

FIRE AND FLAME DETECTION METHODS IN IMAGES AND VIDEOS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Yusuf Hakan Habiboglu

August 2010

I certify that I have read this thesis and that in my opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. A. Enis Çetin(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Sinan Gezici

I certify that I have read this thesis and that in my opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Murat Köksal

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Levent Onural
Director of Institute of Engineering and Sciences

ABSTRACT

FIRE AND FLAME DETECTION METHODS IN IMAGES AND VIDEOS

Yusuf Hakan Habiboglu

M.S. in Electrical and Electronics Engineering

Supervisor: Prof. Dr. A. Enis Çetin

August 2010

In this thesis, automatic fire detection methods are studied in color domain, spatial domain and temporal domain. We first investigated fire and flame colors of pixels. Chromatic Model, Fisher's linear discriminant, Gaussian mixture color model and artificial neural networks are implemented and tested for flame color modeling. For images a system that extracts patches and classifies them using textural features is proposed. Performance of this system is given according to different thresholds and different features. A real-time detection system that uses information in color, spatial and temporal domains is proposed for videos. This system, which is developed by modifying previously implemented systems, divides video into spatiotemporal blocks and uses features extracted from these blocks to detect fire.

Keywords: Fisher's linear discriminant, artificial neural networks, covariance descriptors, codifference descriptors, Laws' masks, Gabor filters.

ÖZET

RESİMLERDE VE VİDEOLARDA ATEŞ VE ALEV TESPİT YÖNTEMLERİ

Yusuf Hakan Habiboglu

Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Prof. Dr. A. Enis Çetin

Ağustos 2010

Bu tezde renk düzleminde, uzamsal düzlemde ve zaman düzleminde ateş ve alev tespit yöntemleri üzerinde çalışılmıştır. İlk olarak ateş ve alev pixellerinin renkleri incelendi. Kromatik model, Fisher'in doğrusal ayrımı, Gausssal renk karışım modeli ve yapay sinir ağları tanımlandılar ve alev renk modeli için test edildiler. Resimler için resimleri parçalayan ve bu parçalardan elde ettiği öznitelik vektörlerini kullanarak parçaları sınıflandıran bir sistem önerilmiştir. Farklı eşik değerlerine ve farklı öznitelik vektörlerine göre olan sistem performansları verilmiştir. Videolar için renk, uzamsal, ve zamansal bilgileri kullanan gerçek zamanlı çalışan bir sistem geliştirilmiştir. Önceki sistemlerin değiştirilmesiyle oluşturulan bu sistem videoları zaman-uzamsal bloklara parçalar ve ateşi tespit etmek için bu parçalardan elde edilen öznitelik vektörlerini kullanır.

Anahtar Kelimeler: Fisher'in doğrusal ayrımı, yapay sinir ağları, kovaryans tanımlayıcıları, ortak fark tanımlayıcıları, Laws filtreleri, Gabor filtreleri

ACKNOWLEDGMENTS

I would like to express my special thanks to my supervisor A. Enis Çetin for his patience and guidance.

I gratefully acknowledge Sinan Gezici for his support, encouragement and advise.

I am grateful to Murat Köksal for his valuable contributions by taking place in my thesis defense committee.

I would also like to thank Osman Günay for his precious contributions to my thesis.

Lastly, I offer my regards to my friends.

Contents

1	INTRODUCTION	1
1.2	Early Works	1
2	FIRE AND FLAME DETECTION IN COLOR SPACE	4
2.1	Chromatic Model	5
2.2	Fisher's Linear Discriminant	6
2.3	Gaussian Mixture Color Model	10
2.4	Artificial Neural Network	10
2.5	Results and Summary	12
3	FIRE DETECTION IN IMAGES	15
3.1	Fire Detection with Patches	16
3.2	Covariance Descriptors	17
3.3	Codifference Descriptors	22
3.4	Laws' Masks	25
3.5	Gabor Filters	27
3.6	Markov Chain	30
3.7	Results and Summary	32
3.8	Covariance Descriptors For Video	35

3.9	Training and Testing	38
3.10	Results and Summary	39
4	CONCLUSIONS	47

List of Figures

2.1	Sample images from training set.	5
2.2	Characteristics of chromatic model. Rate vs R_T	7
2.3	Characteristics of chromatic model 2C. Rate vs R_T	8
2.4	Characteristics of Fisher's linear discriminant. Rate vs L_T	9
2.5	Characteristics of Gaussian mixture color model. Rate vs G_T	11
2.6	The artificial neural network model used in flame color detection.	11
2.7	Characteristics of artificial neural network model. Rate vs N_T	12
2.8	Classifications results of a fire containing image in color space.	13
2.9	Classifications results of an image in color space.	14
3.1	Classification of patches of a test image when various covariance descriptors are used.	20
3.2	Classification of patches of a test image when various covariance descriptors are used.	21
3.3	Classification of patches of a test image when various codifference descriptors are used.	23
3.4	Classification of patches of a test image when various codifference descriptors are used.	24
3.5	Classification of patches of a test image when Law's masks are used.	27

3.6	Classification of patches of a test image when Law's masks are used.	27
3.7	Classification of patches of a test image when Gabor filters are used.	29
3.8	Classification of patches of a test image when Gabor filters are used.	29
3.9	Classification of patches of a test image when Markov chains are used.	31
3.10	Classification of patches of a test image when Markov chains are used.	32
3.11	An example for decisions of patches of an image.	33
3.12	Positive and negative test blocks for $S_T = 3$	34
3.13	Sample image frames from videos.	40

To My Family ...

Chapter 1

INTRODUCTION

Automatic fire detection in images and videos is an important task. Especially early detection of fire is crucial. With early detection, small uncontrolled fires can be stopped before they turn into catastrophic events.

In this thesis a real-time system is developed step by step. Firstly, fire detection techniques in color domain is investigated. Then spatial information is taken into consideration. At last a real-time system that uses color, spatial and temporal information is implemented and tested with different parameters in video. Processing frame rate of developed system is generally at 20 fps. It works very well when fire is clearly visible.

1.2 Early Works

In [1], a Gaussian-smoothed histogram is used as a color model. A fire color probability is estimated by using several values of a fixed pixel over time. Considering temporal variation of fire and non-fire pixels with fire color probability a heuristic analysis is done. A solution to the reflection problem is proposed, and

a region growing algorithm is implemented.

After moving pixels are segmented, chromatic features are used for determining fire and smoke regions in [2]. Dynamics analysis of flames is done. According to this analysis, dynamic features of fire and smoke regions are used to determine whether it is a real flame or a flame-alias.

In another work [3], moving and fire colored pixels are found according to a background estimation algorithm and Gaussian mixture models. Quasi-periodic behavior in flame boundaries and color variations are detected by using temporal and spatial wavelet analysis. Irregularity of the boundary of the fire-colored region is taken into consideration at the decision step.

In [4], background estimation algorithm and chrominance model are used to detect both moving and fire colored pixels. Two Markov models are proposed to separate flame flicker motion from flame colored ordinary object motion. Same Markov model is used to evaluate spatial color variance.

An adaptive flame detector is implemented and trained by using weighted majority based online training in [5]. Outputs of Markov models representing the flame and flame colored ordinary moving objects and spatial wavelet analysis of boundaries of flames are used as weak classifiers for training.

In [6], HSI color model based segmentation algorithm is used to find fire colored regions. Image differencing, which is taking the absolute value of the pixel-wise difference between two consecutive frames, and same segmentation process are used to separate fire colored objects from flames. A method for estimating degree of fire flames is proposed.

The organization of this thesis is follows. In Chapter 2 flame color models are investigated. In Chapter 3 a method for fire and flame detection in images

is proposed. In Chapter 5 a real-time fire and flame detection algorithm is implemented.

Chapter 2

FIRE AND FLAME DETECTION IN COLOR SPACE

We first investigate fire and flame colors of pixels. In this part of the thesis we try to find a reasonable answer to the question of color modeling of fire and flame pixels in terms of accuracy and computational complexity.

We constructed train and test sets from fire and non-fire images. Fire pixels are manually chosen. Since this method is a basic step of a complex fire detection system our aim is to achieve a high detection rate with low complexity. In this part, false alarm rate is not as crucial as true detection rate because other algorithms can distinguish fire pixels from non-fire ones by using temporal and spatial information. As you can see from Figure 2.1, non-fire pixels in positive samples are painted to black. In negative training images there is not any image about autumn or sunset, since color of most of the pixels of these scenes are very

similar to color of the fire pixels. We tested chromatic model, Fisher’s linear discriminant, Gaussian mixture model and neural network models based on our training set. In [1] Gaussian-smoothed color histogram is suggested for pixel separation. This method thresholds color histogram to construct a look-up table which has 2^{24} elements. Due to its high memory requirements this method is not tested.

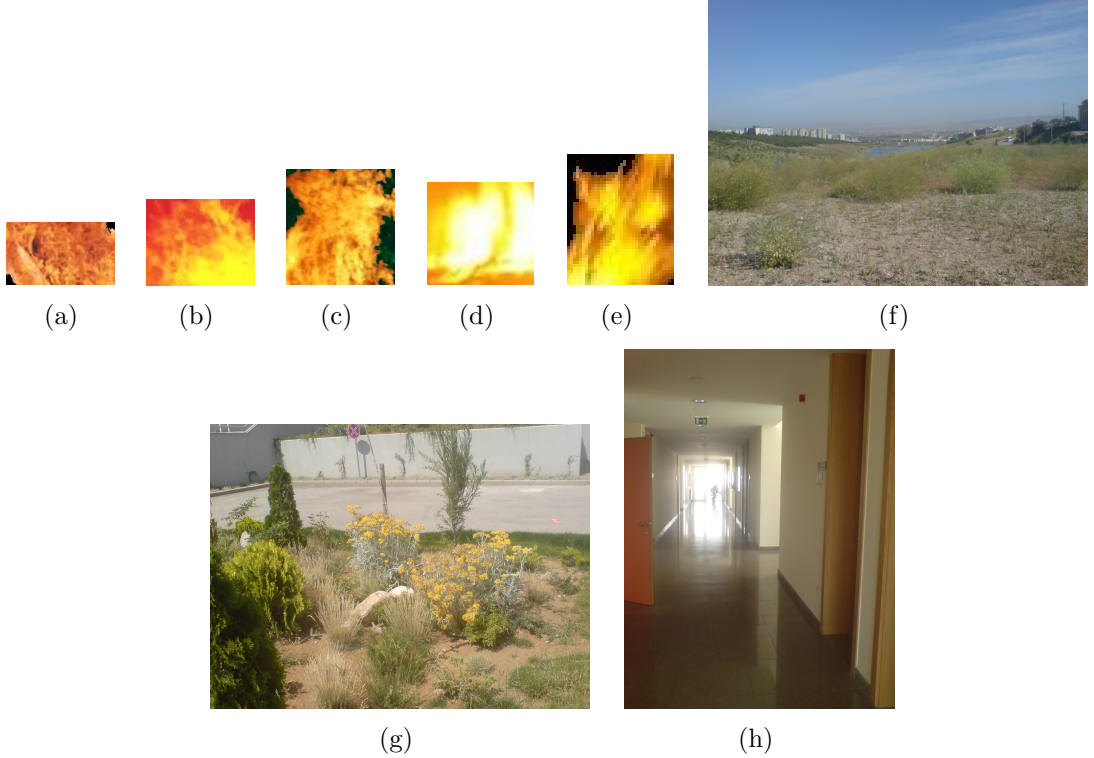


Figure 2.1: (a) to (e): samples from positive training set, (f) to (h): samples from negative training set

2.1 Chromatic Model

In [2] Chen, Wu and Chiou suggested a chromatic model to classify pixel colors. Its implementation is simple and its computational cost is low in terms of memory

and processing power. They analyzed fire colored pixels and realized that hue of fire colored pixels are in the range of 0° and 60° . RGB domain equivalent of this condition is

Condition 1. $R \geq G > B$

Since fire is a light source its pixel values must be bigger than some threshold. R_T is the threshold for red channel.

Condition 2. $R > R_T$

Last rule is about saturation. S is the saturation value of a pixel and Λ_T is the saturation value of this pixel when R is R_T .

Condition 3. $S > (255 - R)\Lambda_T R_T$

As it can be seen in Figure 2.2 true detection rate versus R_T forms a bell curve. We modified this model by excluding condition 3 to reduce the computational cost and tested this new model under the name chromatic model 2C. Classification results are presented in Figure 2.3.

