AUTOMATIC TV LOGO DETECTION AND CLASSIFICATION

by

Nedret ÖZAY

B.S., Computer Engineering, Dokuz Eylul University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2009

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis supervisor Prof. Bülent Sankur for his guidance, patience, and support throughout the thesis. I would like to thank Prof. Fikret Gürgen, Prof. Lale Akarun, Prof. Ethem Alpaydın, and Assist. Prof. Burak Acar for participating in my jury, and their useful comments and feedbacks.

I would like to thank my family for their support and encouragement throughout my education career. And my sincere gratitudes to my wife for her endless love.

I would like to express my gratitude to TÜBİTAK for their support with BIDEB 2210-National Scholarship Programme for MSc Students.

ABSTRACT

AUTOMATIC TV LOGO DETECTION AND CLASSIFICATION

Television (TV) logos are the only semantic objects that appear commonly in all TV broadcast videos. And they can be utilized in the development of many useful applications such as TV commercial detection, and audience measurement. In this study, we have developed an automatic TV logo identification system. The proposed TV logo identification system consists of two parts, namely, TV logo detection and TV logo classification. In the TV logo detection part, we utilized from the idea that 'the broadcast video content is changing over time except the TV logos' and we used *time averaged edges* method to obtain static regions (TV logos) in broadcast videos. In the TV logo classification part, we have used Support Vector Machine(SVM) as classifier. We have compared some well known subspace analysis methods such as Principle Component Analysis (PCA), Non-negative Matrix Factorization (NMF), Independent Component Analysis (ICA), and Discrete Cosine Transform (DCT) to find best feature to describe TV logos. Before applying the subspace analysis methods, all logo images are converted into a fixed size representation by using Grid Descriptor(GD) method. For classification experiments, a TV logo DB of 3040 images is constructed from 152 different TV channels. The best classification performance is obtained by ICA2 with an accuracy rate of 99.21%. For the logo detection and identification experiments, we have collected 240 videos from the 12 most popular TV channels of Turkey. The proposed system achieves to 99.17% logo detection rate and 96.03% average accuracy rate for logo identification. Results of the experiments show that the proposed TV logo identification system works with high accuracy rates and can be utilized in an audience measurement process.

ÖZET

OTOMATİK TV LOGO TESPİTİ VE SINIFLANDIRILMASI

Televizyon (TV) kanallarına ait logolar televizyon yayınlarındaki anlam taşıyan yegâne nesnelerdir. Bu logolardan yararlanılarak TV reklam tespiti ve ya izleyici oranlarının ölçümü gibi pekçok faydalı uygulama geliştirilebilir. Bu çalışmada otomatik bir TV logo tanılama sistemi geliştirilmiştir. Bu sistem iki kısımdan oluşmaktadır: TV logo tespiti ve TV logo sınıflandırması. TV logo tespiti kısmında, şu fikirden yola çıktık: 'bir televizyon yayınında tüm içerik zamanla değişir, değişmeyen tek şey TV logolarıdır'. Videodaki değişmeyen bu sabit alanları(TV logolarını) bulabilmek için zamana göre ortalaması alınmış ayrıtlardan yararlanılmıştır. TV logo sınıflandırması kısmında ise sınıflandırıcı olarak Karar Destek Makinesi (SVM) kullanılmıştır. TV logolarını en iyi temsil edebilecek öznitelikleri belirleyebilmek için yaygın kullanılan bazı altuzay analiz yöntemlerinden Temel Bileşenler Analizi (PCA), Negatif Olmayan Matrislerde Çarpanlara Ayırma (NMF), Bağımsız Bileşenler Analizi(ICA) ve Ayrık Kosinüs Dönüşümü (DCT) yöntemleri karşılaştırılmıştır. Bu altuzay analiz yöntemlerini uygulayabilmek için tüm logo imgeleri Izgara Öznitelikleri (GD) kullanılarak sabit boyutlu gösterime dönüştürülmüştür. Sınıflandırma deneyleri için 152 farklı kanaldan toplanmış 3040 imgeden oluşan bir logo veritabanı oluşturulmuştur. En iyi sonucu %99.21 ile ICA2 vermiştir. Logo tespit ve tanılama deneyleri için ise Türkiye'de en çok izlenen 12 TV kanalından kayıtlar yapılmış ve 240 kayıttan oluşan bir veritabanı oluşturulmuştur. Önerilen sistem ile logo tespiti için %99.17, logo tanılama için ise %96.03 gibi başarım oranlarına ulaşılmıştır. Yapılan deneyler sonucunda önerilen TV logo tanılama sisteminin yüksek başarım oranlarıyla çalıştığı ve izleyici oranlarını ölçme sürecinde kullanılabileceği gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

E_i	Edge Image of frame i
L_i	Logo Mask of frame i
S_i	Time Averaged Edge Image of i frames
S_{th}	Thresholded Time Averaged Edge Image
W	Number of frames in time window

ANN	Artificial Neural Network
CRT	Cathode Ray Tube
TV	Television
GSR	Golden Section Rule
CIF	Common Intermediate Format
GD	Grid Descriptors
PCA	Principle Component Analysis
NMF	Non-Negative Matrix Factorization
ICA	Independent Component Analysis
ICA1	ICA Architecture 1
ICA2	ICA Architecture 2

1. INTRODUCTION

In Television(TV) broadcasting industry, every TV channel has a logo to identify the channel. Similar to the functionality of identity cards of people, logos can be thought as identity cards of TV channels. And TV logos are used to declare content ownership of broadcast videos.

Two decades ago, there was only one or two TV channels in most countries (including Turkey). With the advances in TV broadcasting industry, consumer electronics and video technologies many new TV channels has started to broadcast. Today, hundreds of TV channels broadcast in each country. And the digitized content of videos becomes larger and larger, which is hard to manage. Using classical keyword-based annotation to manage the extremely large video content is not an efficient and effective way, due to the problems in manual annotation phase (time consumption, subjectivity,... etc.). There is a need of semantic objects which are automatically extracted from content of video to make Content Based Indexing and Retrieval possible. TV logos are the most important semantic objects in broadcast videos that will help us to overcome content management problem.

TV Logos can be utilized in many applications, such as:

- TV Commercial Detection: TV logos partly or fully disappear while commercials are active. With using that cue we will be able to detect commercials and we can classify parts of broadcast videos as commercial or non-commercial(normal program).
- TV Commercial Removal: This application can be used on stored broadcast videos. Typically, a user can save a broadcast video by using a recorder, for example using a Personal Video Recorder(PVR). And then, the commercial blocks can be cut by finding the frames of commercial blocks and removing them from video. Thus, in the playback phase user will be able to watch the recorded video without commercials.

