Real-Time Fire Detection in Camera Stream Using Statistical Analysis

KARE KOPLÍK, PETER JANKŮ, OLGA VOZNIUK, TOMÁŠ DULÍK, PETR SNOPEK
Faculty of applied informatics
Tomas Bata University in Zlin
Nad Stranemi 4511, 760 05 Zlin,
CZECH REPUBLIC
janku@fai.utb.cz http://www.utb.cz/fai

Abstract: - The paper describes a new algorithm designed to be fast and efficient for detecting fire. It is based on finding and investigating suspicious regions in each frame of video stream. The investigation consists of tracking regions across frames and performing statistical analysis on their trajectory.
If the trajectory has characteristic similar to fire, a test on persistence is performed. If the fire-like characteristics persists, the alarm is triggered. This criterion enables to eliminate a large proportion of false alarms. Given it's simplicity, the algorithm can be used separately in some environments and can improve existing algorithms as well.

Key-Words: - Fire detection. Statistical analysis. Digital signal processing

1 Introduction

In this paper, we present new, innovative and simple algorithm for detecting fire in video stream eg. from a camera. It can be used as a standalone algorithm and it can also improve more complex algorithms by adding more certainty to final decision.

Its main advantage is it performs well without demanding too much computational power and when added to existing algorithm, it can work with data already produced in initial steps.

Most algorithms detect suspicious regions and then investigate them for farther fire-like characteristics. Our algorithm is based on tracking these regions in time and statistically analysing their trajectory.
Our goal was to develop a fast algorithm which would run on a relatively affordable hardware with minimum false alarm ratio.

Similarly to other researchers before us, we attempt to follow the idea that systems solely based on processing signals from sensors are outdated and need to be replaced with much more effective and reliable computer-vision based solution.

Sensors have their limitations. They cannot work outside. They need to be relatively close to fire and there has to be multiple of them to cover all potentially dangerous places.

Cameras can monitor much larger areas and the respective systems can make final decision with more certainty. It is therefore desirable to build a system which can redefine the existing standard.

2 Detailed description

The entire algorithm is very simple. It consists of the following steps:

1. Detect suspicious regions
2. Find bounding rectangles
3. Track rectangles in time
4. Analyse trajectories
5. Check for persistence

In the following sections we will look at each step. To demonstrate, let's use a video of a burning tree as an example.

2.1 Detecting suspicious regions

The regions of interest in this case would be the pixels which have fire-like colour (P_C) and also pixels which change visibly in time (i.e. contain movement; P_M). Suspicious pixels (P_S) are then an intersection of these two sets:

(1) $$P_S = P_C \cap P_M$$

2.1.1 Detecting pixels with fire-like colour

Finding pixels based on colour is very simple and it can be done using different colour models like RGB, YUV, YCbCr HSI or HSV. [1][6][8] The first two focus on colour spectrum and the latter two on colour intensity. Fire is usually the brightest part in the video so using brightness as an additional criterion is very useful.

In our algorithm we work with RGB colour model and the procedure looks like this: We take red (R), green (G), and blue (B) channels and calculate colour saturation (S). The rules describe by equation 2 are applied. If the result of this rules is true than the pixel belongs to the P_c set.

$$R > G > B \\ S > S_r \tag{2}$$

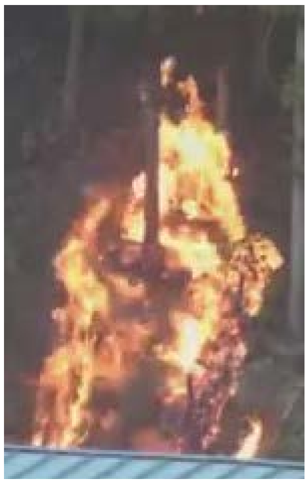
Fig. 1 Burning tree

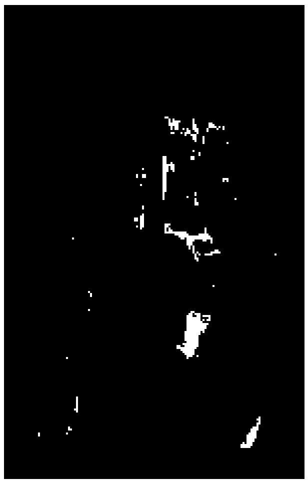
Fig. 2 Fire colour map

2.1.2 Detecting movement

We need three frames to detect movement. Because we need the previous, current and next frame, the algorithm is always one frame behind a real-time stream (e.g. a camera) input.