2.2 Fisher's Linear Discriminant

Fisher's linear discriminant is a dimensionality reduction technique used in pattern recognition and machine learning. Most popular application of this method is Fisher faces. Lu et al. [7] proposed linear discriminant flame model. Both linear discriminant analysis and Fisher's linear discriminant assume that each class has a Gaussian distribution. The difference between the methods is the assumption of the covariance matrix. Linear discriminant analysis assumes that

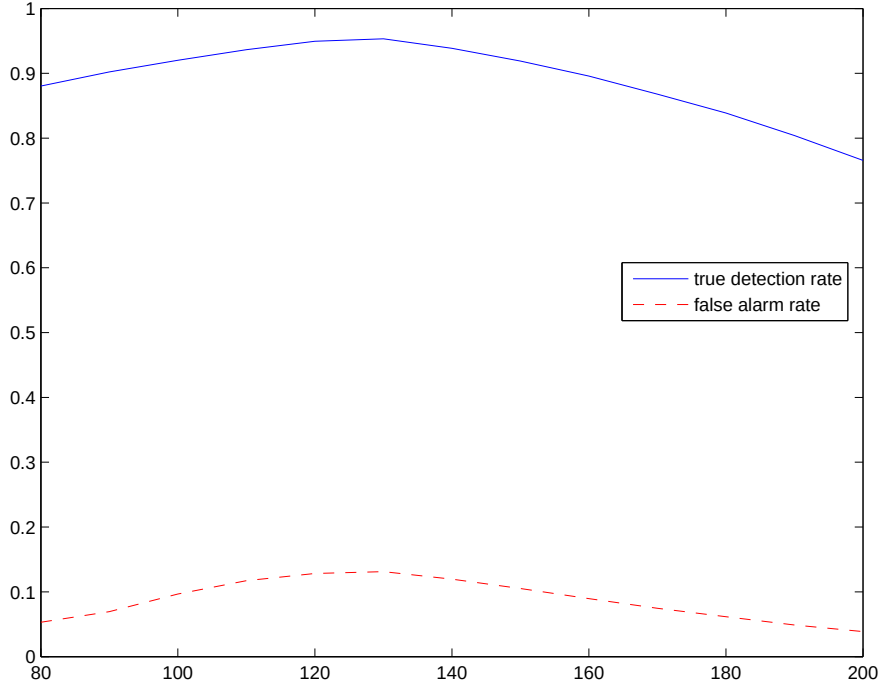


Figure 2.2: Characteristics of chromatic model. Rate vs R_T .

each class has identical covariance matrices. Covariance matrices do not have to be identical in Fisher's linear discriminant method, they can be different.

In our problem we have two classes, fire and non-fire colors. Since Fisher's linear discriminant assumes that each class has a Gaussian distribution, we can define each distribution by using its mean (μ) and its covariance matrix (Σ). In this method our aim is to find the projection vector, J_w , which maximizes S_w (2.5). Since projection of Gaussian distribution is also a Gaussian distribution the class means and class variances of projected data can be defined as equations

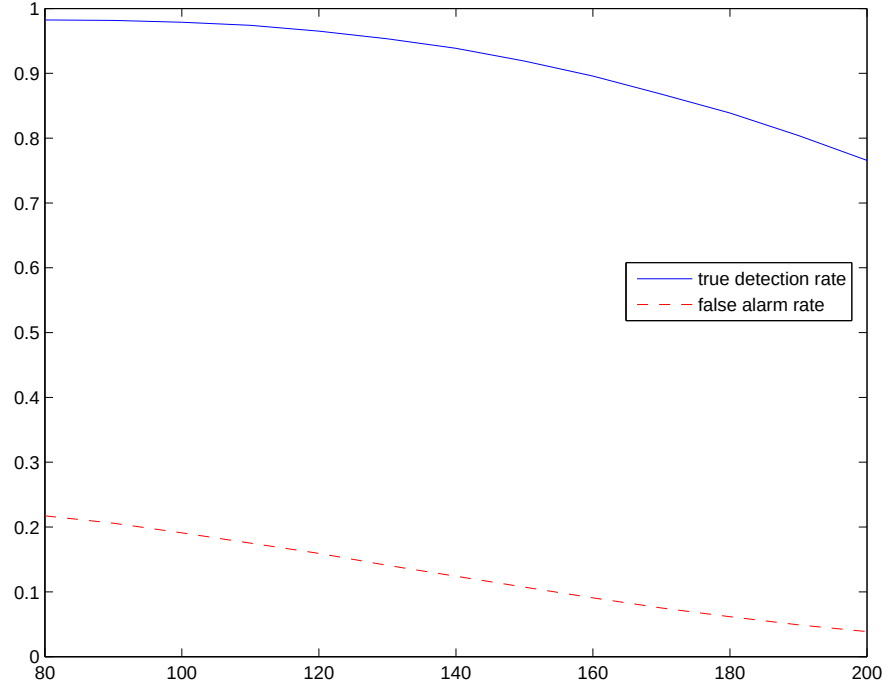


Figure 2.3: Characteristics of chromatic model 2C. Rate vs R_T .

(2.1), (2.2), (2.3), (2.4).

$$\mu_{fire} = J_w \mu_{fire} \quad (2.1)$$

$$\mu_{non-fire} = J_w \mu_{non-fire} \quad (2.2)$$

$$\sigma^2_{fire} = J_w \Sigma_{fire} J_w^T \quad (2.3)$$

$$\sigma^2_{non-fire} = J_w \Sigma_{non-fire} J_w^T \quad (2.4)$$

This method tries to maximize distance between projected class means as much as possible while keeping projected class variances as low as possible using S_w .

$$S_w = \frac{|\mu_{fire} - \mu_{non-fire}|^2}{\sigma^2_{fire} + \sigma^2_{non-fire}} \quad (2.5)$$

According to Fisher's linear discriminant for our two class problem the best projection vector, which maximizes S_w , is:

$$J_w = (\Sigma_{fire} + \Sigma_{non-fire})^{-1}(\mu_{fire} - \mu_{non-fire}) \quad (2.6)$$

A test pixel P is classified according to:

$$FLD(P) = \begin{cases} \text{true} & \text{if } \left| \frac{J_w P - \mu(w)_{non-fire}}{J_w P - \mu(w)_{fire}} \right| > L_T \\ \text{false} & \text{otherwise} \end{cases} \quad (2.7)$$

where L_T is a threshold. Classification results of this method are presented in Figure 2.4.

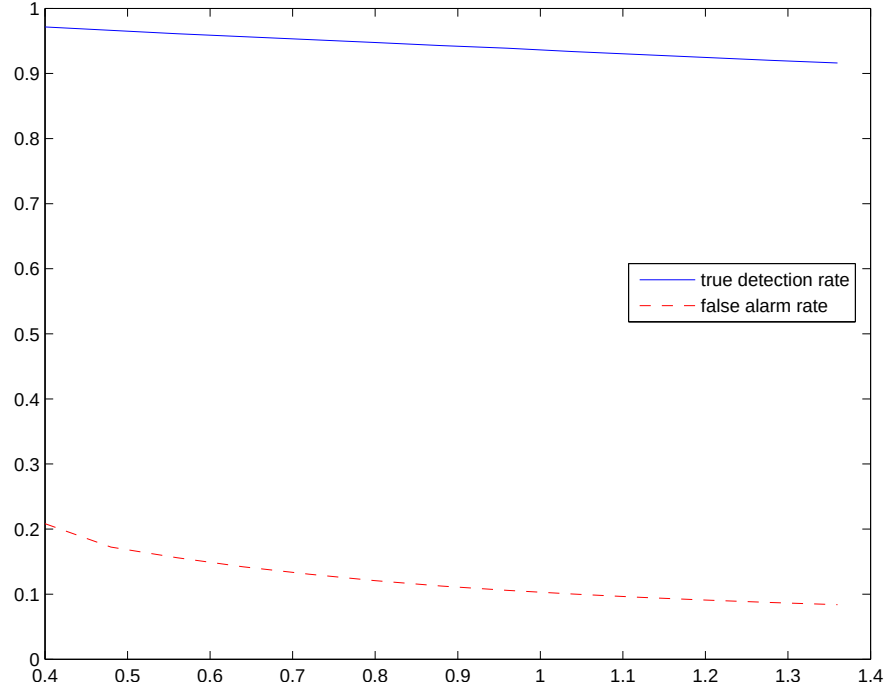


Figure 2.4: Characteristics of Fisher's linear discriminant. Rate vs L_T .

2.3 Gaussian Mixture Color Model

Gaussian mixture models are used for modeling distributions. With enough components it can model every distribution with an acceptable accuracy. A Gaussian mixture model is a linear combination of N gaussian distributions.

$$Gmm(P) = \sum_{i=1}^N W_i G(P; \mu_i, \Sigma_i) \quad (2.8)$$

$$\sum_{i=1}^N W_i = 1 \quad (2.9)$$

$$G(P; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp \left[-\frac{1}{2} (P - \mu)^T \Sigma^{-1} (P - \mu) \right] \quad (2.10)$$

In [3] and [8] a Gaussian mixture model is used for color classification. Our main difference with earlier works is using 2 Gaussian mixture models instead of a single model. One for fire colors and one for non-fire colors. We used 2 components ($N = 2$). Models are trained by using EM algorithm. Decision is obtained by thresholding the ratio of Gaussian mixture models with threshold G_T . Classification results of this method are presented in Figure 2.5.

$$GMCM(P) = \begin{cases} \text{true} & \text{if } \frac{Gmm_{fire}(P)}{Gmm_{non-fire}(P)} > G_T \\ \text{false} & \text{otherwise} \end{cases} \quad (2.11)$$

2.4 Artificial Neural Network

An artificial neural network is a mathematical model which is used for classification and regression. Artificial neural networks are inspired from neural networks of living things. In our problem we used a hidden layer and an output layer. In

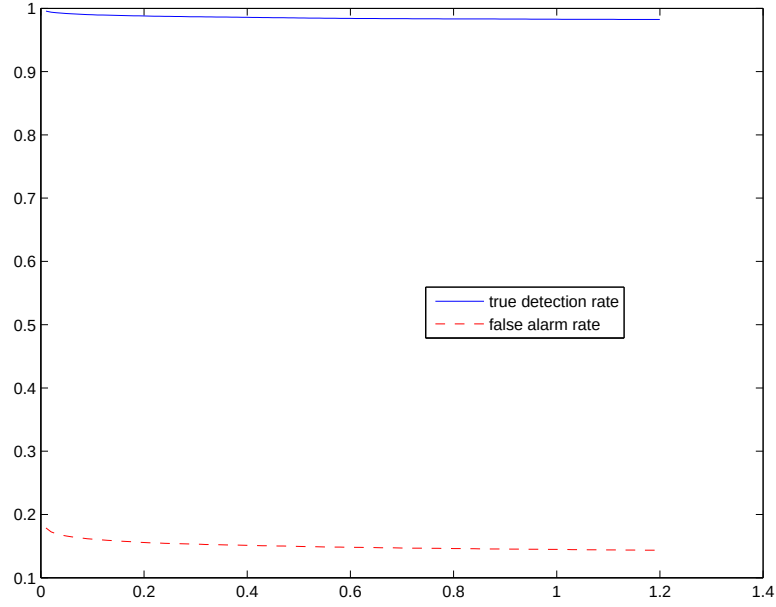


Figure 2.5: Characteristics of Gaussian mixture color model. Rate vs G_T .

hidden layer there is only one neuron. Low complexity and better speed performance is the reason behind using only one hidden neuron. In Figure 2.6 there is a scheme of the model we used. $Nnet(P)$ returns the output of this model, a value between 0 and 1.

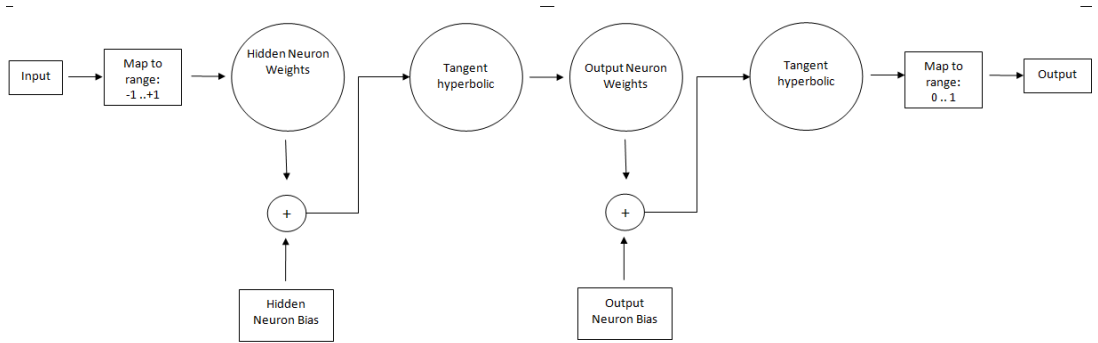


Figure 2.6: The artificial neural network model used in flame color detection.