- TV Logo Removal: Sometimes a video content owned by a TV channel is rebroadcast by another TV channel and two different TV logos appear on the screen which leads to a viewing displeasure. To overcome this viewing displeasure a logo removal application can be used before rebroadcasting.
- Audience Measurement (Rating Measurement): Nowadays, most of the channels determine their schedules according to results of rating measurement process. Even quality of the programs is effected by that measurement process. Therefore audience measurement is a crucial process that has to be performed with high accuracy rates. TV logo identification can be used in the process of audience measurement. In our thesis, we have developed a TV logo identification system that can be used as an audience measurement application.
- Video Archive Classification and Search: Everyday broadcast video records are added to video archives. For classification of records and search in those video archives TV logos can be used.
- Protection Of Non-CRT Displays [6]: With the advances in consumer electronics, traditional CRT (Cathode Ray Tube) displays are replaced with new thin panels (i.e. LCD - Liquid Crystal Display, and Plasma Display). And a ghost effect problem arises with the new non-CRT displays. When a static region stays on the screen for a long time, that static region becomes blur, and it may cause to deformation on the panel. Since logos are static regions, TV logo detection can help to localize the static region, and run some panel protection algorithms.

In the logo identification point of view, the problem mainly consists of three subproblems, which are:

1. TV Logo Detection: Detection of TV logos means locating the logo in a video frame. Typical input to a TV logo detection system is a broadcast video, and output is a boolean to imply logo presence (i.e. *logo exist* or *logo does not exist*). If the answer is positive then position of logo is also provided to locate the logo in the video frame. Typical output of a logo detection process is shown in Figure 1.1, detected logo is surrounded with a red rectangle. Much of the work in the literature is concentrated on this part of the problem, those works will be given

Figure 1.1. TV logo detection

in section 1.1 Literature Review.

2. TV Logo Tracking: After detecting logo in a frame, tracking process is started. Tracking is conducted to detect logo disappearance. Typically a TV logo disappears because of two main reasons, first one is a commercial break, second one is a channel change. In a commercial break a logo is partly or fully disappeared. In a channel change operation, current logo is disappeared and a new logo is appeared on the video frame. TV logo tracking can be though as a part of continuous logo detection process. Therefore, we explain the whole process together in Chapter 2 TV Logo Detection.
3. TV Logo Classification: In this phase, detected logo is classified according to a preconstructed TV logo database(DB). Typical input to a TV logo classification system is the detected logo candidate, which is the output of the logo detection system, and the output of the system is prediction of the TV channel name.

There are many types of TV logos, but mainly we can categorize logo types according to two main features: motion, and transparency. Combination of two features will yield us four different categories:

- Static-Opaque, or simply Opaque
- Static-Transparent, or simply Transparent
- Animated-Opaque, or simply Animated
- Animated-Transparent

Figure 1.2.

Logo type examples. Rows: static-transparent, static-opaque, animated-opaque

Most of the TV logos are static-opaque (we can simply say *opaque*, due to dominant feature of opaqueness), some of them are static-transparent(we can simply say *transparent*) with an increasing popularity, few of them are animated-opaque(we can simply say *animated*). And until now we have not seen any animated-transparent logo. In Appendix A, list of TV logos and their types are given. Some logo type examples are shown in Figure 1.2. Generally, detection of opaque logos are easier than detection of transparent and animated logos, due to the static content of logo region.

1.1. Literature Review

There are many works on TV logo detection area in the literature. Since logo detection is the first step of logo related applications, most of the works are on logo detection. There is only one work [4] including classification step. We can simply group TV logo detection methods according to used features in detection process as following:

- Using Temporal and Spatial Features
  - Pixel-wise Difference
  - Time Averaged Gradients
- Using Only Spatial Features

The methods of the first group use both temporal and spatial features. Generally methods start with a temporal segmentation step using a sequence of frames, and

continue with a refinement phase using spatial features of a frame. The key idea behind using temporal segmentation is detecting static regions of video. Since video content is changing over time except the TV logo, those static regions are probably TV logos. To extract static regions from video with temporal segmentation, there is a need of motion. If all scene is static then it is not possible to detect logo region, and should be waited for motion. Since motion is one of the main characteristics of broadcast videos, temporal segmentation has a big attraction in logo detection area.

One of the successful methods in the literature is proposed by Albiol *et al.* [1]. They introduce Time Averaged Gradients. In their work TV logo detection algorithm is used in a TV commercial detection application which aims to classify TV shots either as commercial or program shots. A Hidden Markov Model (HMM) is trained with two observations: logo presence, and shot duration. In commercial shots logo does not present and shot duration is shorter. Viterbi decoder is used for shot labeling. Logo presence is decided by detecting stable contours over time. If stable contours exist it can probably be a logo. The algorithm to find stable contours starts with a shot detection to extract one frame per shot. Gradients are calculated for each frame, then gradients are averaged over time to find stable contours as shown in Eq. 1.1.

$$S_i = \frac{i-1}{i} S_{i-1} + \frac{G_i}{i} \tag{1.1}$$

where i is the number of shots processed, G_i is the gradient of i^{th} frame, and S_i is the averaged gradient of i frames. Averaged gradients higher than a threshold yield us stable contours. Afterwards, a refinement procedure is started including morphological operations(closing, opening) in order to connect neighboring pixel groups and remove small regions. As the output of the process, a binary logo mask is obtained. If the obtained logo mask remains stable for a period of time it concludes to presence of TV logo. After detection of TV logo, tracking procedure is started to detect logo disappearance. In our work, we have also inspired very much from Albiol's method for

the logo detection due to its successful detection of opaque and transparent logos.

Another work that can be considered in Time Averaged Gradient group is proposed by Wang *et al.* [8]. They introduce Generalized Gradient method. The method is similar to Albiol's method but with a difference in gradient calculation phase. They use color frames instead of gray-scale frames for calculation of gradients.

Some of the works in the literature [2], [3], [5] uses Pixel-wise Difference method for logo detection. In [2], a two step logo detection algorithm is proposed for an automatic logo detection and removal application. In the first step, pixel-wise frame differencing is performed to detect stable pixels. In the second step, a Bayesian approach is used with two information, a prediction from a pre-trained Artificial Neural Network (ANN), and position information. They introduce the term *logo-let*. A *logo-let* is a fragment of a logo with a size of 12x12 pixel. A raster scan is performed with 12x12 pixel-sized windows in spatial domain. Main purpose is to classify each 12x12 pixel-sized regions as *logo-let* or *non-logo-let*, and then combine *logo-lets* together to construct a logo. Each 12x12 pixel-sized region is queried to ANN to get a probability value of being a *logo-let*. ANN is trained with positive and negative *logo-let* examples. RGB values of each 12x12 region are used as feature vectors. Since TV logos are mostly located in one of the four corners, position of 12x12 region also effects probability of being a *logo-let*. With the combination of those two information, a 12x12 region R is classified as *logo-let* or *non-logo-let* by classifier C according to Eq. 1.2.

$$C : r_R = \begin{cases} 1 & P(r_R = 1 \quad | \quad F_R, l_R) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

where F_R is RGB feature vector of a 12x12 pixel-sized region R, l_R is the location of R, $r_R = 1$ implies region R is a *logo-let*, and $r_R = 0$ implies R is a *non-logo-let*. After obtaining *logo-lets* they are combined to construct logos. And lastly in refinement

step the best logo mask is selected between all logo mask. The logo mask that has minimum edge length is the best logo mask. Minimum edge length is used as a clue of a clear background. Another method in the group of Pixel-wise Difference methods is proposed by Meisinger *et al.* [3]. TV logo detection method is used in a TV logo removal application. Algorithm consists of two steps. In the first step, a difference image D is obtained with pixel-wise difference of sequential frames. In the second step, they use contour relaxation method to refine the rough logo mask obtained in the first step.