We need to convert all frames to greyscale so we can then simply subtract the background to detect any motion in the foreground.

First we calculate an absolute difference between previous (f_{i-1}) and next (f_{i+1}) frame. This will subtract the background. Then we calculate an absolute difference between current (f_i) and the next frame. This will update that information. Then we apply binary AND operator on the two calculated differences to obtain information about the movement in the foreground.

$$|f_{i-1} - f_{i+1}| \wedge |f_i - f_{i+1}| \tag{3}$$

It is also necessary to filter out lone pixels (i.e. noise) in the resulting movement map. The amount of noise depends on the camera and lighting conditions.

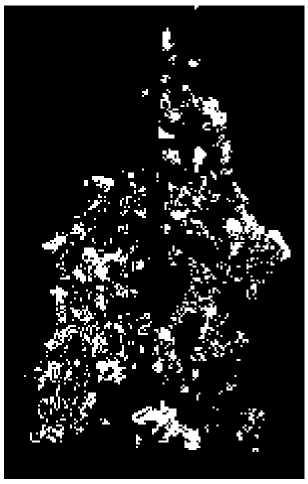
Fig. 3 Movement map

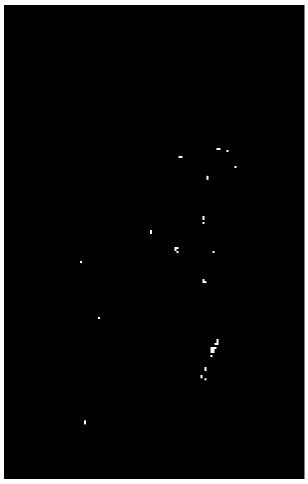
Fig. 4 Suspicious areas

2.2 Finding bounding rectangles

The binary map of suspicious pixels will most likely consist of lots of small disconnected areas concentrating in separated larger regions. At first, we tried to find bounding rectangles for each of them. This turned out to be very costly in terms of processing time. The solution was to dilate the small areas using a morphology operator.

Dilatation (as a morphological operation) consists of convoluting an image with a small kernel shaped like a simple shape (circle, square, etc.).

This results into much fewer bounding rectangles to be found and aggregated in the next step and significantly increases the algorithm speed.

2.3 Tracking suspicious regions in time

Similarly to [7] we test suspicious regions' fire-like characteristic in time. Our method consists of tracking. Before we start we need to aggregate found areas in space and then link them in time.

As it turned out, this was initially easier said than done. If we simply find bounding rectangles for small disconnected areas (as seen in Fig. 4) we get lots of small bounding rectangles inside the actual fire region.

We also cannot use the fire-colour map or the movement map (Fig. 2 and 3) (despite these being more dense) because they cover regions of both fire and fire-like background in a way that they very often merge into one inseparable mixed region.

Another problem is, the map of suspicious pixels tends to change very rapidly between two frames. If we simply connect small areas which are close to each other (applying some tolerance constant), the result will be very unstable in time. The bounding rectangles would keep changing to a multiple (or small fraction) of their previous size. It would also split into several small rectangles then merge into fewer large ones. If we set the tolerance constant higher to make sure we get one rectangle covering the whole fire region we increase the error in more complex videos with more than one fire region.

The solution to all these problems was to stop connecting suspicious areas in each frame (i.e. in space) individually and then look for overlaps across frames (i.e. in time) and start connecting them both in space and time in one slightly more sophisticated process.

Our algorithm involves a tree data structure consisting of two layers of bounding rectangles. We describe it in more detail in the following subsections.