Classification is done by thresholding the output of the artificial neural network with N_T . Classification results of this method are presented in Figure 2.7.

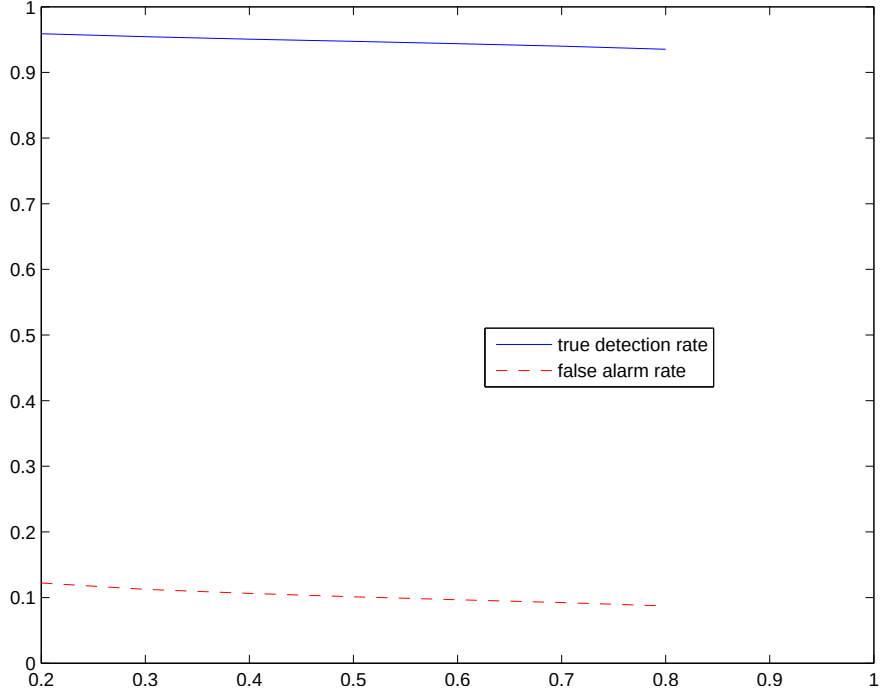


Figure 2.7: Characteristics of artificial neural network model. Rate vs N_T .

2.5 Results and Summary

We tested four methods for fire colored pixel detection. Consider the example image shown in Figure 2.8. As seen from results of each classifier with a proper threshold each model can provide a sufficient true detection rate. Generally false alarm rate and true detection rate decreases when threshold increases except for the chromatic model. Change of detection rate via threshold forms a bell curve in Chromatic model. Accuracy of chromatic model 2C is lower than other models as seen in Figure 2.8. It classifies reddish wall behind the flame as fire, but it is simple and fast. False alarms are not crucial in this step because the pixels which cause false alarms can be eliminated by later steps, which use spatial or temporal information.

In Figure 2.9 classifiers are tested with an image which does not contain fire.

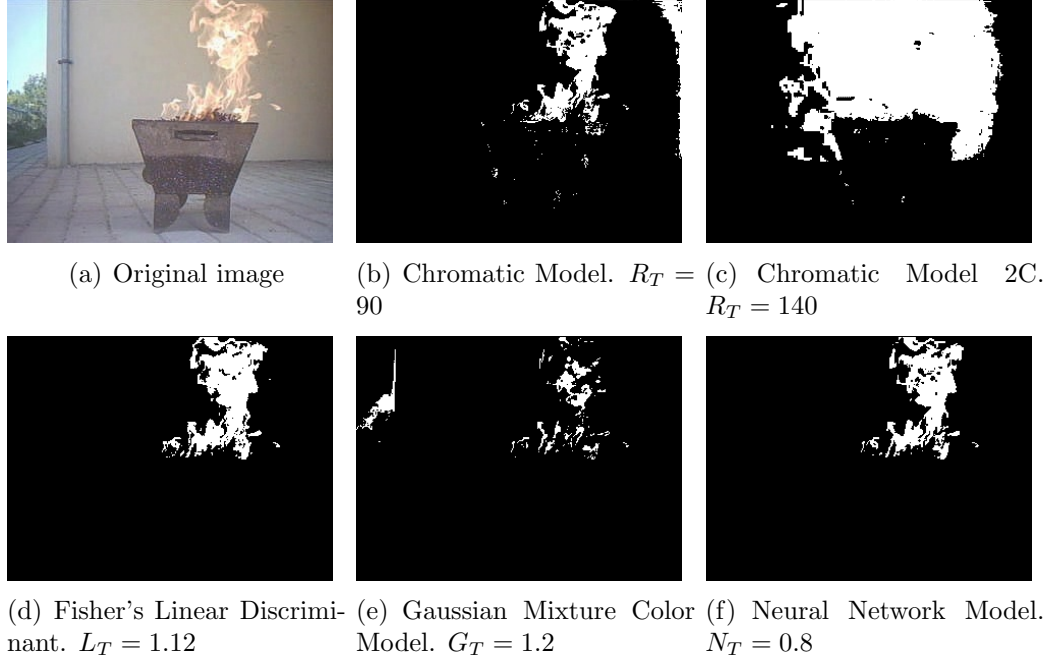
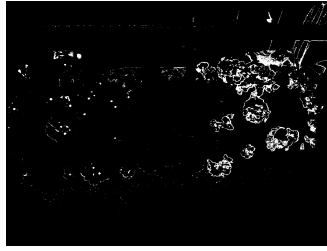


Figure 2.8: Classifications results of a fire containing image in color space.

Cameras have their own color balancing and light adjustment settings, therefore it is impossible to detect fire and flames using only the color information with a high detection accuracy.



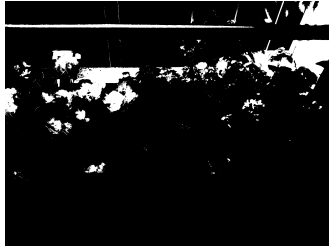
(a) Original image



(b) Chromatic Model. $R_T = 90$



(c) Chromatic Model 2C. $R_T = 140$



(d) Fisher's Linear Discriminant. $L_T = 1.12$



(e) Gaussian Mixture Color Model. $G_T = 1.2$



(f) Neural Network Model. $N_T = 0.8$

Figure 2.9: Classifications results of an image in color space.

Chapter 3

FIRE DETECTION IN IMAGES

In this chapter, fire detection problem in images is investigated. Since every video is actually a sequence of images, this problem occurs in all real time fire detection systems.

Although there are several works about fire detection in videos, as far as we know, there is not any work on fire detection in images. Of course by modifying or excluding some steps of fire detection algorithms, which are developed for videos, they can be used for fire detection in images, but none of them are developed for this aim. The study in this chapter analyzes fire detection problem in color space and spatial domain. It is assumed that fire has its own texture style and by using texture classification methods it can be separated from non-fire objects. But due to lighting conditions, type of burning material, type of camera and distance of camera and camera settings, there can be several different textures of fire.

A dataset of fire and non-fire images is constructed for training and testing as in Chapter 2. Images in the dataset can be categorized under the index terms “flame”, “fire”, “wildfires”, “fireplace”, “burning”, “sunset”, “autumn”, “flower

garden”, “indoor”, “outdoor”, “lake” and “animals”. It is randomly divided into two parts: train and test sections. The test part of this dataset and the train part of the dataset used in Chapter 2 are independent.

3.1 Fire Detection with Patches

Images that contain fire in the train part are cropped manually such that only fire regions exists. Then 32 by 32 non-overlapping patches are extracted from these cropped images. Since cropped images can be in different sizes, generally different number of patches are extracted from different cropped images. If the number of extracted patches from an image is greater than a limit, L_{pos} , then L_{pos} patches are randomly selected. If random sampling is not applied, the biggest cropped image can dominate training set and the smallest cropped image may be neglected by the classifier. 32 by 32 non-overlapping negative training patches are also randomly selected with a limit, L_{neg} , when number of patches extracted from a negative image is bigger than L_{neg} .

After training patches were constructed, features are extracted and used to form a training set. A support vector machine (SVM) [9] is chosen for classification. A SVM classifier is trained for each feature set. Each feature set extracted from patches of a test image is tested by the SVM classifier, which is trained by the same type of feature set. This training process and the test process are repeated with different features. For texture classification problem, variations of covariance descriptors, codifference descriptors, Law’s Masks, Gabor filters and Markov chains are used as features.

3.2 Covariance Descriptors

Tuzel, Porikli and Meer suggested covariance descriptors [10] and applied this method to object detection and texture classification problems. Assuming that properties of pixels of an image have multivariate normal distribution they used lower triangular part of estimated of covariance matrix as the feature vector. Half of the covariance matrix is used because it is symmetric. The covariance matrix can be estimated as follows:

$$\hat{\Sigma} = \frac{1}{N-1} \sum_i \sum_j (\Phi_{i,j} - \bar{\Phi})(\Phi_{i,j} - \bar{\Phi})^T \quad (3.1)$$

$$\bar{\Phi} = \frac{1}{N} \sum_i \sum_j \Phi_{i,j} \quad (3.2)$$

where N is the number of pixels.

$\Phi_{i,j}$ is a property vector of the pixel at location (i, j) ($P_{i,j}$). Let us define $Red(i, j)$, $Green(i, j)$, $Blue(i, j)$, and $Intensity(i, j)$ as red, green, blue and intensity values of the pixel $P_{i,j}$. Respectively $\Phi_{i,j}$ can include locations of pixels and intensity values of pixels, $\Phi_{i,j} = \begin{bmatrix} X(i, j) & Y(i, j) & Intensity(i, j) \end{bmatrix}^T$, or it can only include red, green, blue values of pixels, $\Phi_{i,j} = \begin{bmatrix} Red(i, j) & Green(i, j) & Blue(i, j) \end{bmatrix}^T$. Several pixel properties are defined from Equation (3.3) to Equation (3.19). By using combinations of these properties, different covariance descriptors are defined and tested. A list of defined covariance descriptors and the property set of the corresponding descriptor can be found in Table 3.1.

$$Y_{i,j} = j, \quad (3.3)$$

$$X_{i,j} = i, \quad (3.4)$$

$$R_{i,j} = Red(i, j), \quad (3.5)$$

$$Rx_{i,j} = \left| \frac{\partial Red(i, j)}{\partial i} \right|, \quad (3.6)$$

$$Ry_{i,j} = \left| \frac{\partial Red(i, j)}{\partial j} \right|, \quad (3.7)$$

$$Rxx_{i,j} = \left| \frac{\partial^2 Red(i, j)}{\partial i^2} \right|, \quad (3.8)$$

$$Ryy_{i,j} = \left| \frac{\partial^2 Red(i, j)}{\partial j^2} \right|, \quad (3.9)$$

$$G_{i,j} = Green(i, j), \quad (3.10)$$

$$B_{i,j} = Blue(i, j), \quad (3.11)$$

$$I_{i,j} = Intensity(i, j), \quad (3.12)$$

$$Ix_{i,j} = \left| \frac{\partial Intensity(i, j)}{\partial i} \right|, \quad (3.13)$$

$$Iy_{i,j} = \left| \frac{\partial Intensity(i, j)}{\partial j} \right|, \quad (3.14)$$

$$Ixx_{i,j} = \left| \frac{\partial^2 Intensity(i, j)}{\partial i^2} \right|, \quad (3.15)$$

$$Iyy_{i,j} = \left| \frac{\partial^2 Intensity(i, j)}{\partial j^2} \right|, \quad (3.16)$$

$$Ha_{i,j} = \begin{cases} 1 & \text{if } Red(i, j) + 5 > Green(i, j) \text{ and } Green(i, j) \geq Blue(i, j) \\ 0 & \text{otherwise} \end{cases} \quad (3.17)$$

$$Hb_{i,j} = \begin{cases} 1 & \text{if } Red(i,j) > 110 \\ 0 & \text{otherwise} \end{cases} \quad (3.18)$$

and

$$Fp_{i,j} = \begin{cases} 1 & \text{if } Nnet(P) > 0.5 \\ 0 & \text{otherwise} \end{cases} \quad (3.19)$$

Note that, $Nnet(P)$ is the output of neural network model defined in section 2.4. First derivative of the image is taken by filtering the image with $[-1 \ 0 \ 1]$ and second derivative is found by filtering the image with $[1 \ -2 \ 1]$.

Table 3.1: List of defined covariance descriptors and which properties are used.