Cozar *et al.* [5] propose a logo detection method for a video cataloging application. The method is composed of temporal segmentation, and spatial segmentation parts. In temporal segmentation part, minimum luminance variance regions (MLVR) are obtained with pixel-wise frame differencing method. In spatial segmentation part, many controls are performed to reduce noise and obtain a good logo mask. Connected components are found to check area, bounding box, and aspect ratio of MLVRs.

Another work that uses both temporal and spatial features is proposed by Santos *et al.* [4]. They aims to construct a TV audience measurement application. The method they use in TV logo detection part is similar to Albiol's method but there is a difference in averaging order. They first obtain an averaged frame, then gradient of the averaged frame is calculated. After logo detection part they continue with a classification part. They construct a logo database of 10 TV logos and perform logo classification by logo template matching method. Template matching is performed on edge images.

There are some works in the literature [6], [7] that use only spatial features for logo detection. Ekin *et al.* [6] propose a TV logo detection algorithm for non-CRT display protection application. They first divide a frame into nine regions according to Golden Section Rule (partitioning screen in 3:5:3 scale horizontally and vertically) and extract corner regions. Then they calculate scene models (Color histograms in YPbPr color space) for each corner. The outlier pixels are found by using those scene models, and the outlier pixels are assumed to be pixels of TV Logos. After finding outlier pixels,

a refinement process is started according to type of the logo (i.e.texture or picture based logo). Refinement process includes morphological operations and thresholding.

Duffner *et al.* [7] propose the other method that uses only spatial features. Their main aim is detecting transparent TV logos. They construct an Artificial Neural Network (ANN) with positive and negative examples of a specific transparent TV channel logo. They use an image pyramid to feed pixels to ANN in different scales. Location of logo is determined according to results of the ANN. Gunsel *et al.* [10] utilizes from TV logos to develop a Content-Based TV News Program Management System. They classify each shot either as Commercial or News according to logo presence. They use frame templates and perform template matching to decide logo presence.

If we put all these methods together we can say that, Pixel-wise Differencing methods are used widely but due to changing content of transparent logos they are not good as Time-Averaged Gradients for transparent logos. Time-Averaged Gradients are good at both opaque and transparent logos. And utilizing temporal features beside spatial features is a better choice than using only spatial features.

1.2. Our Approach and Contributions

In this thesis, we have developed an automatic TV logo identification system. So far most of the works in the literature concentrated on the logo detection part and does not cover the classification part. We try to fill this gap by putting TV logo classification on top of the TV logo detection process. We have constructed a TV logo DB consisting of almost all logos of Turkish TV channels and some of European TV channels (totally 152 TV channels) for the classification step. We have compared results of many features such as PCA, NMF, ICA, and DCT to find best feature to describe TV logos. We have used Support Vector Machines (SVM) in classification step. SVM is a robust method and used in many pattern recognition problems resulting with high accuracies.

In the TV logo detection part, we have utilized from some works in the literature especially from [1]. We have used Time-Averaged Edges method to detect TV logos

in broadcast videos. In order to increase logo detection rate we introduce the adaptive structuring element method. And for more robust thresholding we use hysteresis thresholding.

Finally, we construct the TV logo identification system by combining detection and classification parts. In order to achieve a robust identification system, a decision system has been designed. The decision system is used to eliminate inconsistent SVM predictions.

1.3. Outline of the Thesis

The organization of the thesis is as following, in Chapter 2 we describe the method that we used for logo detection in broadcast videos. Since logo tracking is a part of our continuous logo detection system, it is also presented in the same chapter. Then classification part is explained in Chapter 3, starting with explanation of well known features used in many pattern recognition problems such as, PCA, ICA, NMF, DCT, and going on with description of the SVM classifier. The complete logo identification system which is obtained by combining the logo detection and logo classification systems is described in Chapter 4. The experiments conducted on classification and identification systems are given in Chapter 5. Finally, conclusions drawn in Chapter 6.

2. TV LOGO DETECTION

In the first part of the TV logo identification problem, we try to detect the region TV logos. Detection of a TV logo means locating the region of interest(ROI) of the TV logo on a frame. In our logo detection method, no manual interaction is needed, all processes are performed in a fully automatic manner. We were inspired very much by Albiol's method [1] and have used time averaged edges to detect logos. Albiol's method mainly covers opaque and transparent logos but also works for some animated logos. Since there are very few animated TV logos, we have focused on developing a robust TV logo detection system that works with opaque and transparent logos.

For the detection of TV logos, we used the stationarity property of TV logos. Since motion is one of the common features of all TV broadcast videos, the only static region over time is the logo region. Therefore, by detecting the static regions over time we hope to be able to detect TV logos. However, there are some TV programs such as talk shows, etc. that have very little or almost no motion, and it is hard to detect TV logos in such sequences due to absence of motion. Fortunately, in such programs there are multiple cameras and the video content is changing when switching among active cameras, which give us a mean to extract the static regions that are common to all camera outputs.