2.3.1 Aggregating regions in space

In the first frame we add rectangles to the root of the tree. If a rectangle to be added overlaps (with added tolerance) any first-layer rectangle, they are merged into a new rectangle which is the smallest rectangle containing all of them and they are added as its children (i.e. they are added to the second layer of the tree).

If they already have children, their children are moved to the new parent and the original parent is discarded (i.e. the third-layer rectangles are always pushed to the second layer, replacing the second-layer parent). Example tree structure can be seen at Fig. 5.

Once all the rectangles have been added to the tree, we check each first-layer rectangle if it has any children.

If it has, we find the smallest bounding rectangle for all children and resize the parent accordingly. Then we remove the children from the second layer.

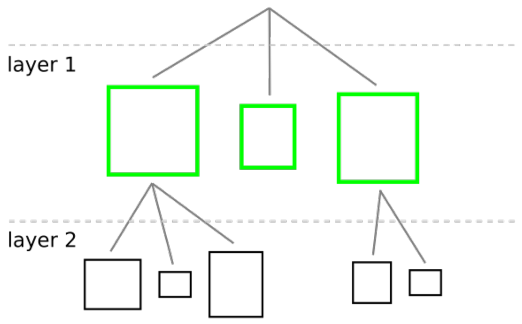

Fig. 5 Example tree structure

For better understanding, please look at Fig. 6. The child rectangles are drawn with thin, full, black line. If they overlap (with added tolerance depicted with thin, dashed, gray line), they are added to the same parent. The parent's rectangle is then the smallest bounding rectangle (thick, full, green line).

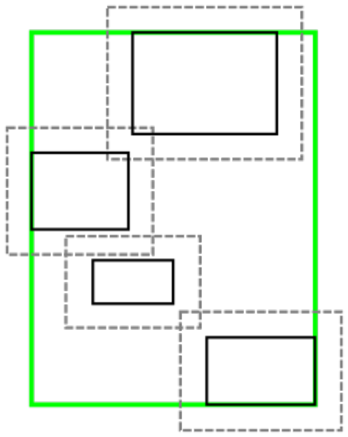

Fig. 6 First frame parent rectangle

2.3.2 Linking regions in time

In all of the following frames we continue same as with the first frame only now we have preexisting first-layer and from now on we will resize the parent to the average of its final size in the previous frame and the smallest bounding rectangle calculated for the current frame.

It will look like as shown at Fig. 7. Despite the small rectangles not overlapping, we know they belong to the same fire region because they are inside the bounding rectangle from the previous frame (drawn with thick, dashed, light green line). The minimal bounding rectangle (thick, full, light green line) and the previous size will give us the new parent size (thick, full, green line) by calculating their average.

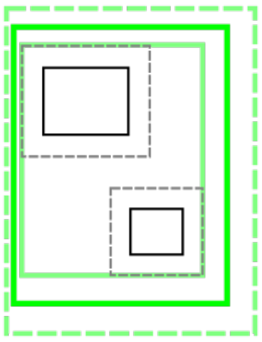
Fig. 7 Resizing parent rectangle

If the parent ends up with no children, we reduce its RTL (right to live) parameter. If it has at least one child, we will set RTL to predefined maximum value.

The RTL parameter basically tells us how long will we keep a rectangle which has not been updated (e.g. for 3 frames).

2.3.3 Tracking regions in time

The tracking is based on keeping a history of middle points. These are not the centre points of bounding rectangles. Instead, we take the original map of suspicious regions (without applied dilatation) and average the coordinates of suspicious pixels. The new new average point is then added to a stack.

The reason we use the stack structure is that we remove points if their count surpasses defined limit (the points are no longer needed for analysis).

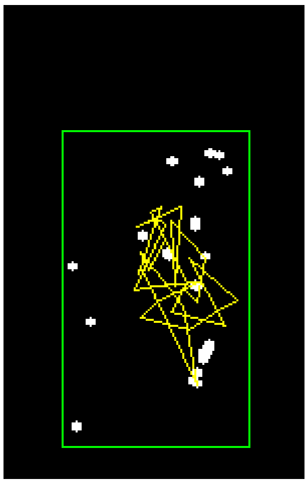
Fig. 8 Suspicious region's trajectory

2.4 Analysing trajectories

This and the following step are very important. Without them, we would be detecting a large spectrum of objects and features which are similar to fire in colour and in movement characteristics.