Name	Y	X	R	Rx	Ry	Rxx	Ryy	G	B	I	Ix	Iy	Ixx	Iyy	Ha	Hb	Fp
cov_R5	✓	✓	✓	✓	✓												
cov_R7	✓	✓	✓	✓	✓	✓	✓										
cov_I5	✓	✓								✓	✓	✓					
cov_I7	✓	✓								✓	✓	✓	✓	✓			
cov_RGB8	✓	✓	✓					✓	✓	✓	✓	✓					
cov_RGB9	✓	✓	✓					✓	✓	✓	✓	✓					✓
cov_RGB10	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓			
cov_RGB10_c	✓	✓	✓					✓	✓	✓	✓	✓			✓	✓	
cov_RGB11	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓			✓
cov_RGB12	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	

Lower triangular parts of covariance matrices of property sets of covariance descriptors form the feature vectors except variance of X , variance of Y and covariance of X with Y . Since these values are constant for a 32 by 32 image, they are not used in feature vector. In Figure 3.1, a test image that contains fire and in Figure 3.2 a negative test image without fire can be found.

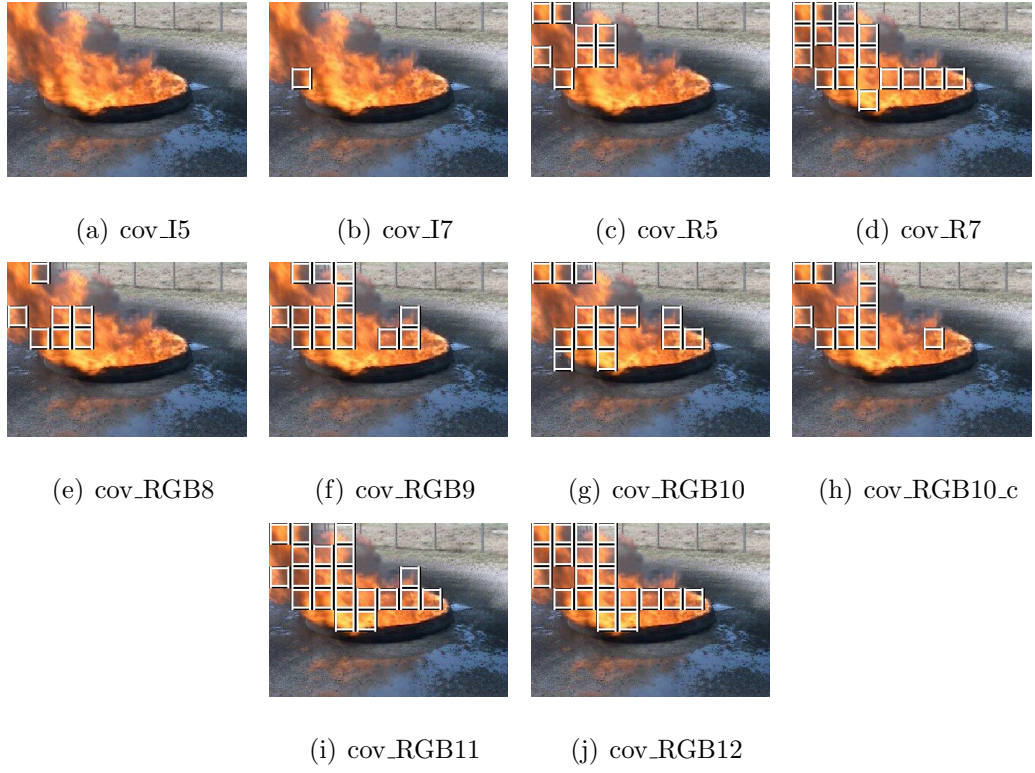


Figure 3.1: Classification of patches of a test image when various covariance descriptors are used.

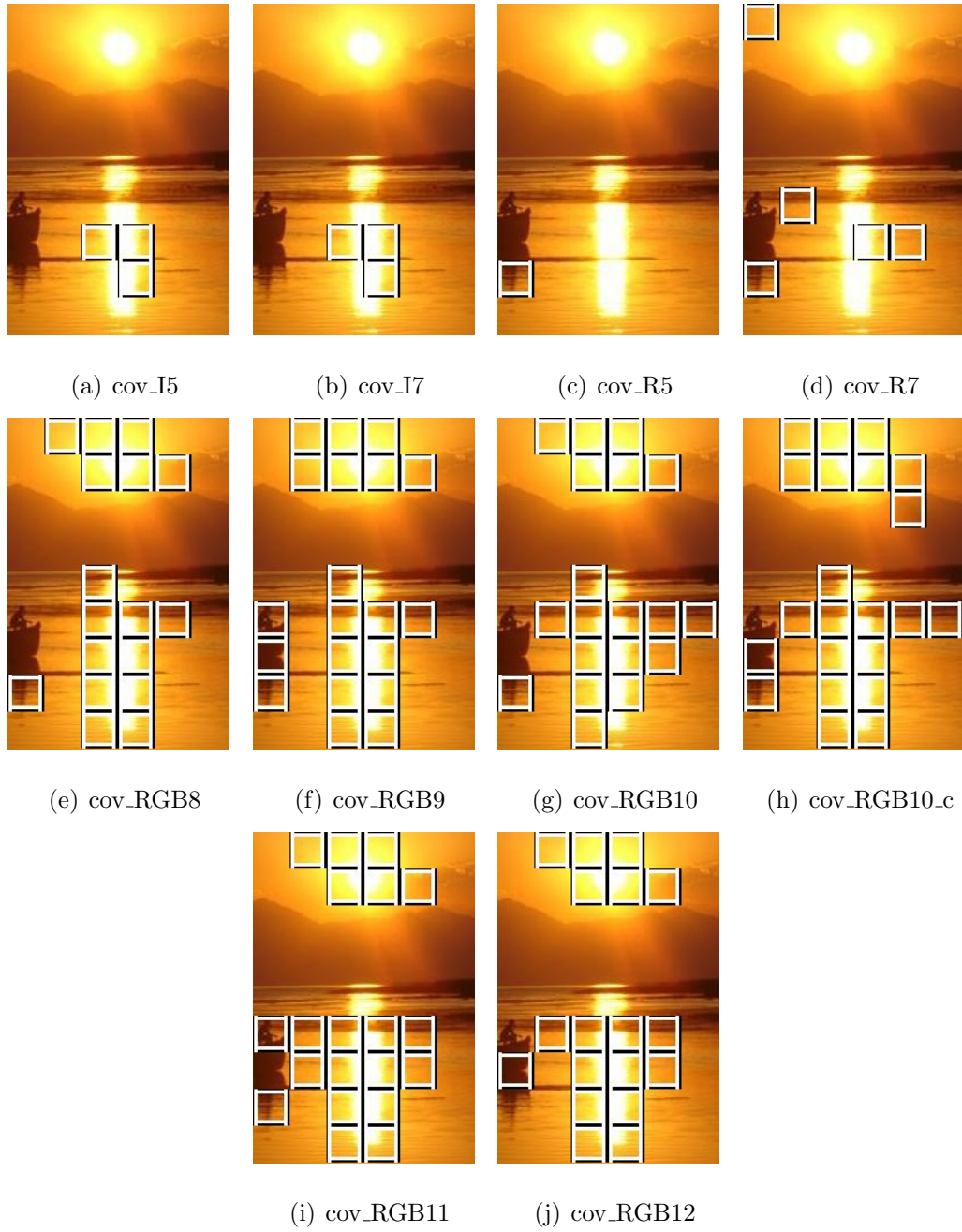


Figure 3.2: Classification of patches of a test image when various covariance descriptors are used.

3.3 Codifference Descriptors

Codifference descriptors, proposed by Tuna, Onaran and Cetin in [11], are defined in order to reduce the complexity of the calculation of the covariance matrix. It is suitable for processors which do not have multiplication command. Although today's computers have multiplication command, sometimes codifference descriptors perform better than covariance descriptors. Definition of a codifference descriptor is same as covariance descriptor except instead of using scalar multiplication operator a new operator, $\oplus$, is used. This operator is defined as follows:

$$a \oplus b = \text{sign}(ab) (|a| + |b|) \quad (3.20)$$

where $\text{sign}(0)$ is defined as 1. Or it can be defined as Equation (3.21).

$$a \oplus b = \begin{cases} a + b & \text{if } a > 0 \text{ and } b > 0 \\ a - b & \text{if } a < 0 \text{ and } b > 0 \\ -a + b & \text{if } a > 0 \text{ and } b < 0 \\ -a - b & \text{if } a < 0 \text{ and } b < 0 \\ |a| + |b| & \text{if } a = 0 \text{ or } b = 0 \end{cases} \quad (3.21)$$

Codifference matrix is defined as follows:

$$\hat{S} = \frac{1}{1023} \sum_{i=1}^{32} \sum_{j=1}^{32} (\Phi_{i,j} - \bar{\Phi}) \odot (\Phi_{i,j} - \bar{\Phi})^T \quad (3.22)$$

where $\odot$ is the matrix multiplication operator that uses $\oplus$ operator instead of scalar multiplication.

Similar to the covariance descriptors several codifference descriptors, that have different property sets, are defined. Similar descriptors use the same Φ , i.e. both

cov_R5 and cod_R5 use Y , X , R , Rx and Ry . Therefore Table 3.1 also defines property sets of covariance descriptors.

Similar to the covariance descriptors only lower triangular part of $\hat{S}$ is used because $\hat{S}$ is also symmetric. Since codifference of the position X with itself, codifference of the position Y with itself and codifference of X with Y are constant for a 32 by 32 image, they are not used in the feature vector. In Figure 3.3 an image containing fire and in Figure 3.4 negative example results are presented.

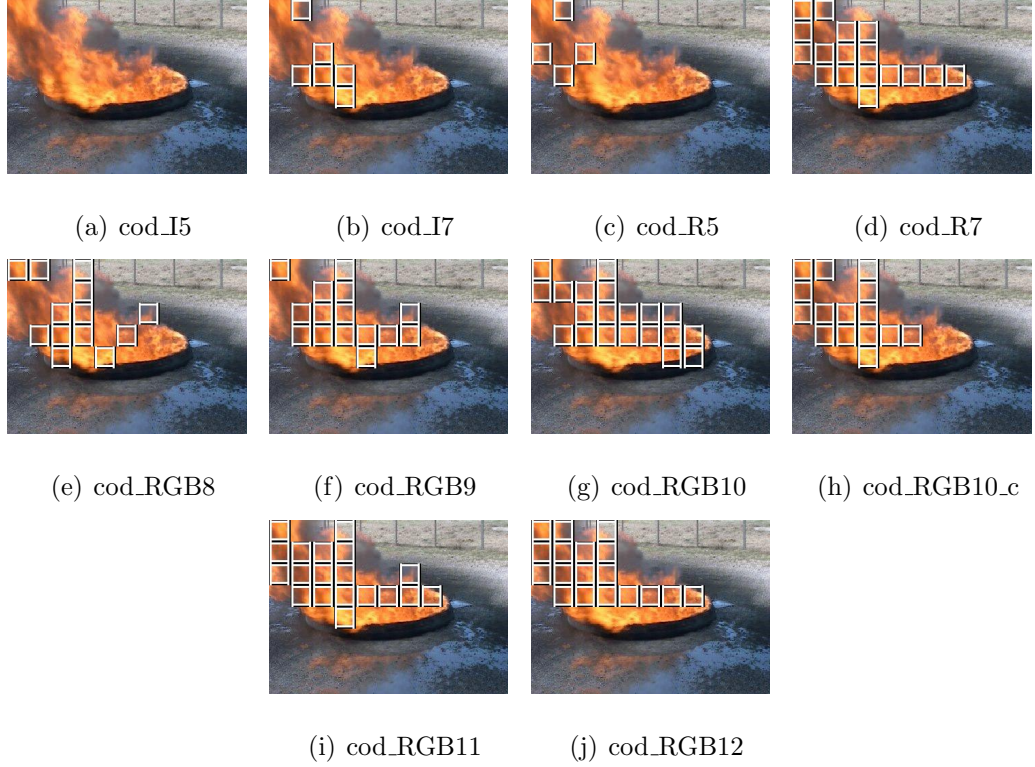


Figure 3.3: Classification of patches of a test image when various codifference descriptors are used.