We have two figures to describe our logo detection method. In Figure 2.1, the flowchart of our logo detection method is given. To illustrate the proposed method, logo detection process is shown on a broadcast video example in Figure 2.2. We use both spatial and temporal features to locate TV logos. The algorithm starts with getting one frame per second from a broadcast video to process. Since logos can appear only in one of the corners, four corner regions are obtained. Then for each corner image, edge detection is performed. Edges are averaged over time, and thresholded to obtain stable contours. Those stable contours are converted to binary logo masks by utilizing morphological operations (i.e. closing, opening). After the morphological operations, each logo mask is checked according to some shape constraints (i.e. Area, aspect ratio,

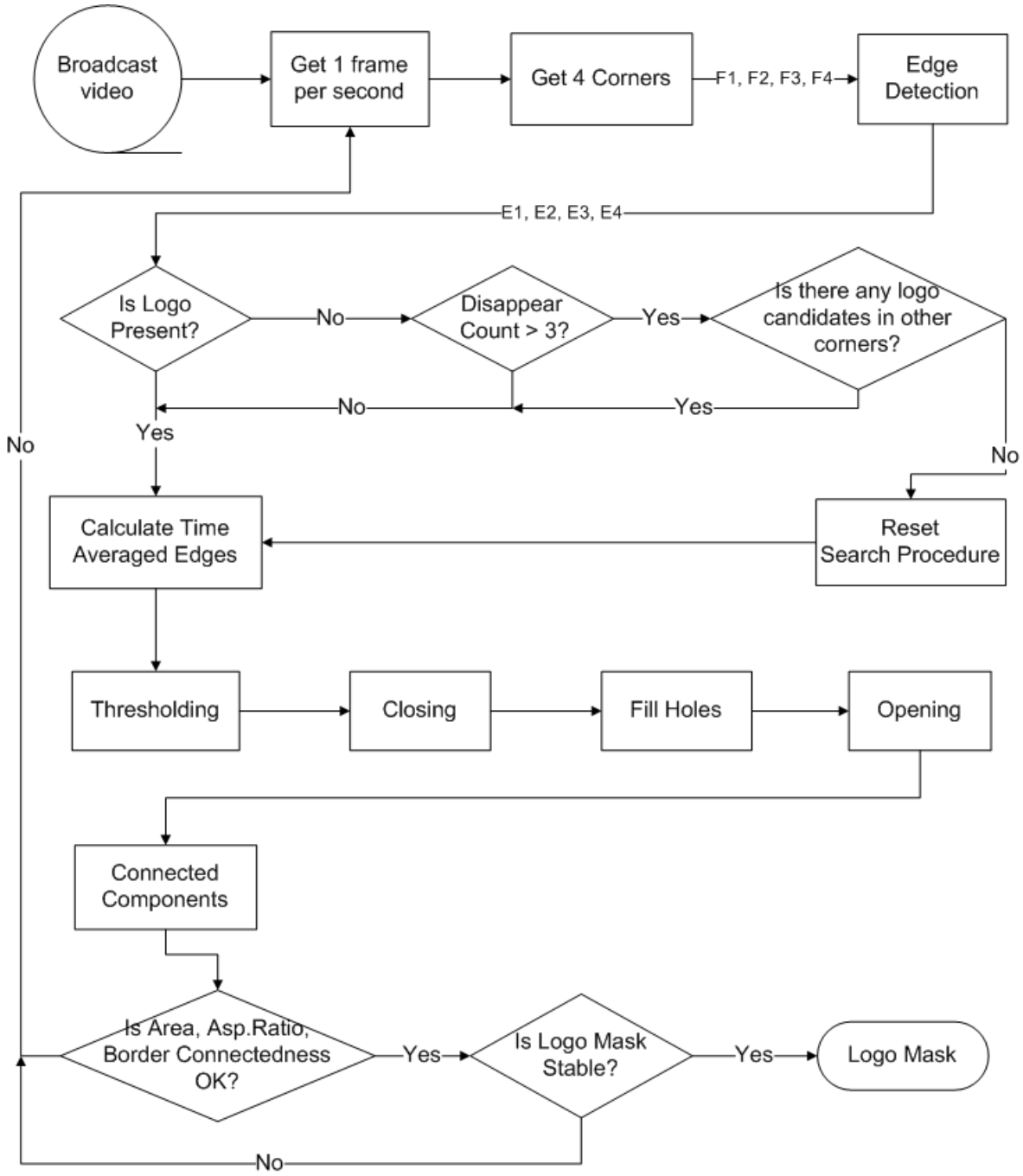


Figure 2.1. Logo detection flowchart

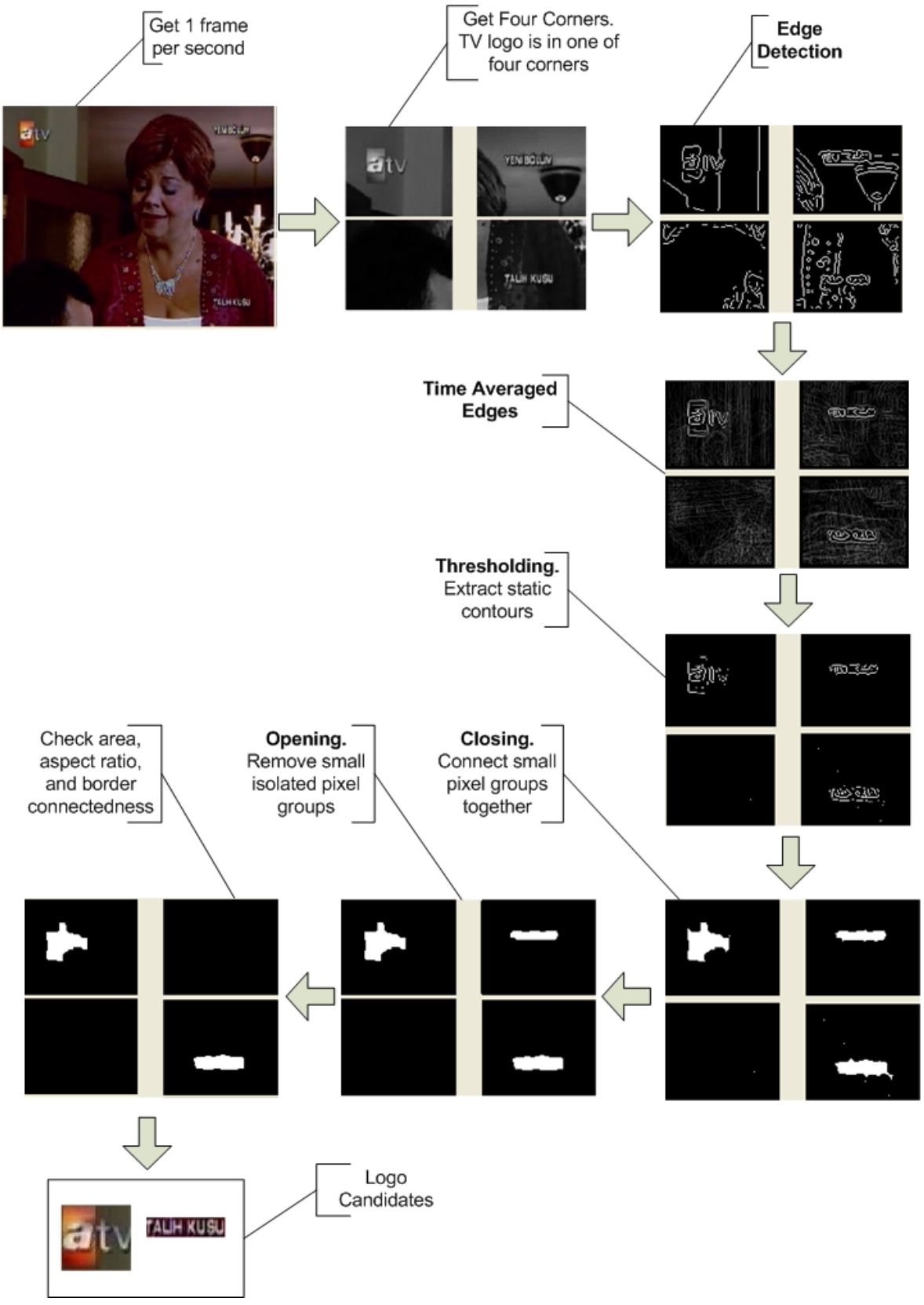


Figure 2.2. Logo detection example

etc.) to make sure that it corresponds to a logo candidate. The logo masks that satisfy the constraints are promoted as logo candidates.

