Computer Vision Based Fire Detection

Nicholas True
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093
ntrue@ucsd.edu

Abstract

In this paper we use a combination of techniques to detect fire in video data. First, the algorithm locates regions of the video where there is movement. From these regions fire-colored pixels are extracted using a perceptron. Lastly, we use dynamic texture analysis to confirm that these moving, fire-colored regions have the temporal and motion characteristics of fire.

1. Introduction

Automatic fire detection devices have been around since the first smoke alarm was invented by Francis Upton in 1890 [2]. After further technological advances in the mid 1960s reduced the price of smoke detectors, these devices started showing up in buildings all over the world, becoming the ubiquitous and essential devices that they are today [2]. However, automated fire detection devices such as smoke detectors have some significant limitations which make them useless in many important situations. For instance, smoke detectors, the most common type of fire detection device, only work well in small enclosed spaces like those find in homes and offices. However, in large open spaces such as warehouses, atriums, theaters, and the outdoors, smoke detectors are ineffective because they require smoke to build up to sufficient levels to set them off. In open spaces, the fire is usually out of control by the time smoke has built up sufficiently to set off the alarm. Heat sensors suffer from the same shortcomings as smoke detectors.

Video-based fire detection does not suffer from the space constraints that smoke and heat detection do. Cameras can detect and pinpoint fire from long distances as soon as the fire starts, allowing the fire to be dealt with before it gets out of control. Furthermore, cameras can cover very large areas, potentially mitigating their high cost compared to other fire detection technologies. Video-based fire detection even has the potential to be placed on mobile platforms such as planes and robots.

2. Approach

Since fire is a complex but unusual visual phenomenon, we decided upon a multi-feature-based approach for our algorithm. The hope and the goal of such an algorithm is to find a combination of features whose mutual occurrence leaves fire as their only combined possible cause.

Fire has distinctive features such as color, motion, shape, growth, and smoke behavior. For this project we focused on features such as color and motion and we hope to include additional feature analysis in future work.

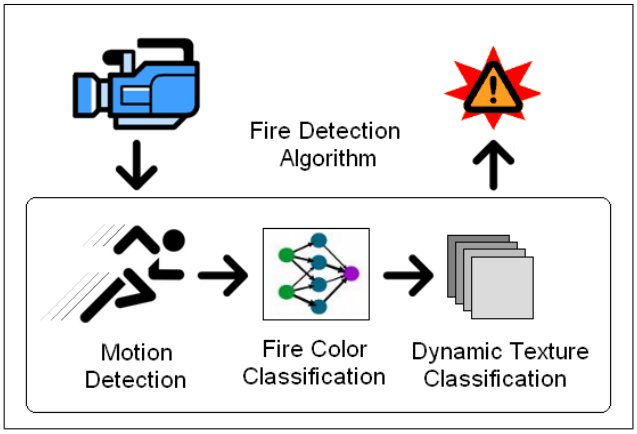

Figure 1. The fire detection algorithm outline.

1234

To reduce the total computational load, the first step of our algorithm is to perform frame differencing to get a rough idea where motion occurred. The regions of the video which are moving are fed to a fire color classification algorithm.

There are a number of different ways of detecting fire-colored pixels. In [15], the authors used a mixture of Gaussian in the RGB color space to classify pixels as being fire-colored or not. In [7], pixels whose color landed in a spe-

[1] faculty.ksu.edu.sa
[2] www.sumo1.com
[3] hubblesite.org
[4] cksinfo.com

cific segment of the HSV color space were classified as being fire colored. We decided to use a multilayer perceptron, like [9], to classify pixels as being fire colored or not. Spatial clustering and analysis is performed on the pixels that get classified as being fire colored.

The next stage involves grabbing a short 50 to 100 frame sequence of the video focused and centered at each of these moving fire-colored regions. These short video sequences are then fed to a module which performs dynamic texture analysis it. The result is that the target region is either classified as being fire or not.

2.1. Motion Detection

The first step for our algorithm is to find regions in the video stream where there is motion. This is done through frame differencing based on image intensity values.