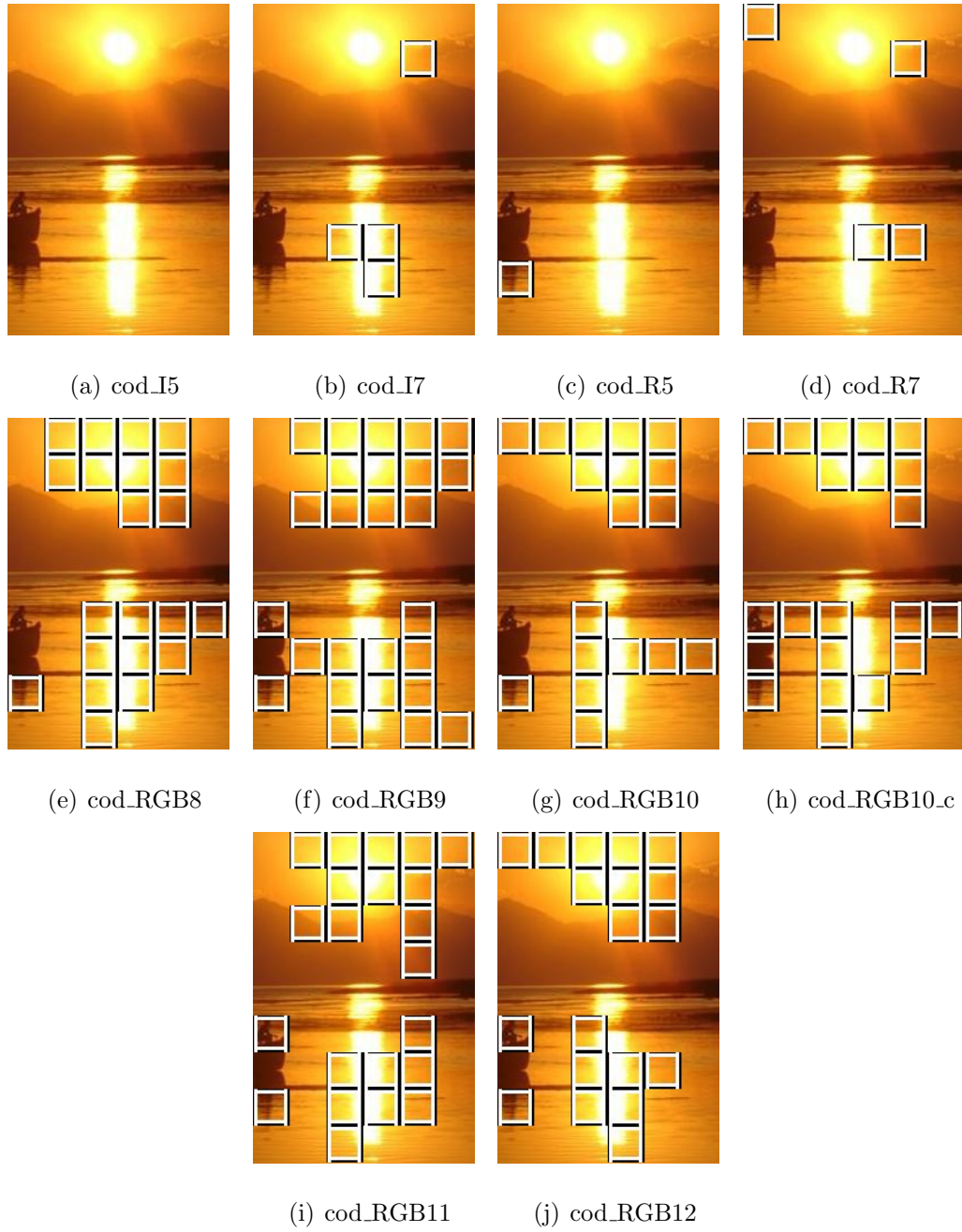


Figure 3.4: Classification of patches of a test image when various codifference descriptors are used.

3.4 Laws' Masks

In [12] Laws suggested using 25 filters for texture classification problem. All filters are separable and they can be build from five 1-D filters. These 1-D filters are:

$$\begin{aligned}L_{h1} &= \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \end{bmatrix}^T \\L_{h2} &= \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}^T \\L_{h3} &= \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \end{bmatrix}^T \\L_{h4} &= \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \end{bmatrix}^T \\L_{h5} &= \begin{bmatrix} -1 & 2 & 0 & -2 & -1 \end{bmatrix}^T\end{aligned}$$

Assuming that Z is a matrix of image property (intensity, red channel, ...) $Lf_{a,b}(Z)$ is defined as the filtered version of Z with L_{ha} and L_{hb}^T . As shown in equation (3.23), mean of the absolute values of filtered images are used as features.

Codifference and covariance descriptors using Laws' masks can also be constructed.

$$L\phi(Z) = \begin{bmatrix} \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} |Lf_{1,1}(Z; i, j)| \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} |Lf_{1,2}(Z; i, j)| \\ \cdot \\ \cdot \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} |Lf_{1,5}(Z; i, j)| \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} |Lf_{2,1}(Z; i, j)| \\ \cdot \\ \cdot \\ \cdot \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} |Lf_{5,5}(Z; i, j)| \end{bmatrix} \quad (3.23)$$

Note that $Z(i, j)$ is the (i'th, j'th) element of matrix Z and $Lf_{a,b}(Z; i, j)$ is the (i'th, j'th) element of the filtered image. $L\phi$ has 25 elements. Using $L\phi$ notation 3 feature types are defined:

$$\begin{aligned} \text{laws_I} &= L\phi(Intensity) \\ \text{laws_R} &= L\phi(Red) \\ \text{laws_RGB} &= \begin{bmatrix} L\phi(Red) \\ L\phi(Green) \\ L\phi(Blue) \end{bmatrix} \end{aligned}$$

Figure 3.5 and Figure 3.6 contain images whose patches are classified according to these features.

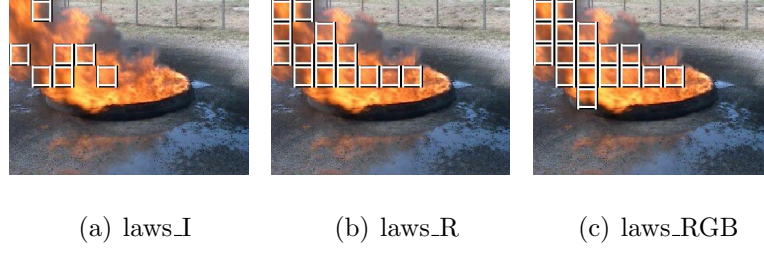


Figure 3.5: Classification of patches of a test image when Law's masks are used.

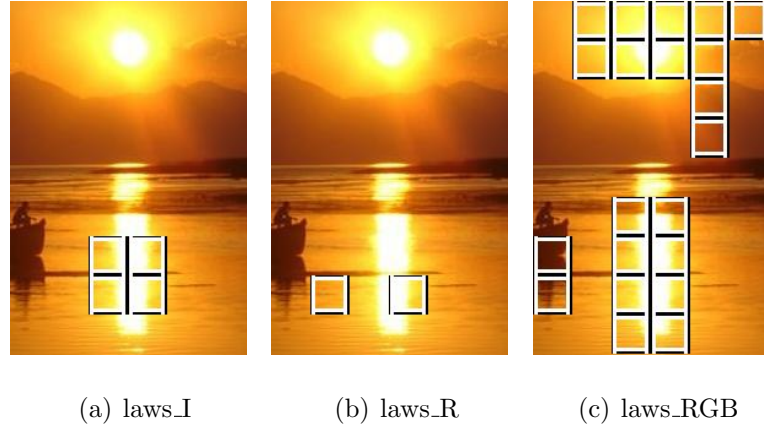


Figure 3.6: Classification of patches of a test image when Law's masks are used.

3.5 Gabor Filters

In [13] Jain and Farrokhnia suggested using Gabor filter banks for texture segmentation. Kernel of Gabor filters, used in this work, are constructed by using Equation (3.24). There are 3 parameters that can be changed. As it is seen from Equation (3.24) impulse response of a Gabor filter has infinite length. But as we get away from origin, it get closer to zero. Therefore using a finite part of it (generally a rectangle whose center is at the origin) is enough. Area of this finite part is related to the value of σ . In our problem every patch has a size of 32 by 32. Therefore setting σ to a big value is not a good idea. We used $\sigma = 2$,

for all Gabor filters. θ is the filter orientation and f_0 is the discrete radial center frequency. In this part, four orientations (0° , 45° , 90° and 135°) and five discrete radial center frequencies $\left(\frac{\sqrt{2}}{2^6}, \frac{\sqrt{2}}{2^4}, \frac{\sqrt{2}}{2^4}, \frac{\sqrt{2}}{2^3}, \frac{\sqrt{2}}{2^2}\right)$ are used.

$$G_h(\theta, f_0, \sigma; i, j) = \exp\left(-\frac{(i\cos\theta + j\sin\theta)^2 + (-i\sin\theta + j\cos\theta)^2}{2\sigma^2}\right) \cos(2\pi f_0 i) \quad (3.24)$$

Let us define $G_f(Z, \theta, f_0; i, j)$ as filtered version of $Z(i, j)$ with $G_h(\theta, f_0, 2; i, j)$. Means of the absolute values of these filtered images form features as shown in Equation (3.25).

$$G\phi(Z) = \begin{bmatrix} \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} \left| Gf(Z, 0, \frac{\sqrt{2}}{2^6}; i, j) \right| \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} \left| Gf(Z, 0, \frac{\sqrt{2}}{2^5}; i, j) \right| \\ \cdot \\ \cdot \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} \left| Gf(Z, 0, \frac{\sqrt{2}}{2^2}; i, j) \right| \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} \left| Gf(Z, 45^\circ, \frac{\sqrt{2}}{2^6}; i, j) \right| \\ \cdot \\ \cdot \\ \cdot \\ \frac{1}{1024} \sum_{i=1}^{32} \sum_{j=1}^{32} \left| Gf(Z, 135^\circ, \frac{\sqrt{2}}{2^2}; i, j) \right| \end{bmatrix} \quad (3.25)$$

$G\phi$ has 20 elements. Using $G\phi$ notation 3 feature types are defined:

$$\begin{aligned}
\text{gabor_I} &= G\phi(\text{Intensity}) \\
\text{gabor_R} &= G\phi(\text{Red}) \\
\text{gabor_RGB} &= \begin{bmatrix} G\phi(\text{Red}) \\ G\phi(\text{Green}) \\ G\phi(\text{Blue}) \end{bmatrix}
\end{aligned}$$

Some of the classification results using these features can be found in Figure 3.7 and in Figure 3.8.

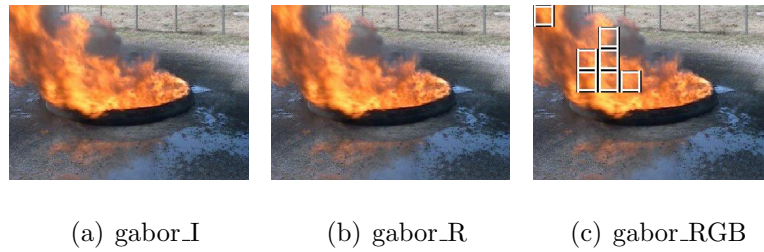


Figure 3.7: Classification of patches of a test image when Gabor filters are used.

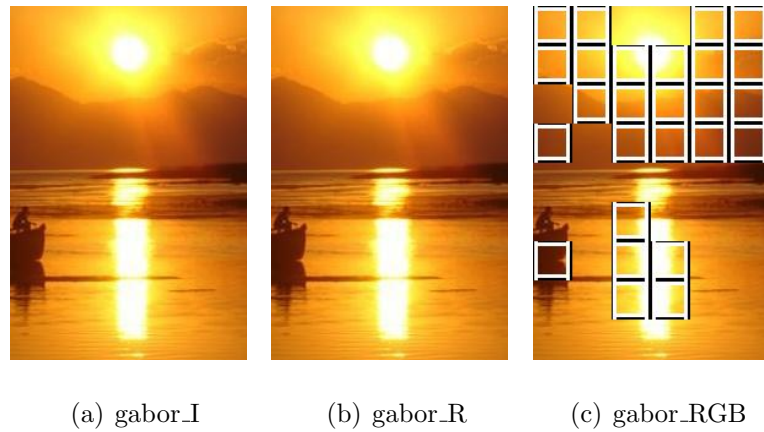


Figure 3.8: Classification of patches of a test image when Gabor filters are used.

3.6 Markov Chain

A discrete time Markov chain is a stochastic process with the property that probability mass function of the next state depends only the current state. The estimated elements of the transition matrix $\left(\hat{T}\right)$ of this process are used as features.

Assuming $C(i, j)$ refers the number of transitions from $state_i$ to $state_j$ occurs in rows of the state matrix. Definition of $\hat{T}$ can be estimated as follows:

$$\hat{T} = \begin{cases} \frac{C(i,j)}{\sum_{k=0} C(i,k)} & \text{if } \sum_{k=0} C(i,k) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.26)$$

MC_Ia is a Markov chain which has 'a' states. $state_0$ is for pixels that are not possible fire-pixels according to chromatic model, which is defined in Section (2.1), when R_T is 90. $state_1$ to $state_{a-1}$ are like bins of a histogram. According to their intensities every possible fire-pixel assigns to a state. But different from histogram bins, definitions of these states are adaptive to the intensity of test patch. Definitions of $state_1$ to $state_{a-1}$ change such that they always divide the range $\min(Intensity)$ to $\max(Intensity)$ equally.