The output of the detection process yields regions that are static in broadcast videos. Notice, however, that there can be many such regions that are both static and containing text on the frame. These objects can represent textual information for the watched program, or indicate some special event, announcement etc. In general, separating TV channel logos from program logos or program name texts is not an easy task. Some shape features (Area, Aspect Ratio, etc.) of TV logos are used as constraints to eliminate static regions which can not possibly be a TV logo. But some program logos or program texts can satisfy the shape constraints and can not be easily eliminated at the output of detection process. An example of such a program text can be seen in Figure 2.2. Program name text at the lower right corner is also detected as a logo candidate. Those remaining static regions will be eliminated in the identification step, as explained later in Section 4.

The changes related to TV logos in broadcast videos occur due to two reasons, the first one is channel switching which is performed by the viewer, the second one is a commercial block which is activated by the broadcast station. In the channel switching case, the logo of the current TV channel disappears and the logo of the new TV channel is appears not necessarily in the same corner. In the case of commercial, that is, when a commercial break starts, the TV logo partly or fully disappears on the frame. TV logos come back when the commercial break ends, and normal program is resumed. Thus, TV logo tracking algorithm should cover these two cases by continuously monitoring the presence of logo.

2.1. Terminology

In this section, we try to introduce some concepts that are mentioned frequently throughout the thesis.

Edge Image: We obtain edge images as the output of edge detection process.

Edges are shown as white pixels on a black background. An example edge image is shown in Figure 2.3a.

Logo Mask: Logo mask is a binary image where objects (in our case logos) are shown as white pixels on a black background. Logo mask is obtained from edge image by using morphological operations, and used to locate logos. Logos are assumed to be located under the bounding box of the logo mask. A logo mask example is shown in Figure 2.3b.

Logo Candidate: Logo candidate is a gray-scale or coloured image, and obtained by finding the image region that resides under bounding box of logo mask. An example logo candidate is shown in Figure 2.3c. Logo candidates are the output of logo detection process, and given as input to logo classification process.

(a) (b) (c)

Figure 2.3. a) An edge image b) A logo mask c) A logo candidate

2.2. Obtaining Corner Regions

TV logos can appear only in one of the four corners of a frame. Therefore, in detection process we do not need to use the whole frame, instead we just consider the corner regions where a logo can be present. Thus, we narrow our search area and reduce required amount of processing. We have used a modified version of Golden Section Rule (GSR)[11] technique which is also used in [6].

We divide a frame into nine regions in 3:5:3 scale, horizontally and vertically, as shown in Figure 2.4 and extract the four corner regions. We had many experiments on different channel videos, and we have seen some exceptions where the 3:5:3 subdivision was not adequate for some TV channels. Therefore, we have added a margin parameter to enlarge corner regions and to ensure that logo always resides in one of the corner regions. For videos in CIF resolution (i.e. 352x288) we have used a margin of 10 pixels.

Figure 2.4. 3:5:3 Frame division according to GSR

And, the sizes of corner regions are 106x89.

2.3. Edge Detection

One of the crucial steps of our logo detection method is edge detection. We know that, since TV logos have different appearance than video content, we can obtain TV logo edges in the output of edge detection process, even if the TV logo is a transparent logo.

We have selected Canny edge detection method [12] for edge detection, due to its popularity between edge detection methods. The Canny method finds edges by looking for local maximum of the gradient of image. At the first step, the image is blurred using a Gaussian filter to reduce noise. Then the gradient of the image is calculated with orientation information. Afterwards a non-maximum suppression step is conducted to obtain thin lines. Finally a two step thresholding is applied. Two thresholds are used to detect strong and weak edges, and the weak edges are included in the output only if they are connected to strong edges. This method is therefore less likely than the others to be affected by noise, and more likely to detect true weak edges.

There is one thing that need to be mentioned in the edge detection method. Since TV logos generally are found near the center of corner regions, they can not be connected to frame borders. Therefore, any edge found in the 3 lines (horizontally and vertically) in the neighborhood of frame borders are removed. Removal of those border edges reduces the side effects caused by morphological operations. An example of such

(a) (b) (c)

Figure 2.5. An example of side effect caused by border edges a) After edge detection b) After closing c) After hole filling

a side effect is shown in Figure 2.5. The vertical edge at right side of the left most image creates an undesired hole, that is filled out and leads to a deformation on the logo mask. By removing edges in first and last 3 lines, we overcome those side effects.

2.4. Time Averaged Edges

As we explained before, the key point in our logo detection algorithm is finding static regions in a sequence of frames. We utilize the concept of *stable contours* to determine static regions. In a sequence of frames, we can obtain *stable contours* by averaging edges of each frame in the sequence. We previously mentioned Albiol's method for finding time averaged gradients in Eq. 1.1. We have slightly modified the equation and have the following Eq. 2.1 to find time averaged edges,

$$S_i = \alpha * S_{i-1} + (1-\alpha) * E_i : \begin{cases} \alpha = \frac{i-1}{i}, & \text{if } i \leq n_{refresh} \\ \alpha = \frac{n_{refresh}-1}{n_{refresh}}, & \text{otherwise} \end{cases} \tag{2.1}$$

where i is index of processed frame, E_i is edges of i^{th} frame, S_i is time averaged edges. The difference between Eq. 2.1 and Eq. 1.1 is the maximum limit of weight of S_{i-1}, or in other words, minimum limit of weight of current frame (E_i). We added $n_{refresh}$ parameter to obtain such a limitation. If $n_{refresh}$ is too big then we can not make quick response to logo changes and can not update logo mask in a reasonable time. We have used $n_{refresh} = 20$ in our experiments with 1 minute duration videos(i.e. 60 usable frames out of 1500 one). We guarantee thus weight of the current frame would

be 0.05 at least, which implies that after 20 frames logo mask will be totally updated. With this lower bound on the weight, we assure that the contribution of the current frame does not vanish, and that the system can react to changes in a reasonable time. For example, think about a channel switching operation. If the new channel logo appears in the same corner as the previous channel logo, then new logo mask can not be obtained quickly due to overlapping regions with previous logo. To obtain correct logo mask, we need to rapidly refresh time averaged edges. The setting of $n_{refresh}$ enables us to adequately refresh the edge field and to recover the correct logo mask quickly.

