Flame Recognition in Video

Walter Phillips III Mubarak Shah Niels da Vitoria Lobo
Computer Vision Laboratory
Department of Computer Science
University of Central Florida
Orlando, Fl 32816
{wrp65547,shah,niels}@cs.ucf.edu

Abstract

This paper presents an automatic system for fire detection in video sequences. There are many previous methods to detect fire, however, all except two use spectroscopy or particle sensors. The two that use visual information suffer from the inability to cope with a moving camera or a moving scene. One of these is not able to work on general data, such as movie sequences. The other is too simplistic and unrestrictive in determining what is considered fire, so that it can be used reliably only in aircraft dry bays. Our system uses color and motion information computed from video sequences to locate fire. This is done by first using an approach that is based upon creating a Gaussian-smoothed color histogram to determine the fire-colored pixels, and then using the temporal variation of pixels to determine which of these pixels are actually fire. Unlike the two previous vision-based methods for fire detection, our method is applicable to more areas because of its insensitivity to camera motion. Two specific applications not possible with previous algorithms are the recognition of fire in the presence of global camera motion or scene motion and the recognition of fire in movies for possible use in an automatic rating system. We show that our method works in a variety of conditions, and that it can automatically determine when it has insufficient information.

1. Introduction

Visual fire detection has the potential to be useful in conditions that conventional methods cannot be used – especially in the recognition of fire in movies. This could be useful in categorizing movies according to the level of violence.

A vision-based approach also serves to supplement current methods. Particle sampling, temperature sampling, and air transparency testing are simple methods used most frequently today for fire detection. Unfortunately, these methods require a close proximity to the fire. In addition, these methods are not always reliable, as they do not always detect the combustion itself. Instead, they detect the byproducts of combustion, though these byproducts could be produced in other ways.

Existing methods of visual fire detection rely almost exclusively upon spectral analysis using rare and usually costly spectroscopy equipment. This limits fire detection to those individuals who can afford the high price of the expensive sensors that are necessary to implement these methods. In addition, this approach is still vulnerable to false alarms caused by objects that are the same color as fire, especially the sun.

There are two previous vision-based methods that seem promising [2][3]. However, both of these rely upon ideal conditions. In the first [2], camera initialization requires the manual creation of rectangles based upon the distance of portions of a scene from the camera. Because camera initialization is so difficult, the camera must also be stationary.

The second method [3] is based upon strictly grayscale images. Though computationally inexpensive, this method only works where there is very little that may be mistaken for fire (in Aircraft dry bays, the locations of study in the second method, there is almost nothing else to find). Once again, the camera must be stationary for this method to work.

The method described in our paper employs only color video input, does not require a stationary camera, and is designed to detect fire in nearly any environment. In addition, if it is available, it may be implemented more effectively through the use of spectral imagery, because the training method can use all available color information.

In this technique, first, a color predicate is built using the method presented in sections 2.1 and 2.2. Based upon both the color properties, and the temporal variation of a small subset of images (section 3), a label is assigned to each pixel location indicating the inference that each pixel is a fire pixel (section 4). Based upon some conditions also presented in section 4, we can determine if this test will be reliable. The reason this is an effective combination is explained in section 5. If the test to find fire has been successful, an erode operation (section 6) is performed to remove spurious fire pixels. A region-growing algorithm designed to find fire regions not

