goedeke A (1993) A system for real-time fire detection. IEEE computer society conference on computer vision and pattern recognition (CVPR93), pp 605–606
14. Horng W, Peng J, Chen C (2005) A new image-based real-time flame detection method using color analysis. IEEE proceedings on networking, sensing and control, pp 100–105
15. Jurie F, Triggs B (2005) Creating efficient codebooks for visual recognition. Computer Vision, 2005. ICCV 2005. The 10th IEEE international conference on ICCV 2005, vol 1, pp 604–610
16. Ko BC, Cheong KH, Nam JY (2009) Fire detection based on vision sensor and support vector machines. Fire Saf J 44(3):322–329
17. Lai H, Pan Y, Liu C, Lin L, Wu J (2013) Sparse learning-to-rank via an efficient primal-dual algorithm. IEEE Trans Comput 62(6):1221–1233
18. Lee B, Han D (2007) Real-time fire detection using camera sequence image in tunnel environment. In: Proceedings of the intelligent computing 3rd international conference on Advanced intelligent computing theories and applications, ICIC'07, pp 1209–1220
19. Lin L, Liu X, Zhu SC (2010) Layered graph matching with composite cluster sampling. IEEE Trans Pattern Anal Mach Intell 32(8):1426–1442
20. Lin L, Lu Y, Pan Y, Chen X (2012) Integrating graph partitioning and matching for trajectory analysis in video surveillance. IEEE Trans Image Process 21(12):4844–4857
21. Lin L, Luo P, Chen X, Zeng K (2012) Representing and recognizing objects with massive local image patches. Pattern Recog 45(1):231–240
22. Lin L, Wang Y, Liu Y, Xiong C, Zeng K (2009) Marker-less registration based on template tracking for augmented reality. Multimedia Tools Appl 41(2):235–252
23. Liu CB, Ahuja N (2004) Vision based fire detection. Pattern Recognition, 2004. Proceedings of the 17th international conference on ICPR 2004, vol 4, pp 134–137
24. Liu X, Lin L, Jin H (2013) Contextualized trajectory parsing with spatio-temporal graph. IEEE Trans Pattern Anal Mach Intell 35(12):3010–3024
25. Liu X, Lin L, Jin H, Yan S, Tao W (2011) Integrating spatio-temporal context with multiview representation for object recognition in visual surveillance. IEEE Trans Circ Syst Vi Technol 21(4): 393–407
26. Lowe D (1999) Object recognition from local scale-invariant features. Computer Vision, 1999. Proceedings of the 17th IEEE international conference, vol 2, pp 1150–1157
27. Luo R, Su K (2007) Autonomous fire-detection system using adaptive sensory fusion for intelligent security robot. IEEE/ASME Trans Mechatron 12(3):274–281
28. Nowak E, Jurie F, Triggs B (2006) Sampling strategies for bag-of-features image classification. European Conference on Computer Vision
29. Podraj P, Hashimoto H (2008) Intelligent space as a framework for fire detection and evacuation. Fire Technol 44:65–76
30. van de Sande K, Gevers T, Snoek C (2010) Evaluating color descriptors for object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp 1582–1596
31. Stauffer C, Grimson WEL (1999) Adaptive background mixture models for real-time tracking. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99), pp 2246–2252
32. Toreyin BU, Cetin AE (2007) Online detection of fire in video. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR07), pp 1–5
33. Toreyin BU, Dedeoglu Y, Gdkbay U, Cetin AE (2006) Computer vision based method for real-time fire and flame detection. Pattern Recog Lett 49–58
34. Walter PI, Shah M, Lobo NV (2002) Flame recognition in video. Pattern Recog Lett 23(1–3):319–327
35. van de Weijer J, Gevers T, Bagdanov A (2006) Boosting color saliency in image feature detection. Pattern Anal Mach Intell IEEE Trans 150–156
36. Yao B, Yang X, Lin L, Lee M, Zhu SC (2010) I2t: Image parsing to text description. Proc IEEE 98(8):1485–1508

Bo Jiang received his B.S. and M.S. degrees in Guangxi University, Nanning, China, in 2003 and 2006, respectively. He is the research engineer at the Experimental Teaching Center of Sun Yat-sen University. His research interests include machine learning, intelligent mobile systems, and embedding software.

Yongyi Lu received his B.S. and M.S. degrees in Sun Yat-Sen University. His research interests include machine learning and video surveillance.

Xiying Li received her Ph.D. dgree from Beijing Institute of Technology (BIT), Beijing. She is currently an associate professor with the Engineering School of Sun Yat-Sen University (SYSU), China. She has published a number of academic papers in international conferences and journals.

Liang Lin received the B.S. and Ph.D. degrees from Beijing Institute of Technology (BIT), Beijing, China, in 1999 and 2008, respectively. From 2006 to 2007, he was a joint Ph.D. student with the Department of Statistics, University of California, Los Angeles (UCLA). He was a Post-Doctoral Research Fellow with the Center for Vision, Cognition, Learning, and Art of UCLA. He is currently an associate professor with the Software School of Sun Yat-Sen University (SYSU), China. He was awarded by the "Hundred Talents Program" of SYSU in 2009, the "Program for New Century Excellent Talents" of Ministry of Education (China) in 2012, and the Guangdong Natural Science Funds for Distinguished Young Scholars in 2013. He received several academic honors, including China National Excellent PhD Thesis Award Nomination in 2010, Best Paper Runners-Up Award in ACM NPAR 2010, and Google Faculty Award in 2012. He fulfills review duties for more than 10 journals and various conferences including TIP, TCSVT, IJCV, NeuroComputing, PR, CVPR, ICCV, and ICPR. He is an Assistant Editor of Journal of Pattern Recognition Research (JPRR) from 2013.