2.5. Thresholding

After calculation of time averaged edges, we perform a thresholding operation in order to highlight the *stable contours*. We use two different thresholding mechanisms, first one is simple thresholding which uses one threshold value, second one is hysteresis thresholding which uses two threshold values (i.e. high and low thresholds).

2.5.1. Simple Thresholding

For each pixel (x, y) in time averaged edges image, S_i, a threshold Th is applied as shown in Eq. 2.2.

$$S_{th}(x,y) = \begin{cases} 1, & \text{if } S_i(x,y) > Th \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

where S_{th} is a binary image, and the output of the thresholding operation. White pixels (i.e. pixels with value 1) in S_{th} shows the *stable contours*.

(a) (b) (c)

Figure 2.6. Hysteresis thresholding a) Output of high threshold b) Output of low threshold c) Final output.

2.5.2. Hysteresis Thresholding

Hysteresis thresholding method is a two level thresholding method which is also used in Canny edge detector method [12]. There are two thresholds in this method, namely high threshold and low threshold. In the first step of the method high threshold is applied to image and strong edges are obtained. In the second step low threshold is applied and weak edges are obtained. And finally weak edges are promoted as strong edges only if they are connected to strong edges. Note that, since logos are located in the center of corner regions we do not consider the weak edges which are close to borders. Thus we avoid side effects of noisy weak edges located near to borders. Illustration of hysteresis thresholding is given in Figure 2.6.

2.6. Morphological Operations

We utilized morphological operations to obtain a logo mask from an edge image. Morphological operations that we used in our algorithm are:

- Closing: In order to strengthen edges, and connect neighbouring pixel groups together.
- Hole Filling: In order to protect some parts of logo masks from being removed in the opening phase.
- Opening: In order to remove small isolated pixel groups (i.e. noise).

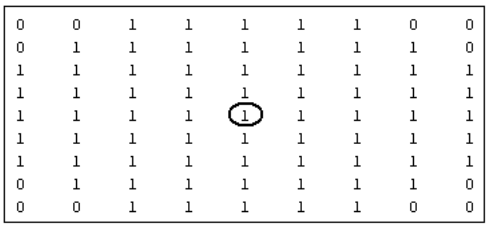


Figure 2.7. Disk structuring element used in closing

2.6.1. Closing

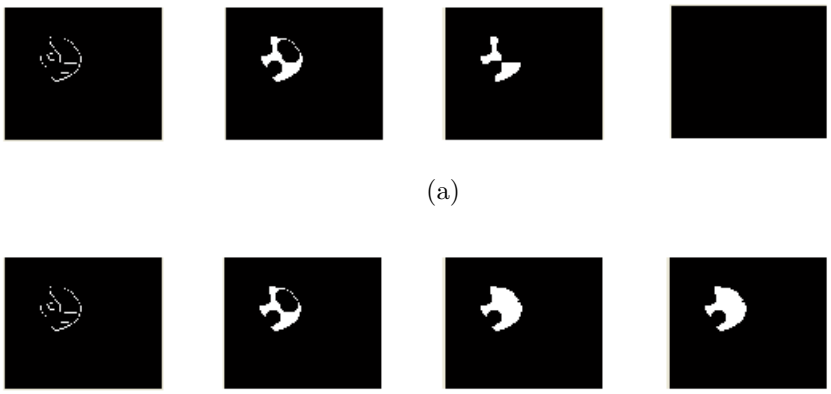
Morphological closing operation is used to connect small pixel groups together which are close enough to each other. Closing operation is completed by applying two basic morphological operations sequentially, dilation, and erosion.

An example to the closing operation can be seen in Figure 2.2. Edge images are converted to logo mask by applying morphological closing operation.

Morphological operations use a structuring element. Size of structuring element affects the impact of operation. The structuring element must be judiciously chosen; if the structuring element is too big, then isolated background pixel groups may appear as part of the logo mask and can distort the original logo mask. In our experiments, we used a disk structuring element with a radius $r = 5$ for CIF size videos. The pixel mask of the disk structuring element is shown in Figure 2.7.

2.6.2. Hole Filling

Since gradients are used to extract edges, no edge can be found inside body of TV logo if body is composed of a uniform color or texture. Often big holes appear in the body of TV logo. Furthermore thin borders of a TV logo may become too weak edges and be removed in the opening phase. In order to avoid loss of such thin borders, we need to fill the holes in the TV logo body. An example of effects of hole filling is shown in Figure 2.8. In Figure 2.8a no hole filling is applied and opening operation damages the logo mask, and finally logo mask disappears after shape constraint controls. In Figure 2.8b the same case is shown where logo mask is preserved thanks to hole filling.



(a)

(b)

Figure 2.8. Effect of hole filling. a) No hole filling applied b) Hole filling applied (Images left to right: edge image, after closing, after opening, after shape constraints applied).

2.6.3. Opening

Morphological opening operation is used to remove small isolated pixel groups. Opening operation is completed by applying two basic morphological operations sequentially, erosion and dilation.

In morphological opening operations, a bigger structuring elements would yield to a smaller logo mask, but at the same time clear all background noise components. An optimum structuring element should remove all noises in the image without distorting the logo mask itself. In our experiments, we used a rectangular structuring element with a size of 3x5 for CIF size videos, which is determined experimentally. The pixel mask of rectangular structuring element is shown in Figure 2.9.

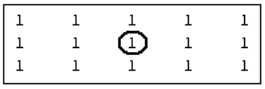


Figure 2.9. Rectangular structuring element used in opening

2.6.4. Adaptive Structuring Element Method

This method applies morphological operations by using two different structuring element groups, namely big structuring element group and small structuring element

group. In the first iteration, big structuring elements are used in closing and opening, and if no logo mask is obtained in a corner, then small structuring elements take place in morphological operation. The objective of the method is to increase the chance of extracting the logo mask by selecting structuring elements adaptively. The adaptive selection of structuring elements reduces the logo mask deformation problems caused by closing operation.