Given that flames often flicker and jump, the algorithm has a built in timer which keeps track how long it's been since there's been movement at a particular pixel. This helps 'smooth' out the results for the query, 'where is there motion in the video'.

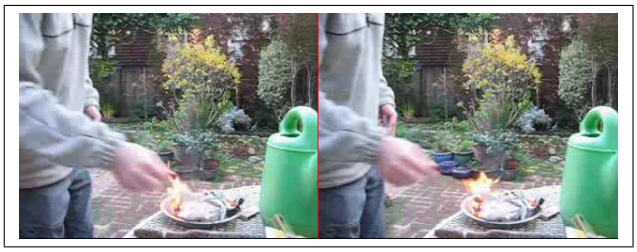

Figure 2. Original frame sequence.

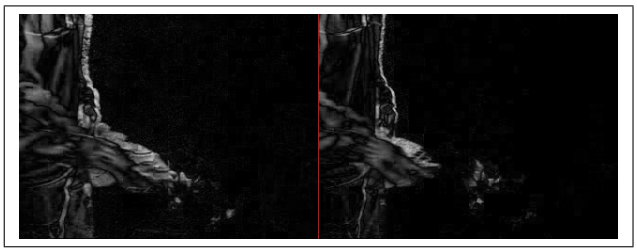

Figure 3. motion detected by frame differencing.

2.2. Detecting Fire-Colored Pixels

To classify pixels as being fire colored or not, we decided to use a multilayer perceptron. The perceptron has two layers, three input nodes for each of the color channels, and one node in the hidden layer. (Given that perceptrons are well described in the literature and known in many fields, I will forgo their description here.)

At first glance one might say that fire color classification fails miserably because it tends to label lots of non fire pixels as being fire. However, the goal of this first classifier is

Figure 4. Original image.

Figure 5. Red denotes pixels that were classified as being the color of fire.

to get a very high true positive rate and a low false negative rate, irregardless of the false positive rate. This is because the color classifier is the first in a sequence of classifiers whose job is to weed out the color classifier's false positives. Thus, these additional classification algorithms will help to reduce the overall false positive rate.

One of the advantages of choosing a perceptron to perform color classification is that it is easy to retrain. Given that the input images might be coming from different types of cameras with different levels of saturation and different color ranges, ease of retraining is an important feature. This way, it is easy to apply this technology to existing camera systems with a minimum amount of time and difficulty.

2.3. Motion + Color

The follow images are the result of passing the image through the motion detection module and then passing those results through the fire-color classifier.

As you can see, these combined results are much more

Figure 6. Fire-color classification sans motion detection.

Figure 7. Fire-color classification plus motion detection.

accurate than just motion or color alone. Furthermore, by only passing regions of the image that display movement, significantly fewer pixels have to get fed to the color classifier, resulting in sizable time savings. Chaining these classifiers also reduces the size and number of regions that the dynamic texture classifier has to check.

2.4. Dynamic Texture Analysis

The idea behind dynamic textures is that certain video sequences of moving scenes exhibit specific stationarity properties [5]. Using the statistical properties of sequences such as those created by water, smoke, and fire, it is possible to create an autoregressive moving average (ARMA) model for the target image sequence. One can even use this method to recognize and segment out parts of images based on their dynamic texture characteristics as was shown in [13] and [6] respectively. In our case, we just want to recognize if a given dynamic texture is that of fire or not, since we have already determined the location of the potential fire region and thus don't need the video to be segmented.

A thorough treatment of dynamic textures can be found in [5, 6, 13], however, we think it is well worth covering the basics here in this paper.