0-7695-0813-8/00 $10.00 © 2000 IEEE

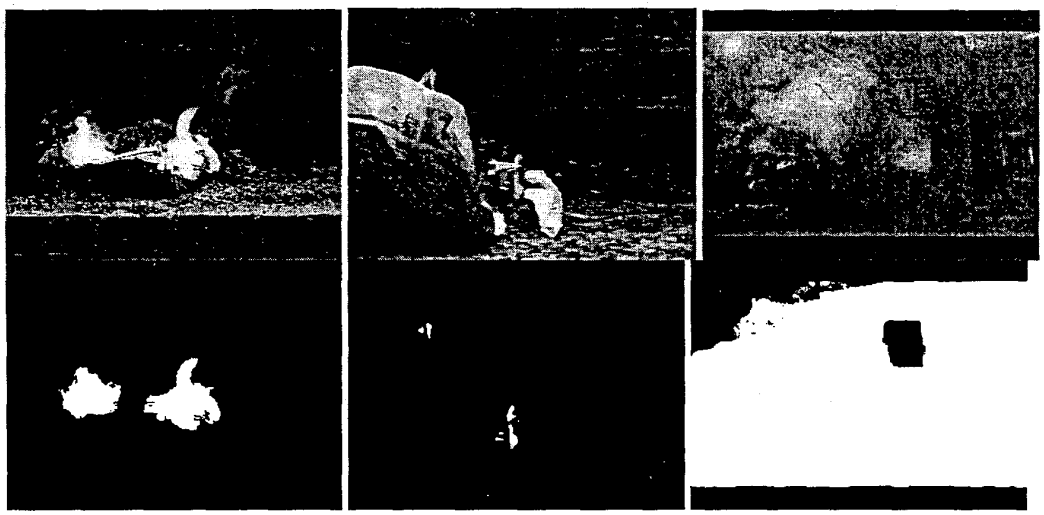

Figure 1: Some training images and their manually created masks

initially found follows this (section 7). An overall summary of the steps of this fire-finding algorithm is given in section 8. The results presented in section 9 show the effectiveness of this algorithm. Future work and conclusions follow in sections 10 and 11, respectively.

2.1. Color Detection

An often-used technique to identify fire is by models generated through color spectroscopy. We did not use this approach because models may ignore slight irregularities not considered for the type of burning material. Instead, our system is based upon training using test data from which the fire has been isolated manually to create a color lookup table, usually known as a color predicate. This is accomplished using the skin detection algorithm described in [4], which employs the creation of a Gaussian-smoothed color histogram that has been thresholded.

This adaptation allows for increased accuracy if training sequences are available for specific kinds of fires, while if training sequences are not available, it allows for a generic fire look-up table (assuming the user can create a generic, all-purpose fire probability table).

This algorithm for color lookup may be summarized by the following steps:

1) Create pairs of training images – each pair consists of a color image, and a Boolean mask, which specifies the locations that the target object occurs. For every pixel in each image that represents a color that is being searched for, there should be a "1" in the corresponding location in the Boolean mask, and a "0" for every background location. From our test, we found that using about ten training images from several of our data sets to be sufficient to construct an effective color predicate. Sample masks and images are shown in figure 1.
2) Construct a color histogram as follows: for every pixel location in the image, if the value in the corresponding mask location is "1" then add a Gaussian distribution to the area in the color histogram centered at the color value that corresponds to the color of the individual pixel. Otherwise, if the value in the corresponding mask location is "0," then subtract a smaller Gaussian distribution from the area in the color histogram centered at the color value that corresponds to the color of the individual pixel. For our work, the positive examples used a Gaussian with σ=2, and the negative examples used a Gaussian with σ=1.
3) Threshold the Gaussian smoothed color histogram to the desired level, resulting in a function which we shall call *Colorlookup,* which, given an (R,G,B) triple, will return a Boolean value, indicating whether or not an input color is in the color predicate.

2.2. Color in Video

Figure 2: Translucent fire with a book behind it.

Fire is gaseous, and as a result, in addition to becoming translucent, it may disperse enough to become undetectable, as in figure 2. This necessitates that we

average the fire color estimate over small windows of time.

A simple way to compute the probability that a pixel is fire-colored over a sequence is by averaging over time the probability that such a pixel is fire.
More precisely:

$$Colorprob(x,y) = \frac{\sum_{i=1}^{n} Colorlookup(P_i(x,y))}{n}$$

$$Color(x,y) = Colorprob(x,y) > k_1$$

where *Colorlookup* is the Boolean color predicate produced by the algorithm in section 2.1, n is the number of images in a sequence subset, P_i is the i^{th} frame in a sequence subset. $P_i(x,y)$ is the (R,G,B) triple found at location (x,y) in the i^{th} image, and k_1 is an experimentally determined constant. From our experimentation, we have determined that choosing n to be between 3 and 7 is sufficient.

Coloprob is a probability (between zero and one) indicating how often fire color occurs in the image subset in each pixel location, while *Color* is a predicate that indicates whether or not fire is present at all. From experimentation, we determined that fire must be detected at least 1/5 of the time by color to indicate the presence of fire. For this reason, we set k_1 to 0.2.