2.7. Shape Constraints

Morphological operations yield logo mask candidates which may or may not correspond to a genuine TV logo in the sequence of frames. TV logos have shape characteristics, and the extracted logo mask should satisfy those characteristics to qualify as a logo candidate. In this section, we try to verify that the obtained logo mask has features that are common to all TV logos. The shape constraints can be listed as,

- Area: Area of the logo mask should be in a certain range. Such as,

$$Area_{min} < Area < Area_{max}$$

- Aspect Ratio: Aspect ratio (i.e. width/height) of the logo mask should be in a certain range. Such as,

$$AR_{min} < AR < AR_{max}$$

- Border Connectedness: Logo mask should not be too close to borders. Such as,

$$dist(logo, border) > dist_{min}$$

In our experiments, for CIF size videos we used $Area_{min} = 280$ (3% of corner region area), and $Area_{max} = 3773$ (40% of corner region area) as area constraint,

$AR_{min} = 0.3$, and $AR_{max} = 5$ as aspect ratio constraint, and $dist_{min} = 10$ as border connectedness constraint.

By using shape constraints, we are able to remove some static regions which are not TV logos. An example of such a removal can be seen in Figure 2.2. After opening step, one of the three logo masks (the one located in the upper right corner of frame) is removed, because the logo mask can not satisfy the aspect ratio constraint (i.e. $AR > AR_{max}$).

We use Connected Components method to find all isolated objects in a corner region. Then for each object, we find shape features, and we remove objects which can not satisfy shape constraints to be a logo candidate. Area is calculated by simply counting number of pixels in object. For aspect ratio calculation, bounding box of object is found. Then by using width and height of bounding box, aspect ratio is calculated. Bounding box of object is also used in calculating the distance to border. The algorithm used for checking shape constraints can be seen in Figure 2.10.

2.8. Logo Mask Stability

After finding a logo mask from a frame, it is expected the logo mask to be stable over time to make sure it is a TV logo. This stage is the last part of the logo detection process before starting the classification process. If the logo mask is verified as stable then classification process will be started. To check stability of a logo mask throughout a sequence of frames, some shape features of logo mask is used. To say a logo mask is stable the following conditions should be satisfied by using shape parameters of current and previous logo masks which are shown in Figure 2.11,

- Area should not change too much. Such as,

$$|Area1 - Area2| < \Delta Area$$

Require: Four binary images B_1, B_2, B_3, B_4 (Outputs of morphological operations)

```
1: //for each of four corner regions
2: for i = 1 to 4 do
3:    Objects ← ConnectedComponents(B_i) //find connected components
4:    for all Obj in Objects do
5:       area ← CalculateArea(Obj)
6:       bbox ← FindBoundingBox(Obj)
7:       ar ← CalculateAspectRatio(Obj)
8:       //Check shape constraints: area, asp.ratio, and border connection
9:       if ( AreaIsOutOfRange(area) OR AspectRatioIsOutOfRange(ar) OR Con-
         nectedToBorders(bbox) ) then
10:         remove Obj
11:      end if
12:   end for
13:   if numberOf(Objects) ≥ 1 then
14:      logoMask(i) ← GetObjectNearestToCenter(Objects)
15:   end if
16: end for
17: return logoMask
```

Figure 2.10. Check Shape Constraints Algorithm

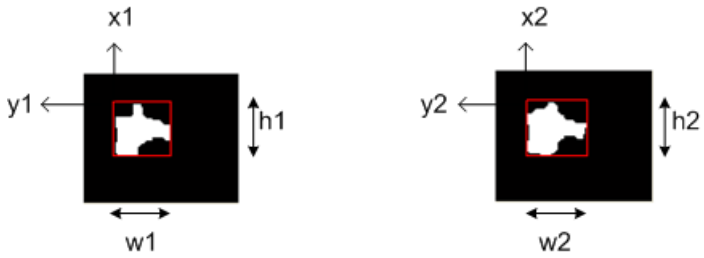


Figure 2.11. Logo mask stability parameters

- Horizontal start position of bounding box should not change too much:

$$|x1 - x2| < \Delta Pos$$

- Vertical start position of bounding box should not change too much:

$$|y1 - y2| < \Delta Pos$$

- Height of bounding box should not change too much:

$$|h1 - h2| < \Delta height$$

- Width of bounding box should not change too much:

$$|w1 - w2| < \Delta width$$

In our experiments, for CIF size videos, we have used 400 for $\Delta Area$, and 20 for $\Delta Pos, \Delta height, \Delta width$ parameters which are determined experimentally.

To make a decision of stability, the conditions given above should be satisfied sequentially for $n_{stability}$ consecutive frames. If stability continues for $n_{stability}$ times without a break, then we can say that logo mask is stable. In our experiments, we used as $n_{stability} = 5$.

(a) (b) (c)

Figure 2.12. a) Edge image of current frame, E, b) Logo mask, L, c) Thresholded time averaged edge image, S_{th}

2.9. Logo Tracking

As we explained before, TV logos can disappear due to two reasons, channel change or commercial block. We should continuously perform a tracking operation to detect logo disappearance. In the flowchart of logo detection (in Figure 2.1) tracking is placed after edge detection process with a diamond shape that has the text "Is Logo Present?". To answer this query, logo existence is checked by the following equation

$$LogoExist = \begin{cases} true, & \text{if } Area(\text{AND}(E, L))/Area(\text{AND}(S_{th}, L)) > Th \\ false, & \text{otherwise} \end{cases} \tag{2.3}$$

where E is the edge image of current frame, L is the logo mask which is obtained in previous frames, S_{th} is the thresholded time averaged edge image. An example of those three images are shown in Figure 2.12. For current frame, if there are enough edges (i.e. E) in the logo mask region (i.e. L) then we can say logo survives. We compare the edge pixel count of the current frame with the edge pixel count of the thresholded time averaged edge image, both measured within the logo mask region. Typical Th value that we use in our experiments is 0.70.

3. TV LOGO CLASSIFICATION

3.1. Feature Selection

To reach a successful TV logo classifier, we have worked on some popular features that have been used commonly in many pattern recognition problems. We have compared the classification results of those features and tried to select the feature that gives the best result for our TV logo classification problem.

3.1.1. Grid Descriptors (GD)

The first feature that we analyzed is Grid Descriptors (GD). A TV logo is projected onto a fixed size of grid (e.g. 10x10 grid cells). Value of each grid cell is the average pixel value of corresponding part of TV logo. A grid example is shown in Figure 3.1.