2.4.1 Dynamic Texture Definition

Generally speaking, dynamic texture analysis treats a sequence of images from a video as a time-evolving state process $x_t \in \mathbb{R}^n$, and "the appearance of frame $y_t \in \mathbb{R}^m$ is a linear function of the current state vector with some observation noise" [3]. Put more formally, y_t is the output of the ARMA model

$$\begin{cases} x_t = Ax_{t-1} + v_t \\ y_t = Cx_t + w_t \end{cases}$$

where $A \in \mathbb{R}^{n \times n}$ is the state-transition matrix, $C \in \mathbb{R}^{m \times n}$ is the observations matrix, $v_t \sim_{iid} \mathcal{N}(0, Q)$, $w_t \sim_{iid} \mathcal{N}(0, R)$, and where

$$\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \geq 0$$

is a positive semi-definite matrix where $S = \mathrm{E}[w_t v_t^T]$ [3, 13].

2.4.2 Finding the Model Parameters

Given a sequence of images $y_{t=1,\ldots,\tau}$ we want to compute the maximum likelihood solution

$$\hat{A}, \hat{C}, \hat{Q}, \hat{R} = \arg \max_{A,C,Q,R} P(y_1, ..., y_\tau | A, C, Q)$$

[5, 13]. Unfortunately, this would be quite difficult for current computers to compute. Thus, the following suboptimal closed form solution was proposed in [5, 6, 13].

Let $Y^\tau = [y_1, ..., y_\tau] \in \mathbb{R}^{m \times \tau}$. Now compute the singular value decomposition (SVD) of Y^τ to get $Y^\tau = U \Sigma V^T$. From this we can compute the following

$$\hat{C} = U, \ \hat{X} = \Sigma V^T$$

and

$$\hat{Q} = \frac{q}{\tau - 1} \sum_{i=1}^{\tau - 1} \hat{v}(i) \hat{v}^T(i)$$

and

$$\hat{Q} = \hat{B} \hat{B}^T$$

and

$$\hat{A} = \Sigma V^T D_1 V (V^T D_2 V)^{-1} \Sigma^{-1}$$

where

$$D_1 = \begin{bmatrix} 0 & 0 \\ I_{\tau-1} & 0 \end{bmatrix}, \; D_2 = \begin{bmatrix} I_{\tau-1} & 0 \\ 0 & 0 \end{bmatrix},$$

[5]. Thus, using the previous equations, one can compute the parameters of the dynamic texture model $M = \{\hat{A}, \hat{C}\}$.

2.4.3 Recognizing Different Dynamic Texture Models

The next step is to recognize different dynamic texture models via a nearest neighbor (NN) classifier which is utilizing Martin Distance [11] as the means to compare dynamic texture models.

The intuition behind Martin Distance is to compare the subspace angles between extended observability matrices of two dynamic textures [4]. Put more formally, let $M_a = \{A_a, C_a\}$ be the first dynamic texture model and let $M_b = \{A_b, C_b\}$ be the second dynamic texture model. The square of the Martin Distance between the two models is defined to be

$$d^b(M_a, M_b) = -\log \prod_{i=a}^{n} \cos^b \theta_i$$

where θ_i are the angles between the extended observability matrices of M_a and M_b [4].

The 64 thousand dollar question is therefore, how does one compute θ_i? First let's define the extended observability matrices of M_a and M_b to be

$$\mathcal{O}_i = \begin{bmatrix} C_i \\ A_i^T C_i \\ \vdots \\ (A_i^T)^n C_i \\ \vdots \end{bmatrix}$$

where $i = \{a, b\}$ and $\mathcal{O}_i \in \mathbb{R}^{\infty \times n}$ [4]. It turns out that $\cos\theta_i$ for i=1, ..., n are just the n largest eigenvalues found by solving the generalized eigenvalue problem

$$\begin{bmatrix} 0 & \mathcal{O}_{ab} \\ \mathcal{O}_{ab}^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} \mathcal{O}_{aa} & 0 \\ 0 & \mathcal{O}_{bb}^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

subject to $x^T \mathcal{O}_{aa} x = 1$ and $y^T \mathcal{O}_{bb} y = 1$ where

$$\mathcal{O}_{pq} = \mathcal{O}_p^T \mathcal{O}_q = \sum_{j=0}^{\infty} (C_p A_p^j)^T C_q A_q^j$$

for some dynamic texture models p and q [4]. At this stage it is handy to note that infinite sums of the form

$$X = \sum_{j=0}^{\infty} A^j Q (A^j)^T$$

can also be written as

$$X_j = A X_{j-1} A + Q^T$$

where in steady state the equation becomes

$$X = AXA + Q^T$$

But this is just the *Lyapunov equation* [1] and can be solved in Matlab using the function `dlyap.m`.