The fire has a very specific way of moving. First of all, it stays in one place and is distinct by constant flickering. This is very useful because it's hard to find anything else what is (yellow and bright and) constantly changing shape while staying in one location. The trajectory of the middle points should basically fit the normal distribution.

To test this with our suspicious regions, we use horizontal and vertical coordinates of the middle points. First we calculate the mean value μ and the standard deviation σ for each axis. If the distribution is normal, according to the gaussian curve (Fig. 9), 68.2% of values should belong to the interval:

$$(\mu - \sigma, \mu + \sigma) \tag{4}$$

Since we work with object that doesn't only change shape but also size over time, it is not wise to rely on too many values to increase precision. Instead, we work with fewer values (according to our tests, the optimal number seems to be around 100) and expect some reasonable error. That is why we had settled to expecting 60% of data to fall into the above specified interval.

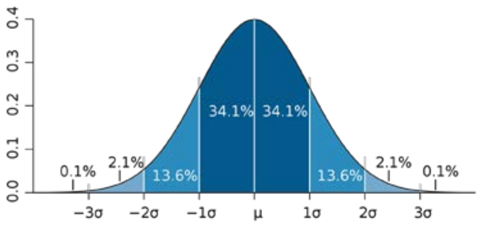

Fig. 9 Gaussian curve

2.4 Checking for persistance

To further eliminate false alarms, we apply one more criterion: persistence. If the previous step gives us positive result, we don't trigger the alarm yet but save it. This we do for each frame.

According video source we can then set the persistence limit to be at least 15 seconds (which would be 375 frames for 25 fps video source) or even more. It is not recommended to set this limit to be less than that. On the other hand, higher limit will result in longer delay. (It will also increase the relatively small chance the delay will surpass the limit in case the criterion fails, i.e. false negative.)

To evaluate persistence, we don't require the positive test on normal distribution in each frame but it has to be in defined higher percentage of frames. Empirically, we had set it to be the top quarter.

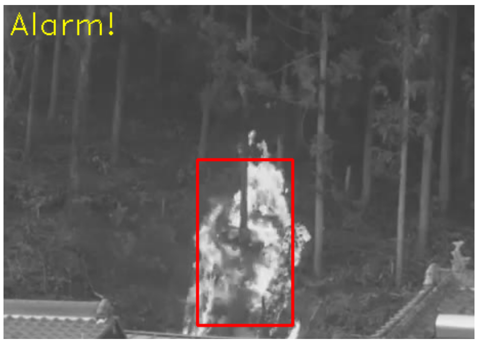

Fig. 10 Output screen

3 Algorithm parameters
There are several parameters which need to be set after a camera providing video stream for our algorithm is installed into a new environment.

3.1 Brightness threshold
Defines the minimum brightness needed for classifying pixels as fire-coloured. Normally the best value would be around 225 (out of 255) but in some test videos it had to be adjusted because the fire was too dark due to the type of video source.

3.2 Movement threshold
During the movement detection processing the camera noise is reduced by filtering out some lone pixels. This parameter sets the threshold. Usually in badly lit areas the threshold needs to be set higher. This is important because noise has normal distribution so it could cause a false alarm.

3.3 Small regions gap size
Maximum distance between two small regions' bounding rectangles which can be aggregated. This parameter should have a small value (e.g. 10 px) and depends on video resolution.

3.4 Regions minimal size
If there are too many very small regions found because of camera noise, we eliminated them by setting a minimal bounding rectangle size. If the size is set too high, small flames will not be detected. If it's too low, the noise can trigger false alarm.

3.5 Region's right to live
As was explained in the previous chapter, this parameter sets how long will we keep a region appeared to be empty in a few frames. This mostly happens when there is no movement detected. Since the fire has multiple independently moving flames it can be expected with high probability it will not stop moving for more than few (e.g. 3) frames.