3. Finding Temporal Variation

Color alone is not enough to identify fire. There are many things that share the same color as fire that are not fire, such as a desert sun and red leaves.

The key to distinguishing between the fire and the fire-colored objects is the nature of their motion. Between consecutive frames (at 30 frames per second), fire moves significantly (see figure 3). The flames in fire dance around, so any particular pixel will only see fire for a fraction of the time.

For a sequence subset containing n images this temporal variation may be defined as:

$$DIFFS(x,y) = \frac{\sum_{i=2}^{n} \left| I(P_i(x,y)) - I(P_{i-1}(x,y)) \right|}{n-1}$$

where P_i is the i^{th} frame in a sequence of n images, and I is a function that given an (R,G,B) triple, returns the intensity (which is (R+G+B)/3).

Figure 3:
Flames flickering in two consecutive images

The highest possible temporal variation occurs in the case of flicker, that is, when a pixel is changing rapidly from one intensity value to another. This generally occurs only in the presence of fire. Motion of rigid bodies, in contrast, produces lower temporal variation..

By first correcting for the temporal variation of non-fire pixels, it is possible to determine if fire-colored pixels actually represent fire. This is done by:

1) Deciding which pixels are fire candidates using *Color.*
2) Finding the average change in intensity of all non-fire pixels
3) Subtracting this average value from the value in *DIFFS* at each location.

More precisely:
First compute the following:

$$nonfireDiffs = \frac{\sum_{x,y,Color(x,y)=0} DIFFS(x,y)}{\sum_{x,y,Color(x,y)=0} 1}$$

The lower summation represents the number of pixels in the image that are computed to be fire colored.
After computing *nonfireDiffs,* compute σ:
Figure 4 shows the importance of the temporal variation

$$\sigma(x,y) = DIFFS(x,y) - nonfireDiffs$$

because now the algorithm correctly rejects the part of the scene that is fire-colored.

Figure 4:
The Sun in this image is fire colored. This is not detected as fire by our system because the sun has low temporal variation.

4. Heuristic Analysis

Since the test to find fire is directly dependent upon both color and temporal variation, it is best expressed by a simple conjunction:

$$Fire(x,y) = \begin{cases} 1 \text{ if } Color(x,y) \text{ and } \sigma(x,y) > k_2 \\ 0 \text{ otherwise} \end{cases}$$

where k_2 is an experimentally determined constant.

This is a binary measure of the temporal variation of the fire-colored pixels. There are several exceptions that indicate that merely computing the predicate *Fire* is not enough. The first of these occurs specifically in sunlight.

Sunlight may reflect randomly, causing new light sources to appear and disappear. For that reason, there are often some pixels in an image containing the sun that have temporal variation high enough to be recognized as fire. Such sequences, which contain a high number of fire-colored pixels, but which have a low number of fast moving fire-colored pixels must usually be set into a "fire unlikely/undetectable" class. Specifically, this means counting the number of pixels in that are "1" in the predicate *Fire* and comparing it to total number of fire-colored pixels (i.e. those that are "1" in *Color*). If the number of fire colored pixels is less than some threshold, then we say that there is no fire in the sequence at all. For our tests, this threshold was 10 pixels. If the number of pixels detected as fire is greater than this threshold, but the ratio of pixels that are "1" in *Fire* to fire-colored pixels is low, then the sequence must be placed into the "fire unlikely/undetectable" class. For our tests, no more than one out of every thousand fire-colored pixels is found to be in the predicate *Fire*, then the sequence subset is not put into the "fire unlikely/undetectable" class.

There is one other case that contains fire that this method is unable to detect: if a sequence is recorded close enough to a fire, the fire may fully saturate the images with light, keeping the camera from observing changes or even colors other than white. Therefore, if contrast is very low and intensity is very high, as in figure 5, sequences are put into a "fire likely/undetectable" class.