MC_IWL is similar to MC_I4. This time *Intensity* is used after filtering with $[1 \ -2 \ 1]$. This process has constant *state* definitions. $state_1$ to $state_3$ are for possible fire-pixels whose filtered intensities are in $(-\infty.. -5], [-5..5)$ and $[5..\infty)$.

MC_Ra and MC_RWL are same as *MC_Ia* and MC_IWL, but these processes use *Red* instead of *Intensity*. MC_nrmIa, MC_nrmIWL, MC_nrmRa and MC_nrmRWL are estimations of transition matrices of MC_Ia, MC_IWL, MC_Ra, MC_RWL. Sample classification results are presented in Figure 3.9 and in Figure 3.10.

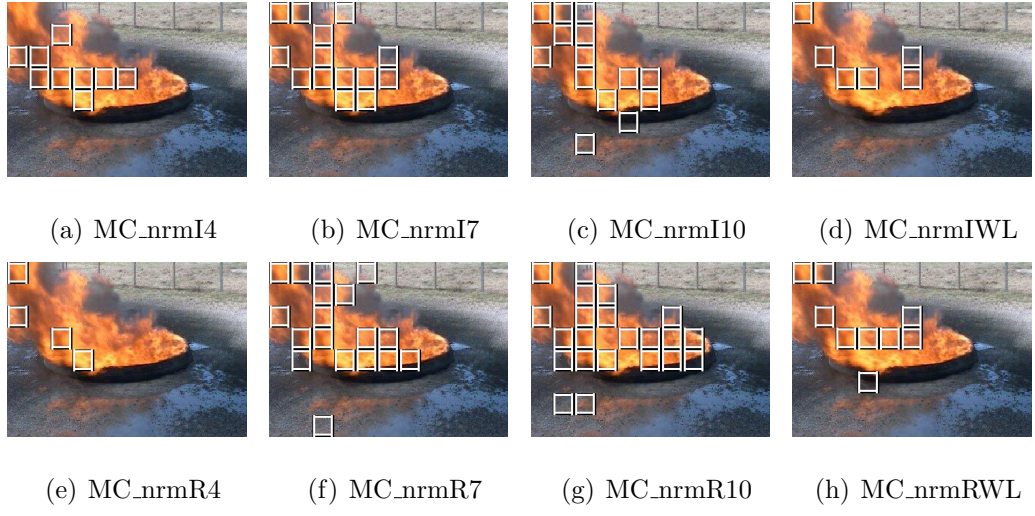


Figure 3.9: Classification of patches of a test image when Markov chains are used.

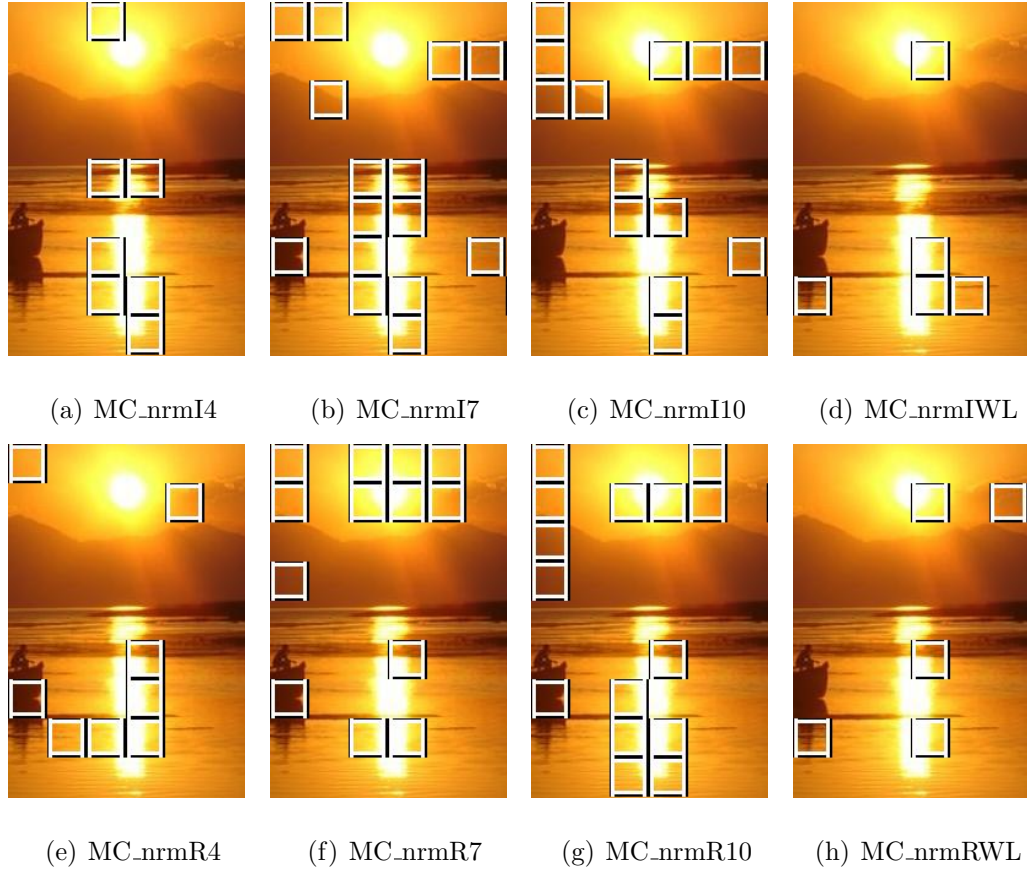


Figure 3.10: Classification of patches of a test image when Markov chains are used.

3.7 Results and Summary

After 32 by 32 patches of an image are classified using one of the feature sets that are described previously, test image is classified by using the decisions of patches. The simplest way of doing this is to check if there is any patch classified as fire. But applying **OR** operator to decisions of patches is not a good way. For a negative test image if any of the patches is miss-classified then image will be miss-classified. Therefore using **OR** operator can cause very high false-alarm

rates.

Let's assume decisions of patches of a test image are like in Figure 3.11. Small blocks mean there is a fire classified patch. Final classification is done by searching 3 by 3 patterns in decisions of a test image. A test block, which consists of 3 by 3 decisions, is shift over all the decisions. If its center decision is fire and sum of the decisions that are fire is greater or equal to S_T in a test block then that image classified as fire. In Figure 3.12 there are positive and negative test blocks for $S_T = 3$. Result of Figure 3.12(f) is negative because its center patch is not classified as fire. Classification results for $S_T = 1$ to $S_T = 4$ are in Table 3.2 to Table 3.5 where true detection is defined as:

$$\frac{\text{the number of correctly classified images that do not contain fire}}{\text{number of images that do not contain fire in test set}} \quad (3.27)$$

and false alarm is defined as:

$$\frac{\text{the number of miss classified images that do not contain fire}}{\text{number of images which do not contain fire in test set}} \quad (3.28)$$

$S_T = 1$ case is same as applying **OR** operator.

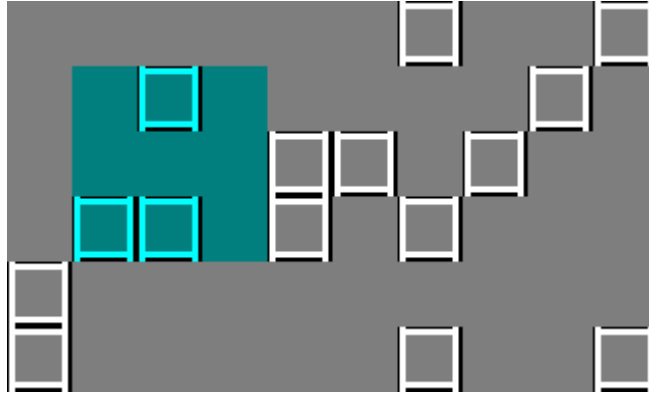


Figure 3.11: An example for decisions of patches of an image.

In conclusion, cov_RGB12 and cod_RGB12 generally give better performance than the other feature sets. The classification rate also depends on the threshold

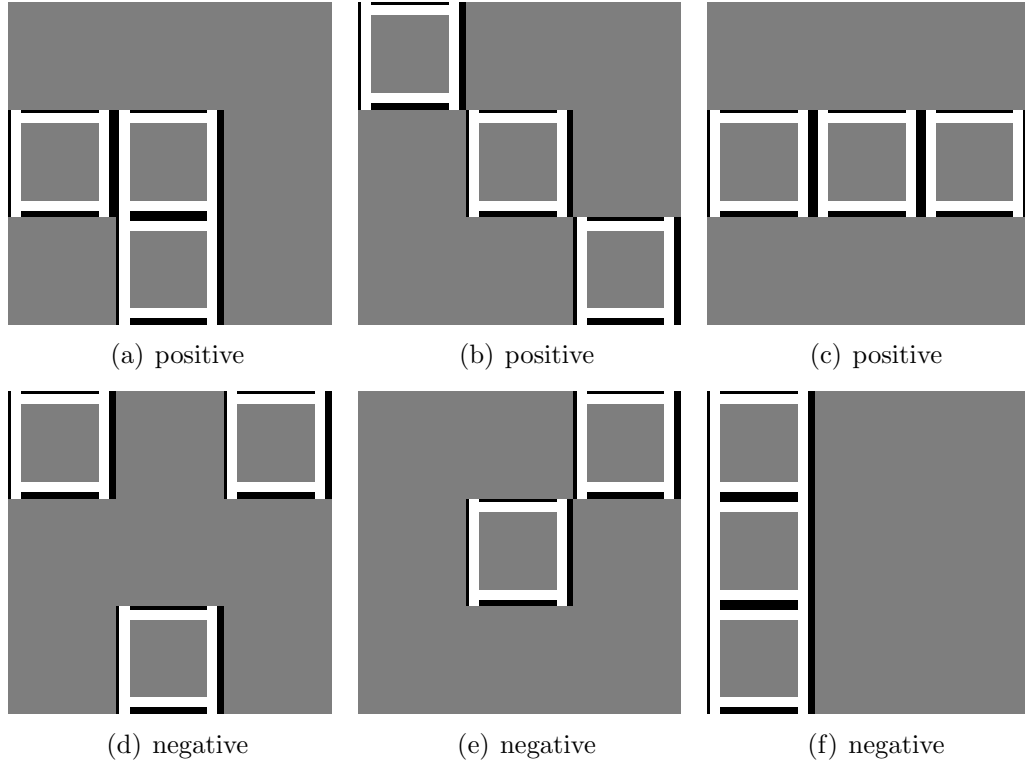


Figure 3.12: Positive and negative test blocks for $S_T = 3$

S_T . As S_T increases both the false alarm rate and the true detection rate decrease. Optimum accuracy can be achieved by changing S_T .

chapter FIRE DETECTION IN VIDEO

In this chapter, a real-time fire detection algorithm is described and tested with videos. We use our expertise that we gather from the previous chapters in the final design. As seen from the results of previous chapters, chromatic color model is simple and it has a high detection rate. Also we observe that covariance descriptors are good features for flame detection in images.

In the rest of the chapter, the idea of covariance region descriptors, which is used in image processing, is extended to video. Similar to the image analysis case, video is divided into $16 \times 16 \times F_{rate}$ spatiotemporal blocks. The height and

the width of the block is 16x16 and F_{rate} is the temporal duration of the block.

3.8 Covariance Descriptors For Video

As explained in Chapter 3.2, assuming a property set (properties of pixels) has a wide sense stationary multivariate normal distribution, covariance descriptors are the lower triangular part of the estimated covariance matrix either. This is a reasonable assumption for a flame colored image region because such regions do not contain strong edges. Therefore, covariance descriptors can be used to model spatial characteristics of fire regions in images. To model the temporal flicker in fire flames we introduce temporally extended covariance descriptors cov_CST. Temporally extended covariance descriptors are designed to describe spatiotemporal blocks.