3.6 Number of middle points
The parameter defines how many middle points we want to remember for each region before we test them for normal distribution. It should be enough for this criterion to be calculated relatively accurately but also not too many so we adapt to changes in the scene quickly. We use empiric value 100.

3.7 Persistency interval
Number of frames for which the positive detection has to persist before triggering the alarm. This parameter needs to be changed according to number of frames per second in each video stream.

The higher this number is the harder it is to fool the algorithm but also the longer delay we get. Fire keeps its characteristic movement all the time, but most other object and phenomena moves chaotically and in short intervals. In our tests, 15 seconds turned out to be sufficient minimal length.

4 Practical tests
We had acquired around 60 videos recorded with a static camera. One third were videos with fire, next third were videos containing random movement (e.g. people, cars, birds) and the last were videos which we thought could prove problematic given the way the algorithm works. All the videos were chosen with the potential algorithm installation environment in mind.

First two categories run without a problem. In case of fire-less videos, the final criterion, the persistency, was 0 at all times. The fire-containing videos had persistency mostly above the alarm-triggering threshold (75%) and were only delayed by the minimal trajectory length requirement (15s) (see Fig. 11).

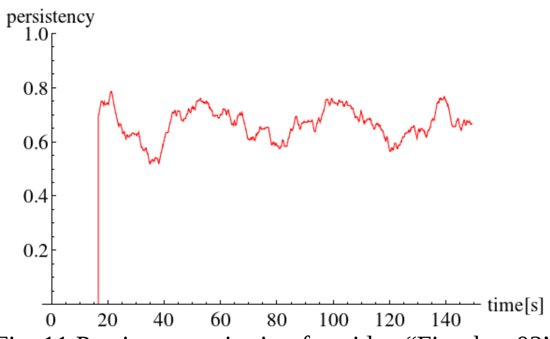

Fig. 11 Persistency criterion for video "Fireplace02"

The third category was interesting. There were three major problems. One was, if the video contained lots of moving objects (e.g. cars on a busy road at night, brightly dressed dancers on a stage), the normal-distribution criterion would pass very often. In this case, the alarm would not be triggered, because of the persistency criterion, which requires the movement to be almost constant (see Fig. 12).

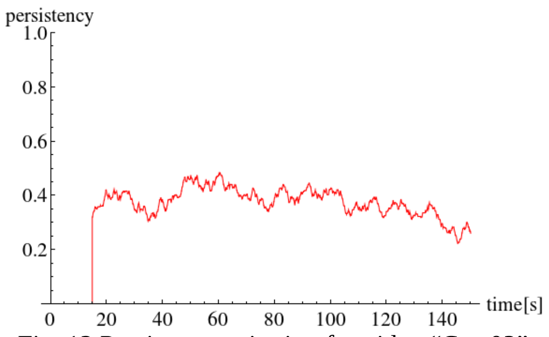

Fig. 12 Persistency criterion for video “Cars03”

Second problem were random movements, which passed the persistency criterion from time to time, if the footage was long enough (see Fig. 13). Of course, this problem can be solved by making the persistency interval longer.

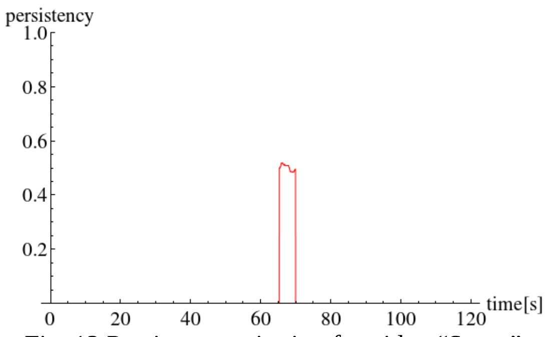

Fig. 12 Persistency criterion for video “Street”

The worst problem is the third one. It happens when all the criteria are passed even when there is no fire. It didn’t occur very often and in some cases, it was possible to eliminate it by reducing noise threshold in the movement criterion or increasing brightness threshold in the colour criterion. However, there were situations, where the algorithm simply failed. One example is a video of sunset over ocean surface. This natural phenomena just looked very similar to fire.