Thus, all one has to do to compute the Martin Distance between M_a and M_b is to compute the extended observability matrices, solve the generalized eigenvalue problem, take the largest n eigenvalues, and compute the Martin Distance. With this distance metric, a nearest neighbor algorithm can be used to classify dynamic texture models as being that of fire or not.

3. Data

The training and testing videos for this project were created using a *Canon PowerShot A540* digital camera. All of the videos are 320×240 and 15fps. We chose such a low power, low resolution camera in an attempt to simulate the poor quality of many CCTV camera networks that this algorithm might be applied to.

The data sets consist of indoor, outdoor, day, and night video sequences with a variety of lighting conditions and distances to the fire in the video. There are 41 training videos and 26 test videos.

We manually created 141 segmented fire videos and 48 segmented non-fire videos for dynamic texture training. We also created 67 segmented fire videos and 24 segmented non-fire videos to test the dynamic texture algorithm.

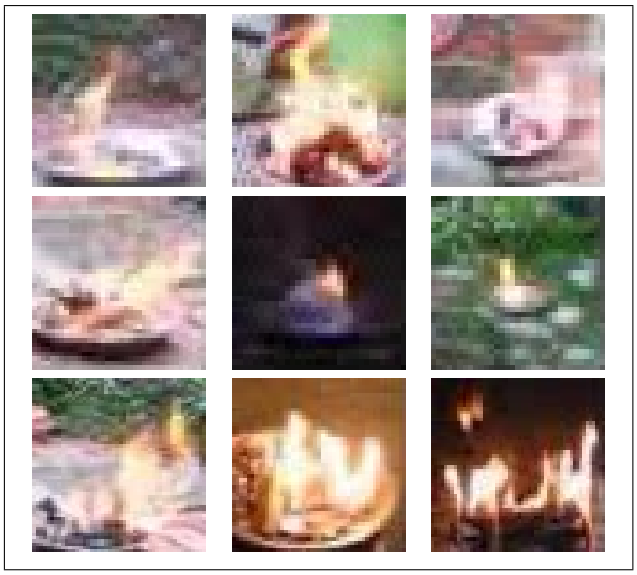

Figure 8. Segmented fire videos.

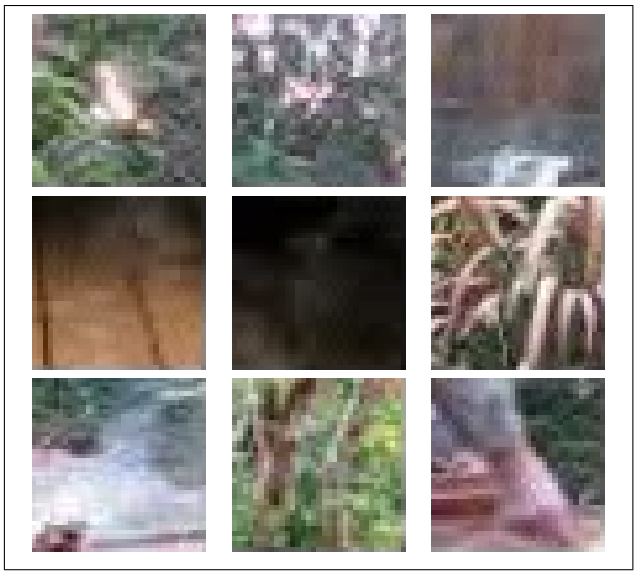

Figure 9. Segmented non-fire videos.

It is important to note that the training and testing segments were created from the raw training and testing video sets. Furthermore, there are no overlapping frame sequences in either the segmented training or testing data. However, all of the videos were created in similar settings and lighting situations. Thus, the following results say nothing about this particular algorithms ability to generalize to as yet unseen and unknown situations. However, given that the algorithm is based on a nearest neighbor classifier, the larger and more varied the training the better.