Assuming $Intensity(i, j, n)$ is the intensity of (i^{th}, j^{th}) pixel of the n^{th} frame of a test block and $Red, Green, Blue$ holds colors of pixels of frames of the test block, the properties defined from equation (3.30) to (3.38) will be used. During the implementation of the covariance method, the first derivative of the image is computed by filtering the image with $[-1 \ 0 \ 1]$ and second derivative is found by filtering the image with $[1 \ -2 \ 1]$ filters, respectively.

$$R_{i,j,n} = Red(i, j, n), \quad (3.29)$$

$$G_{i,j,n} = Green(i, j, n), \quad (3.30)$$

$$B_{i,j,n} = Blue(i, j, n), \quad (3.31)$$

$$I_{i,j,n} = Intensity(i, j, n), \quad (3.32)$$

$$Ix_{i,j,n} = \left| \frac{\partial Intensity(i, j, n)}{\partial i} \right|, \quad (3.33)$$

$$Iy_{i,j,n} = \left| \frac{\partial Intensity(i, j, n)}{\partial j} \right|, \quad (3.34)$$

$$Ixx_{i,j,n} = \left| \frac{\partial^2 Intensity(i, j, n)}{\partial i^2} \right|, \quad (3.35)$$

$$Iyy_{i,j,n} = \left| \frac{\partial^2 Intensity(i, j, n)}{\partial j^2} \right|, \quad (3.36)$$

$$It_{i,j,n} = \left| \frac{\partial Intensity(i, j, n)}{\partial n} \right|, \quad (3.37)$$

$$Itt_{i,j,n} = \left| \frac{\partial^2 Intensity(i, j, n)}{\partial n^2} \right| \quad (3.38)$$

Every input frame is scaled to 320 by 240 in order to run the fire detection algorithm in real-time. 16 is chosen as both width and height of the spatiotemporal block ($240/16 = 15$). Therefore fire may not fill most of the blocks but it can disperse over a few blocks. In order to reduce the effect of non-fire regions, instead of using properties of every pixel only properties of chromatically possible fire-pixels are used in the estimation of covariance matrix. For color analysis chromatic model 2C is used with $R_T = 110$. Only the collection of pixels corresponding to the non-zero values of the following color mask are used in the

extraction of features. The color mask is defined by the following function:

$$\Psi(i, j, n) = \begin{cases} \text{true} & \text{if } Red(i, j) > Green(i, j, n) \text{ and } Green(i, j, n) \geq Blue(i, j, n) \\ & \text{and } Red(i, j, n) > 110 \\ \text{false} & \text{otherwise} \end{cases} \quad (3.39)$$

This is not a tight condition, almost all flame colored reddish regions satisfy Equation (3.39).

A total of 10 property parameters are used for each pixel satisfying the color condition. However this requires $\frac{10 \times 11}{2} = 55$ covariance computations. To reduce the computational cost we compute the covariances of the pixel property vectors

$$\Phi_{color}(i, j, n) = \begin{bmatrix} Red(i, j, n) & Green(i, j, n) & Blue(i, j, n) \end{bmatrix}^T \quad (3.40)$$

and

$$\Phi_{spatioTemporal}(i, j, n) = \begin{bmatrix} I(i, j, n) \\ Ix(i, j, n) \\ Iy(i, j, n) \\ Ixx(i, j, n) \\ Iyy(i, j, n) \\ It(i, j, n) \\ Itt(i, j, n) \end{bmatrix} \quad (3.41)$$

separately. Therefore the property vector Φ_{color} produces $\frac{3 \times 4}{2} = 6$ and the property vector $\Phi_{spatioTemporal}$ produces $\frac{7 \times 8}{2} = 28$ covariance values respectively. In video processing the feature vector `cov_CST` is the augmented vector of these covariance descriptors.

3.9 Training and Testing

For training and testing, 16 by 16 by F_{rate} blocks are extracted from various video clips. The temporal dimension of the blocks are determined by the frame rate parameter F_{rate} which is between 10 and 25 in our train and test videos. These blocks do not overlap in spatial domain but there are 50% overlap in time domain. This means that classification is not repeated after every image frame of the video. An SVM classifier is used as the recognition engine.

The classification is done periodically with the period $F_{rate}/2$. This decreases the cost of classification. On the other hand estimating covariance matrix periodically with the formula given in Equation (3.1) requires accumulating your data at the end of each period. This is because the covariance matrix estimation formula (3.1) needs to know the mean of the data and the mean can not be calculated without knowing the entire data. But luckily there is another covariance matrix estimation formula (3.42), that can be started to calculate without waiting for the entire data. But this formula has a certain drawback. It aggregates products of numbers. In our problem these numbers can be -510 for the second derivative intensities. In a 32 bit environment to prevent overflowing we can take summation of maximum $\text{floor}\left(\frac{2^{32}}{510*510}\right) = 16512$ elements. This means that using 32 by 32 by 20 blocks can cause erroneous calculation of $\hat{\Sigma}$.

$$\hat{\Sigma}(a, b) = \frac{1}{N-1} \left(\sum_i \sum_j \Phi_{i,j}(a) \Phi_{i,j}(b) - \frac{1}{N} \left(\sum_i \sum_j \Phi_{i,j}(a) \right) \left(\sum_i \sum_j \Phi_{i,j}(b) \right) \right) \quad (3.42)$$

After spatiotemporal blocks are extracted, the number of chromatically fire-pixels, $\sum_i \sum_j \sum_n \Psi(i, j, n)$, is found. If this number is lower than or equal to $\frac{3}{5}$ of

the number of the elements of block ($16 \times 16 \times F_{rate}$) then that block is classified as a non-fire block. This thresholding is done because only chromatically fire-pixels according to chromatic model 2C is used and the covariance matrix does not give any information about this number because it is normalized. If the number of possible fire-pixels is enough, then classification is done by the SVM classifier with using cov_CST of the corresponding block.

In this thesis 7 positive and 10 negative videos are used for training. For positive videos only parts of the videos that contain fire are used. Before adding cov_CST of blocks to the training set, the number of chromatically fire-pixels of that block is checked like mentioned above for both positive and negative samples.

At the final step of our flame detection a confidence value is determined according to the number of positively classified video blocks and their positions. After every block is classified spatial neighborhoods of the block are used to decide alarm confidence similar to Section 3.7. If there is no neighbor block classified as fire, confidence level is set to 1. If there is a single neighbor block, which is classified as fire, then confidence level is set to 2. If there are more than 2 blocks neighbor blocks classified as fire then confidence level of that block is set to 3 which is the highest level of confidence that the algorithm provides. In Figure 3.13 there are sample frames after classification.

3.10 Results and Summary

Each video in the test set is cropped manually. If the video is in the positive set, all frames of the video include fire otherwise none of the frames include fire regions. Similar to Section 3.7 a threshold C_T is used in the decision process. If



Figure 3.13: Sample image frames from videos.

the confidence level of any block of the frame is greater than or equal to C_T that frame is marked as a fire containing frame. Results are summarized in Table 3.6 and Table 3.7 where T_X is defined as:

$$\frac{\text{the number of correctly classified frames, which contain fire, in test video}}{\text{number of frames which contain fire in test video}} \quad (3.43)$$

and F_X is defined as:

$$\frac{\text{the number of miss classified frames, which do not contain fire, in test video}}{\text{number of frames which do not contain fire in test video}} \quad (3.44)$$

when $C_T =_X$. First 13 videos do contain fire and the remaining 9 videos do not contain fire and flames but contain flame colored regions.

Table 3.6: True detection rates of real-time fire detection system.

Video name	T_1	T_2	T_3
posVideo1	281/ 293 (95.9%)	266/ 293 (90.8%)	161/ 293 (54.9%)
posVideo2	493/ 510 (96.7%)	463/ 510 (90.8%)	413/ 510 (81.0%)
posVideo3	364/ 381 (95.5%)	345/ 381 (90.6%)	310/ 381 (81.4%)
posVideo4	1643/ 1655 (99.3%)	1643/ 1655 (99.3%)	1643/ 1655 (99.3%)
posVideo5	2394/ 2406 (99.5%)	2394/ 2406 (99.5%)	2394/ 2406 (99.5%)
posVideo6	225/ 258 (87.2%)	110/ 258 (42.6%)	35/ 258 (13.6%)
posVideo7	535/ 547 (97.8%)	530/ 547 (96.9%)	495/ 547 (90.5%)
posVideo8	501/ 513 (97.7%)	501/ 513 (97.7%)	501/ 513 (97.7%)
posVideo9	651/ 663 (98.2%)	651/ 663 (98.2%)	651/ 663 (98.2%)
posVideo10	223/ 235 (94.9%)	223/ 235 (94.9%)	223/ 235 (94.9%)
posVideo11	166/ 178 (93.3%)	126/ 178 (70.8%)	35/ 178 (19.7%)
posVideo12	234/ 246 (95.1%)	234/ 246 (95.1%)	234/ 246 (95.1%)
posVideo13	196/ 208 (94.2%)	196/ 208 (94.2%)	196/ 208 (94.2%)

Table 3.7: False alarm rates of real-time fire detection system.

Video name	F_1	F_2	F_3
negVideo1	940/ 4539 (20.7%)	315/ 4539 (6.9%)	160/ 4539 (3.5%)
negVideo2	25/ 155 (16.1%)	5/ 155 (3.2%)	0/ 155 (0.0%)
negVideo3	0/ 160 (0.0%)	0/ 160 (0.0%)	0/ 160 (0.0%)
negVideo4	945/ 1931 (48.9%)	465/ 1931 (24.1%)	140/ 1931 (7.3%)
negVideo5	107/ 439 (24.4%)	45/ 439 (10.3%)	10/ 439 (2.3%)
negVideo6	520/ 1142 (45.5%)	305/ 1142 (26.7%)	185/ 1142 (16.2%)
negVideo7	85/ 541 (15.7%)	0/ 541 (0.0%)	0/ 541 (0.0%)
negVideo8	20/ 3761 (0.5%)	5/ 3761 (0.1%)	0/ 3761 (0.0%)
negVideo9	30/ 645 (4.7%)	5/ 645 (0.8%)	5/ 645 (0.8%)

The classification rates of the method depend on the threshold C_T . Similar to Section 3.7 increasing C_T decreases detection rates. According to the results, using $C_T = 3$ provides a good performance. Although the true detection rate is low when $C_T = 3$, we do not need to detect all fire frames correctly to issue an alarm. It is enough to detect fire in a short time without too many false alarms.

Table 3.2: Classification results of texture classifiers with patches when $S_T = 1$.

Method Name	True Detection	False Alarm
cov_R5	119/ 140 (85.0%)	73/ 141 (51.8%)
cov_R7	133/ 140 (95.0%)	92/ 141 (65.2%)
cov_I5	82/ 140 (58.6%)	59/ 141 (41.8%)
cov_I7	100/ 140 (71.4%)	77/ 141 (54.6%)
cov_RGB8	129/ 140 (92.1%)	74/ 141 (52.5%)
cov_RGB9	134/ 140 (95.7%)	74/ 141 (52.5%)
cov_RGB10_c	136/ 140 (97.1%)	83/ 141 (58.9%)
cov_RGB10	136/ 140 (97.1%)	78/ 141 (55.3%)
cov_RGB11	135/ 140 (96.4%)	78/ 141 (55.3%)
cov_RGB12	137/ 140 (97.9%)	83/ 141 (58.9%)
cod_R5	107/ 140 (76.4%)	71/ 141 (50.4%)
cod_R7	132/ 140 (94.3%)	96/ 141 (68.1%)
cod_I5	75/ 140 (53.6%)	52/ 141 (36.9%)
cod_I7	111/ 140 (79.3%)	90/ 141 (63.8%)
cod_RGB8	133/ 140 (95.0%)	93/ 141 (66.0%)
cod_RGB9	138/ 140 (98.6%)	91/ 141 (64.5%)
cod_RGB10_c	136/ 140 (97.1%)	89/ 141 (63.1%)
cod_RGB10	136/ 140 (97.1%)	86/ 141 (61.0%)
cod_RGB11	139/ 140 (99.3%)	86/ 141 (61.0%)
cod_RGB12	139/ 140 (99.3%)	89/ 141 (63.1%)
laws_I	118/ 140 (84.3%)	75/ 141 (53.2%)
laws_R	124/ 140 (88.6%)	80/ 141 (56.7%)
laws_RGB	126/ 140 (90.0%)	62/ 141 (44.0%)
gabor_I	1/ 140 (0.7%)	0/ 141 (0.0%)
gabor_R	104/ 140 (74.3%)	64/ 141 (45.4%)
gabor_RGB	128/ 140 (91.4%)	57/ 141 (40.4%)
MC_nrmI4	128/ 140 (91.4%)	93/ 141 (66.0%)
MC_nrmI7	133/ 140 (95.0%)	116/ 141 (82.3%)
MC_nrmI10	137/ 140 (97.9%)	125/ 141 (88.7%)
MC_nrmIWL	131/ 140 (93.6%)	82/ 141 (58.2%)
MC_nrmR4	127/ 140 (90.7%)	100/ 141 (70.9%)
MC_nrmR7	136/ 140 (97.1%)	120/ 141 (85.1%)
MC_nrmR10	139/ 140 (99.3%)	122/ 141 (86.5%)
MC_nrmRWL	133/ 140 (95.0%)	87/ 141 (61.7%)

Table 3.3: Classification results of texture classifiers with patches when $S_T = 2$.