5 Performance

To test our algorithm, we had acquired several videos containing flames, moving bright objects and phenomena similar to fire.

5.1 Algorithm speed

Our algorithm is very simple so when optimized, it could run in real time on average hardware. It doesn’t rely on frequency of fire flickering unlike some algorithms so the frame rate doesn’t have to be that high either.

If we ask how fast can our algorithm detect flame it really depends on the camera setting and scene characteristics but in ideal conditions it depends solely on the defined length of the minimal persistence interval.

Not ideal conditions would be e.g. the flame is too small or too far from the camera, the flame is not fully visible in camera’s field of view or the flame movement is being distorted by another moving object (in one test video it was a burning piece of wire rolling out of flames).

5.1 Strengths and weaknesses

Similarly to other fire-detecting algorithms the most concerning weakness is false alarm ratio and avoiding it relies on combining multiple criteria [2][3][4][5]. If all criteria are fooled, the false alarm occurs.

What was very problematic was white background in some videos. Anything what moves in front of such background can pass the movement-and-brightness criterion because even when the object is not bright the algorithm sees only bright areas changing shape as the object covers and uncovers the background.

Other concern we have is with natural phenomena similar to fire. When testing the algorithm on videos of sunset being reflected on moderately moving ocean surface, the false alarm was triggered every time.

Other than that, our algorithm proved to work without major problems and was almost impossible to fool intentionally when processing real time camera input.

6 Conclusion

We had developed and successfully tested a new algorithm. In this stage in can be used with any camera system and will work mostly without error. However, there are certain situations in which the algorithm triggers false alarm. This will be the subject of our future research.

7 Acknowledgement

This paper is supported by Technology Agency of the Czech Republic (TA ČR) within the Visual Computing Competence Center - V3C project No. TE01020415, by Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme project No. LO1303 (MSMT-7778/2014) and also by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

References:

[1] T. Chen, P. Wu, and Y. Chiou, “An Early Fire-Detection Method Based on Image Processing,” Proc. IEEE Int. Image Process., 2004, pp. 1707-1710.

[2] B.U. Toreyin, Y. Dedeoglu, and A.E. Cetin, “Flame Detection in Video Using Hidden Markov Models,” Proc. IEEE Int. Conf. Image Process., 2005, pp. 1230-1233, 2005.

[3] W. Krüll et al., “Design and Test Methods for a Video-Based Cargo Fire Verification System for Commercial Aircraft,” Fire Safety J., vol. 41, no. 4, 2006, pp. 290-300.

[4] Turgay Celik, "Fast and Efficient Method for Fire Detection Using Image Processing," ETRI Journal, vol. 32, no. 6, Dec. 2010, pp. 881-890.

[5] Liu, Zhigang, George Hadjisophocleous, Guofeng Ding and Choon Siong Lim. Study of a Video Image Fire Detection System for Protection of Large Industrial Applications and Atria [online]. [cit. 2013-12-01].

[6] Poobalan, Kumarguru and Liew, Siau-Chuin. Fire Detection Algorithm using Image Processing Techniques. In: E-Proceeding of the 3rd International Conference on Artificial Intelligence and Computer Science (AICS2015), 12-13 October 2015 , BayView Hotel, Penang, Malaysia. pp. 160-168.. ISBN 978-967-0792-06-4.

[7] Jiang, B., Lu, Y., Li, X. et al. Multimed Tools Appl (2015) 74: 689. doi:10.1007/s11042-014-2106-z.

[8] A. E. Gunawaardena, R. M. M. Ruwanthika, A. G. B. P. Jayasekara, "Computer Vision Based Fire Alarming System", in Proceedings of the 2nd International Moratuwa Engineering Research Conference (MERCon), pp. 325-330, Moratuwa, Sri Lanka, April 2016.