Figure 5: Fire Likely/Undetectable

5. Independence of Color and Motion

It is possible that color and motion information could result in the same information so that knowing one is the same as knowing the other. In order to determine the correlation, we took a random sampling of 81,000 points from video data used in our experiments. For each point, we stored

1. The value of *DIFFS*
2. The value of *COLOR*

We then computed ρ the correlation coefficient:

$$\rho = \frac{\sum (x_i - \mu_x)(y_i - \mu_y)}{(n \cdot \sigma_x \sigma_y)}$$

where x_i is the i^{th} sample taken from *Color*, y_i is the i^{th} sample taken from *DIFFS*, n is the size of the sample, μ_x and μ_y are the sample means of *Color* and *Diffs*, and σ_x and σ_y are the sample variances taken from *Color* and *Diffs*, respectively. The correlation we measured by this method was .072, indicating that these two cues are independent.

6. Dealing with Reflection:

One of the largest problems in the detection of fire is the reflection of fire upon the objects near the fire. However, barring surfaces with high reflectivity, such as

Before Erosion **Fire Detected.**

Figure 6: Reflection on ground detected at lower left. In this and all examples, the detected location of fire is outlined in white.

mirrors, reflections tend to be incomplete. An erode operation can eliminate most of the reflection in an image.

For our study, the following erode operation worked the best: examine the eight-neighbors of each pixel. Remove all pixels from *Fire* that have less than five eight-neighbors. Figure 6 shows the results of this stage.

7. Region Growing

The output from the erosion stage will contain only the most likely fire candidates; to have avoided false positives thus far, our conservative strategy will not have detected all of the fire in a sequence subset. Thus, this is not an accurate measure of the total quantity of fire in the sequence subset. For one thing, some of the fire in a sequence will not appear to be moving because it is right in the center of the fire. Hence, in order to find the rest of

the flame, it is necessary to grow regions by examining color alone.

To find the total quantity of fire pixels in the sequence subset, the following region growing algorithm is applied:

1. Create a new Boolean image, *Fire'* and a variable, *dist*. Then set
   Fire'(x,y)← *Fire(x,y)*, and set *dist* ← *0*
2. For all pixels in *Fire* that are eight-neighbors of pixels *(x',y')* such that *Fire(x',y')=1*, test the following:
   if *Colorprob(x,y)* >*(k3+dist) set Fire'(x,y)*← *1*
3. *Fire(x,y)*← *Fire'(x,y)*
4. *dist*← *dist+k4*
5. Loop to step 2.

Where *dist* is a threshold that begins at zero and is incremented and both k_3 and k_4 are experimentally determined constants. In our experiments, we used 0.1 and *0.025* for k_3 and k_4, respectively. Recall that *Colorprob* is the probability lookup table described in section 2.2. This method is applied until there is no change from one step to the next (i.e., when no steps have any effect on *Fire).*

8. Algorithm for Fire Detection

The steps in the algorithm are the following:

1. Manually select fire from images and create a color predicate using the algorithm in [4] and summarized in section 2.1. Create a function that, given an (R,G,B) triple, returns a real number. Call this *Colorlookup.*

2. For n consecutive images, calculate *DIFFS, Colorprob,* and *Color*

$$DIFFS(x,y) = \frac{\sum_{i=2}^{n} \left| I(P_i(x,y)) - I(P_{i-1}(x,y)) \right|}{n-1}$$

$$Color(x,y) = Colorprob(x,y) > k_1$$

$$Colorprob(x,y) = \frac{\sum_{i=1}^{n} ColorLookup(P_i(x,y))}{n}$$

where *Colorlookup* is the predicate created in step #1 and k_1 is an experimentally determined constant.

3. Determine the motion of the part of the image that is not fire, and compute from each value in sigma. First calculate:

$$nonfireDiffs = \frac{\sum_{x,y,Color(x,y)=0} DIFFS(x,y)}{\sum_{x,y,Color(x,y)=0} 1}$$

and then calculate:

$$\sigma(x,y) = DIFFS(x,y) - nonfireDiffs$$

where the summation is over the *(x,y)* such that *Color(x,y)*<k_1, and k_1 is an experimentally determined constant.

4. Create a fire Boolean image,

$$Fire(x,y) = \begin{cases} 1 \text{ if } color(x,y) \text{ and } \sigma(x,y) > k_2 \\ 0 \text{ otherwise} \end{cases}$$

where k_2 is an experimentally determined constant.

5. Classify sequence as "fire likely/undetectable" if the average intensity is above some experimentally determined value, k_3.