4. Results

The following are the results for the dynamic texture classification part of the algorithm. The results are independent of the color classification or motion detection algorithms.

	Non-Fire	Fire	Combined
Correctly Classified	33%	97%	80%

5. Analysis

As we can see, the results for the dynamic texture classification algorithm are good and reasonably consistent with past results in the literature. It is encouraging to know that through the use of a kernelized dynamic texture classification method such as the one proposed in [3], it is possible to achieve up to a 97.5% correct classiciation rate.

One thing that stood out were the poor results in classifying non-fire video segments correctly. Given that the algorithm in question is a nearest neighbor classifier and given that these results are based off of a very small training set with an even smaller training subset of only 48 non-fire video sequences, it is reasonable to expect that the results could improve significantly from a more training examples.

There is an old adage that goes, *where there's smoke there's fire*. Due to time limitations, we were unable to take advantage of this bit of wisedom. However, we can safely say that our accuracy could be significantly improved by the inclusion of a smoke feature detection algorithm which would bolster the current flame detection algorithms.

6. Conclusion

In conclusion, this paper demonstrates a multi-feature classification approach to detecting fire in video data. Furthermore, we introduce the concept of dynamic texture analysis to the existing body of video-based fire detection research and show that it works well in concert with other tried and true fire feature detection algorithms.

In our future work we will endeavor to incorporate smoke detection algorithms and fire reflection detection algorithms into our current algorithm. Furthermore, we will explore different ways of combining the different feature detection algorithms such as by using a Bayesian belief network or a boosting algorithm to improve the overall results.

7. Acknowledgments

I would like to thank Antoni Chan and Serge Belongie for all their help and advice on this project! And remember, only you can prevent forest fires. But for the times that you can't, bring marshmallows!

References

[1] Lyapunov equation. wikipedia.org. 4

[2] Smoke detector. wikipedia.org. 1

[3] A. Chan and N. Vasconcelos. Classifying video with kernel dynamic textures. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–6, June 2007. 3, 5

[4] K. D. Cock, K. D. Cock, B. D. Moor, and B. D. Moor. Subspace angles between linear stochastic models. In *In Proc. the 39th IEEE Conference on Decision and Control*, pages 1561–6, 2000. 4

[5] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. 51(2):91–109, 2003. 3, 4

[6] G. Doretto, D. Cremers, P. Favaro, and S. Soatto. Dynamic texture segmentation. volume 2, pages 1236–1242, Nice, France, Oct. 2003. 3

[7] J. Elbert and J. Shipley. Computer vision based method for fire detection in color videos. 1

[8] R. Hecht-Nielsen. *Perceptrons*. Imprints, 2008.

[9] W.-B. Horng and J.-W. Peng. Image-based fire detection using neural networks. In *Joint Conference on Information Sciences*. Atlantis Press, 2006. 2

[10] C. Lai, J. Yang, and Y. Chen. A real time video processing based surveillance system for early fire and flood detection. In *Instrumentation and Measurement Technology Conference Proceedings, IEEE*, 2007.

[11] R. Martin. A metric for arma processes. *Signal Processing, IEEE Transactions on*, 48(4):1164–1170, Apr 2000. 4

[12] W. Phillips, I. Mubarak, S. Niels, and V. Lobo. Flame recognition in video. *In Fifth IEEE Workshop on Applications of Computer Vision*, 23:224–229, 2002.

[13] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. volume 2, pages 58–63, Kauai, Hawaii, USA, Dec. 2001. 3

[14] B. U. Töreyin, Y. Dedeoglu, and A. E. Çetin. Contour based smoke detection in video using wavelets.

[15] B. U. Töreyin, Y. Dedeoglu, U. Güdükbay, and A. E. Çetin. Computer vision based method for real-time fire and flame detection. *Pattern Recogn. Lett.*, 27(1):49–58, 2006. 1