Method Name	True Detection	False Alarm
cov_R5	80/ 140 (57.1%)	27/ 141 (19.1%)
cov_R7	120/ 140 (85.7%)	59/ 141 (41.8%)
cov_I5	31/ 140 (22.1%)	22/ 141 (15.6%)
cov_I7	71/ 140 (50.7%)	46/ 141 (32.6%)
cov_RGB8	111/ 140 (79.3%)	52/ 141 (36.9%)
cov_RGB9	125/ 140 (89.3%)	58/ 141 (41.1%)
cov_RGB10_c	128/ 140 (91.4%)	58/ 141 (41.1%)
cov_RGB10	123/ 140 (87.9%)	54/ 141 (38.3%)
cov_RGB11	129/ 140 (92.1%)	55/ 141 (39.0%)
cov_RGB12	130/ 140 (92.9%)	51/ 141 (36.2%)
cod_R5	52/ 140 (37.1%)	27/ 141 (19.1%)
cod_R7	122/ 140 (87.1%)	67/ 141 (47.5%)
cod_I5	27/ 140 (19.3%)	16/ 141 (11.3%)
cod_I7	91/ 140 (65.0%)	48/ 141 (34.0%)
cod_RGB8	125/ 140 (89.3%)	56/ 141 (39.7%)
cod_RGB9	129/ 140 (92.1%)	56/ 141 (39.7%)
cod_RGB10_c	132/ 140 (94.3%)	56/ 141 (39.7%)
cod_RGB10	131/ 140 (93.6%)	55/ 141 (39.0%)
cod_RGB11	133/ 140 (95.0%)	52/ 141 (36.9%)
cod_RGB12	131/ 140 (93.6%)	56/ 141 (39.7%)
laws_I	101/ 140 (72.1%)	41/ 141 (29.1%)
laws_R	116/ 140 (82.9%)	51/ 141 (36.2%)
laws_RGB	117/ 140 (83.6%)	40/ 141 (28.4%)
gabor_I	0/ 140 (0.0%)	0/ 141 (0.0%)
gabor_R	73/ 140 (52.1%)	41/ 141 (29.1%)
gabor_RGB	119/ 140 (85.0%)	44/ 141 (31.2%)
MC_nrmI4	113/ 140 (80.7%)	49/ 141 (34.8%)
MC_nrmI7	125/ 140 (89.3%)	62/ 141 (44.0%)
MC_nrmI10	127/ 140 (90.7%)	74/ 141 (52.5%)
MC_nrmIWL	117/ 140 (83.6%)	50/ 141 (35.5%)
MC_nrmR4	114/ 140 (81.4%)	53/ 141 (37.6%)
MC_nrmR7	127/ 140 (90.7%)	79/ 141 (56.0%)
MC_nrmR10	130/ 140 (92.9%)	79/ 141 (56.0%)
MC_nrmRWL	113/ 140 (80.7%)	53/ 141 (37.6%)

Table 3.4: Classification results of texture classifiers with patches when $S_T = 3$.

Method Name	True Detection	False Alarm
cov_R5	45/ 140 (32.1%)	13/ 141 (9.2%)
cov_R7	104/ 140 (74.3%)	41/ 141 (29.1%)
cov_I5	15/ 140 (10.7%)	11/ 141 (7.8%)
cov_I7	52/ 140 (37.1%)	29/ 141 (20.6%)
cov_RGB8	102/ 140 (72.9%)	40/ 141 (28.4%)
cov_RGB9	118/ 140 (84.3%)	48/ 141 (34.0%)
cov_RGB10_c	117/ 140 (83.6%)	41/ 141 (29.1%)
cov_RGB10	119/ 140 (85.0%)	46/ 141 (32.6%)
cov_RGB11	127/ 140 (90.7%)	43/ 141 (30.5%)
cov_RGB12	126/ 140 (90.0%)	40/ 141 (28.4%)
cod_R5	31/ 140 (22.1%)	13/ 141 (9.2%)
cod_R7	109/ 140 (77.9%)	47/ 141 (33.3%)
cod_I5	12/ 140 (8.6%)	6/ 141 (4.3%)
cod_I7	71/ 140 (50.7%)	29/ 141 (20.6%)
cod_RGB8	113/ 140 (80.7%)	42/ 141 (29.8%)
cod_RGB9	121/ 140 (86.4%)	44/ 141 (31.2%)
cod_RGB10_c	124/ 140 (88.6%)	46/ 141 (32.6%)
cod_RGB10	124/ 140 (88.6%)	46/ 141 (32.6%)
cod_RGB11	125/ 140 (89.3%)	42/ 141 (29.8%)
cod_RGB12	124/ 140 (88.6%)	42/ 141 (29.8%)
laws_I	86/ 140 (61.4%)	29/ 141 (20.6%)
laws_R	101/ 140 (72.1%)	41/ 141 (29.1%)
laws_RGB	111/ 140 (79.3%)	35/ 141 (24.8%)
gabor_I	0/ 140 (0.0%)	0/ 141 (0.0%)
gabor_R	57/ 140 (40.7%)	27/ 141 (19.1%)
gabor_RGB	111/ 140 (79.3%)	38/ 141 (27.0%)
MC_nrmI4	103/ 140 (73.6%)	42/ 141 (29.8%)
MC_nrmI7	112/ 140 (80.0%)	41/ 141 (29.1%)
MC_nrmI10	120/ 140 (85.7%)	54/ 141 (38.3%)
MC_nrmIWL	112/ 140 (80.0%)	39/ 141 (27.7%)
MC_nrmR4	101/ 140 (72.1%)	42/ 141 (29.8%)
MC_nrmR7	117/ 140 (83.6%)	57/ 141 (40.4%)
MC_nrmR10	124/ 140 (88.6%)	59/ 141 (41.8%)
MC_nrmRWL	107/ 140 (76.4%)	36/ 141 (25.5%)

Table 3.5: Classification results of texture classifiers with patches when $S_T = 4$.

Method Name	True Detection	False Alarm
cov_R5	22/ 140 (15.7%)	2/ 141 (1.4%)
cov_R7	88/ 140 (62.9%)	29/ 141 (20.6%)
cov_I5	6/ 140 (4.3%)	3/ 141 (2.1%)
cov_I7	31/ 140 (22.1%)	9/ 141 (6.4%)
cov_RGB8	92/ 140 (65.7%)	30/ 141 (21.3%)
cov_RGB9	106/ 140 (75.7%)	30/ 141 (21.3%)
cov_RGB10_c	104/ 140 (74.3%)	30/ 141 (21.3%)
cov_RGB10	107/ 140 (76.4%)	31/ 141 (22.0%)
cov_RGB11	117/ 140 (83.6%)	32/ 141 (22.7%)
cov_RGB12	115/ 140 (82.1%)	29/ 141 (20.6%)
cod_R5	10/ 140 (7.1%)	5/ 141 (3.5%)
cod_R7	102/ 140 (72.9%)	32/ 141 (22.7%)
cod_I5	3/ 140 (2.1%)	2/ 141 (1.4%)
cod_I7	49/ 140 (35.0%)	17/ 141 (12.1%)
cod_RGB8	101/ 140 (72.1%)	27/ 141 (19.1%)
cod_RGB9	114/ 140 (81.4%)	34/ 141 (24.1%)
cod_RGB10_c	116/ 140 (82.9%)	33/ 141 (23.4%)
cod_RGB10	111/ 140 (79.3%)	33/ 141 (23.4%)
cod_RGB11	120/ 140 (85.7%)	33/ 141 (23.4%)
cod_RGB12	116/ 140 (82.9%)	30/ 141 (21.3%)
laws_I	73/ 140 (52.1%)	21/ 141 (14.9%)
laws_R	89/ 140 (63.6%)	29/ 141 (20.6%)
laws_RGB	105/ 140 (75.0%)	26/ 141 (18.4%)
gabor_I	0/ 140 (0.0%)	0/ 141 (0.0%)
gabor_R	45/ 140 (32.1%)	19/ 141 (13.5%)
gabor_RGB	106/ 140 (75.7%)	33/ 141 (23.4%)
MC_nrmI4	94/ 140 (67.1%)	31/ 141 (22.0%)
MC_nrmI7	105/ 140 (75.0%)	34/ 141 (24.1%)
MC_nrmI10	111/ 140 (79.3%)	41/ 141 (29.1%)
MC_nrmIWL	97/ 140 (69.3%)	31/ 141 (22.0%)
MC_nrmR4	80/ 140 (57.1%)	37/ 141 (26.2%)
MC_nrmR7	113/ 140 (80.7%)	37/ 141 (26.2%)
MC_nrmR10	113/ 140 (80.7%)	41/ 141 (29.1%)
MC_nrmRWL	89/ 140 (63.6%)	28/ 141 (19.9%)

Chapter 4

CONCLUSIONS

In this thesis, fire and flame detection algorithms are developed and experimentally tested. In Chapter 2, several ways of using color information are studied and it is concluded that chromatic model 2C is the most simple in terms of computational cost yet efficient model. It has both high true detection rate and relatively low false alarm rate. False alarm rate of this model is neglected because color modeling of flame pixels is not used individually in a flame detection algorithm. They are generally used with other models that use spatial or temporal information. In general final classification stage of most flame detection methods is to apply **AND** operator to decisions of those models, which use color, spatial or temporal information. This means that, false alarm rates can drop with the use of temporal models together but true detection rate never increases. Therefore a high true detection rate with low complexity is crucial in color modelling.

In Chapter 3, covariance descriptors, codifference descriptors, Laws' masks, Gabor filters, and Markov chains are studied with several configurations and it

is concluded that both covariance and codifference descriptors have high performances with proper configurations. It is observed that without using temporal, information separating fire patches from some type of non-fire patches, like patches that are extracted from reflection of sun, is a challenging task. None of the feature sets give good performance by using only spatial information.

Finally in Chapter 3.7, a real-time video fire detection system is developed based on covariance and codifference texture representation method. The method is computationally efficient and it can process 320 by 240 frames at 20 fps in an ordinary computer. Most fire detection methods use color, spatial and temporal information separately, but in this work we use temporally extended covariance matrices to use all the information together. The proposed method works very well when the fire flames are clearly visible. On the other hand, if the fire is small and far away from the camera or covered by dense smoke the method might perform poorly.

Bibliography

- [1] I. Phillips, W., M. Shah, and N. Da Vitoria Lobo, “Flame recognition in video,” in *Applications of Computer Vision, 2000, Fifth IEEE Workshop on.*, 2000, pp. 224–229.
- [2] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, “An early fire-detection method based on image processing,” in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 3, 24-27 2004, pp. 1707 – 1710 Vol. 3.
- [3] B. U. Töreyn, Y. Dedeoğlu, U. Güdükbay, and A. E. Çetin, “Computer vision based method for real-time fire and flame detection,” *Pattern recognition letters*, vol. 27, no. 1, pp. 49–58, 2006.
- [4] B. U. Töreyn, Y. Dedeoğlu, and A. E. Çetin, “Flame detection in video using hidden markov models,” in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 2, 11-14 2005, pp. II – 1230–3.
- [5] B. U. Töreyn and A. E. Çetin, “Online detection of fire in video,” in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 17-22 2007, pp. 1–5.

- [6] W. Hong, J. Peng, and C. Chen, “A New Image-Based Real-Time Flame Detection Method Using Color Analysis,” in *IEEE International Conference on Networking, Sensing and Control*, 2005, pp. 100–105.
- [7] T.-F. Lu, C.-Y. Peng, W.-B. Horng, and J.-W. Peng, “Flame feature model development and its application to flame detection,” in *Innovative Computing, Information and Control, 2006. ICICIC '06. First International Conference on*, vol. 1, aug. 2006, pp. 158–161.
- [8] Z. Zhang, J. Zhao, Z. Yuan, D. Zhang, S. Han, and C. Qu, “Color Based Segmentation and Shape Based Matching of Forest Flames from Monocular Images,” in *2009 International Conference on Multimedia Information Networking and Security*. IEEE, 2009, pp. 625–628.
- [9] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [10] O. Tuzel, F. Porikli, and P. Meer, “Region covariance: A fast descriptor for detection and classification,” *Computer Vision–ECCV 2006*, pp. 589–600, 2006.
- [11] H. Tuna, I. Onaran, and A. E. Çetin, “Image description using a multiplier-less operator,” *Signal Processing Letters, IEEE*, vol. 16, no. 9, pp. 751–753, sept. 2009.
- [12] K. Laws, “Rapid texture identification,” in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 238, 1980, pp. 376–380.

- [13] A. Jain and F. Farrokhnia, “Unsupervised texture segmentation using gabor filters,” in *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*, 4-7 1990, pp. 14 –19.