6.a. Calculate the the total number of 1's in *Color.* Call this number *Numfire.*
   b. Calculate the the total number of 1's in *Fire.* Call this number *Foundfire.*
   c. Calculate *Foundfire/Numfire.* If this value is less than some experimentally determined constant, k_5 classify the sequence as "fire unlikely/undetectable."

7. Examine the eight-neighbors (the eight adjacent pixels) of each pixel. Remove all pixels from *Fire* that have less than five eight-neighbors that are 1.

8.a. Create a new Boolean image, *Fire'* and a variable, *dist* and set
   Fire'(x,y)← *Fire(x,y). dist*← *0*
   b. For all pixels in *Fire* that are eight neighbors of pixels such that *Fire(x,y)=1,* if *Colorprob(x,y)>(k3+dist)*
   Fire'(x,y)=1
   c. *Fire*← *Fire'*
   d. *dist*← *dist+k4*
   e. Loop to step b.

where *dist* is a constant that begins at zero and is incremented, k_3 and k_4 are experimentally determined constants.

Sequence	Length	Frames w/Fire	False +	False -	Description
Movie 1	598	442	2	14	A burning building
Movie 2	45	45	0	0	Fire in a fireplace
Movie 3	251	251	0	48	Big Candles
Movie 4	44	44	0	44	Small candles
Movie 5	36	36	0	0	Police Car with Fire Behind it
Movie 6	85	0	0	0	A setting sun
Movie 7	284	248	0	16	Fire in a forest
Movie 8	35	35	0	4	Fire in background
Movie 9	41	41	0	3	Homemade recording

Figure 7: A subset of the images tested. All measurement are in number of frames

9. Experimental Results:

Figure 8: The sun is not recognized, even with global motion.

This method has been effective for a large variety of conditions (see figure 7). False alarms, such as video of the sun moving (see figure 8) are not detected by this method because in all realistic sequences, the rate of global motion is almost always much less than the expected speed of the fire.

Lighting conditions also have no effect upon the system; it has been able to detect fire in a large variety of fire images, as in figure 9.

Certain types of fires, such as candles, blow torches, and lighters, are completely controlled, and always burn exactly the same way without flickering (see figure 10). Unfortunately, the algorithm fails for these cases because of the lack of temporal variation. However, these cases are not usually important to recognize because controlled fires are not dangerous.

10. Future Work

The next step in the development of this algorithm would be error reduction. There are three equations stated in this algorithm that have constants that must be determined experimentally. Employing training to determine these values could reduce the error in this method. Because of the low computational demand necessary for this algorithm, it is also possible to use it as part of a robust, real-time system for fire detection. Another direction would be to distinguish between different types of fires. Finally, predicting fire's path in video would be interesting for fire prevention.

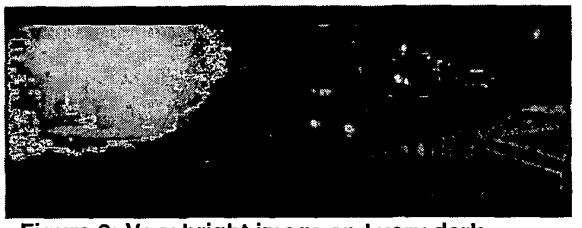

Figure 9: Very bright image and very dark image; detection occurs in both cases.

Figure 10: Detecting a match or candles means detecting based mostly upon color.

11. Conclusion

This paper has presented a robust system for detecting fire in color video sequences. This algorithm employs information gained through both color and temporal variation to detect fire. We have shown a variety of conditions in which fire can be detected, and a way to determine when it cannot. Through these tests, this method has shown promise for detecting fire in real world situations, and in movies.

References

[1]Cleary, T., Grosshandler, W., Survey of fire detection technologies and system evaluation/certification methodologies and their suitability for aircraft cargo compartments, .US. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 1999

[2]Healey, G., Slater, D., Lin, T., Drda, B., Goedeke, A.D. A system for Real-Time Fire Detection, IEEE Conf Computer Vision and Pattern Recognition, 1994.

[3]Foo, S. Y. A rule-based machine vision system for fire detection in aircraft dry bays and engine compartments, Knowledge-Based Systems, vol 9 531 41.

[4]Kjedlsen R, Kender, J. Finding Skin in Color Images, 1996 Face and Gesture Recognition P312-317.