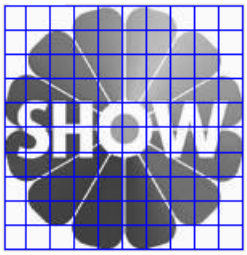


Figure 3.1. A grid example

If we have an image I of $R * C$, and superimpose a grid of $y * x$, each cell will measure by $r = R/y$, and $c = C/x$. Note that r and c should be integers, and number of pixels in each cell should be equal. If the grid can not divide the logo image into equal cells, then an interpolation operation is applied. For example if the image size is 92x142 and grid size is 10x10 then image will be interpolated to have a size of 100x150. After obtaining size of a grid cell (i.e. r, c), we can calculate average pixel value for cell(k, l) as,

$$\mu_{kl} = \frac{1}{r * c} \sum_{(i,j) \in Cell_{kl}} I(i,j) \tag{3.1}$$

After the calculation of values of each cell by traversing the grid from upper left corner to the bottom right corner, we obtain the GD feature vector as shown below (for gray scale images)

$$\mathbf{x} = [\mu_{11}, \mu_{12}, \ldots, \mu_{yx}]^T \tag{3.2}$$

For the colour images (e.g. RGB images) we calculate average value of each plane separately. Then GD feature vector becomes to

$$\mathbf{x} = [\mu_{R11}, \ldots, \mu_{Ryx}, \mu_{G11}, \ldots, \mu_{Gyx}, \mu_{B11}, \ldots, \mu_{Byx}]^T \tag{3.3}$$

There is one more important issue that need to be mentioned here. By calculating the GD we obtain a macro-pixel representation of TV logo images of a given size. This step is also essential for other subspace feature extraction methods, such as PCA, NMF, ICA etc. In other words, since TV logo images have different sizes, we utilize GD method as a preprocessing step to convert the logo images to a given desired size. By using a fixed grid, we obtain fixed size macro-pixel representation of TV logo images. Recall that subspace methods demand input feature vectors of equal size. This preprocessing step can be seen in Figure 3.2. The output of GD block can be input to both dimension reduction and classifier blocks.

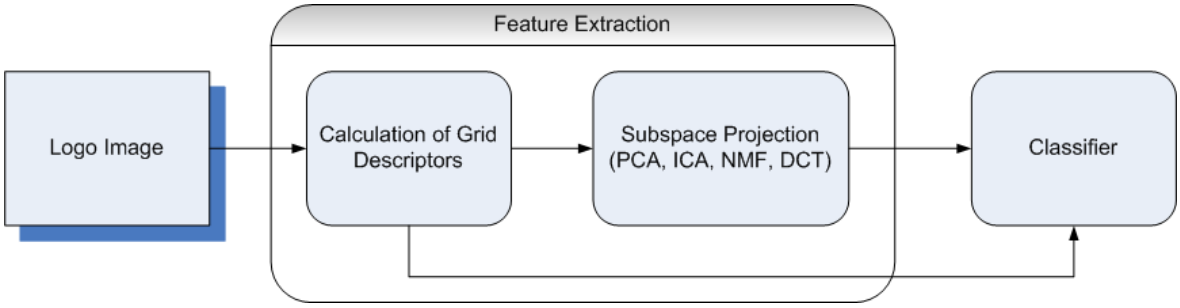


Figure 3.2. Block diagram of the logo classification system

In the sequel, we will denote the MxN data matrix $\mathbf{X}$ as,

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \tag{3.4}$$

where N is the number of logo images, and $\mathbf{x}_i$ is Mx1 macropixel representation of a logo image in the data set.

3.1.2. Principle Component Analysis (PCA)

Principal component analysis (PCA) which was invented in 1901 by Karl Pearson [14], is a vector space transform often used to reduce multidimensional data sets to lower dimensions for analysis. PCA can be used for dimensionality reduction in a data set by retaining those characteristics of the data set that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. Such low-order components often contain the 'most important' aspects of the data.

Often, its operation can be thought of as revealing the internal structure of the data in a way which best explains the variance in the data. If a multivariate dataset is visualized as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA supplies the user with a lower-dimensional picture, a 'shadow' of this object when viewed from its (in some sense) most informative viewpoint.

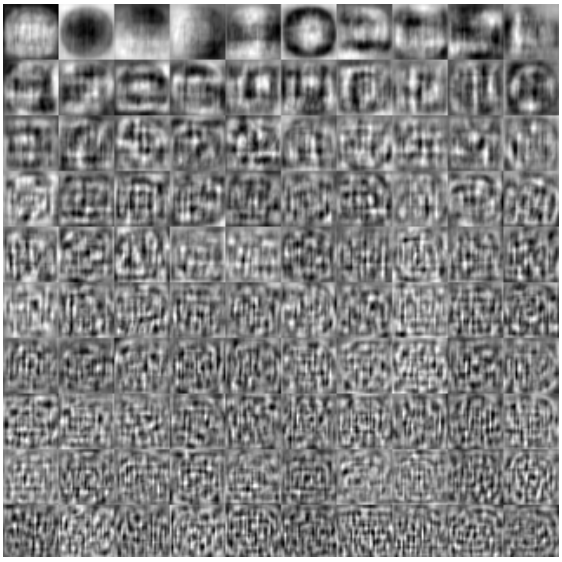


Figure 3.3. PCA basis images

Principle components of a data matrix can be calculated by the calculation of the eigenvalue decomposition of the covariance matrix or singular value decomposition of the data matrix, usually after mean centering the data for each attribute. After the calculation of all principle components, first m components would be enough to contain most of the energy of data (e.g. 95%). And finally new reduced representation of data is obtained by projection of data onto those m principle components. An example to those principle components (i.e. basis images) for gray scale logo images can be seen in Figure 3.3. Pseudocode of PCA is shown below in Figure 3.4.

$$\mathbf{u} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{3.5}$$

$$\mathbf{Cx} = \frac{1}{N-1}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T \tag{3.6}$$

Require: X = MxN matrix (Each column is a logo image),
energy (e.g. 0.95)

1: u ← CalculateMeanImage(X) // Mx1 vector. See Eq. 3.5
2: $\tilde{X}$ ← SubtractMeanImage(X, u) //subtract mean img from each column of X
3: Cx ← CalculateCovarianceMatrix($\tilde{X}$)// See Eq. 3.6
4: [PC, V] ← CalculateEigenvectorsAndEigenvalues(Cx) //PC: eigenvectors, V:eigenvalues
5: v ← GetDiagonal(V)
6: Sort(PC, v) //Sort eigenvalues and corresponding eigenvectors in descreasing order
7: Pm ← GetFirstmEigenvectors(energy, PC, v) //calculate according to cumulative energy content in eigenvalues. See Eq. 3.7
8: R ← ProjectDataOnToPrincipleAxes($\tilde{X}$, Pm) // See Eq. 3.8

Figure 3.4. PCA Pseudocode

$$energy_m = \sum_{i=1}^{m} \mathbf{v}(i) / \sum_{i=1}^{M} \mathbf{v}(i) \quad for \quad 1 \leq m \leq M \tag{3.7}$$

$$\mathbf{R} = \mathbf{P}_m^T \tilde{\mathbf{X}} \tag{3.8}$$

The matrix representation of Eq. 3.8 can be seen in Figure 3.5. $\tilde{\mathbf{X}}$ is mean centered data matrix where columns are given by $\tilde{\mathbf{x}} = \mathbf{x}_i - \mathbf{u}$, $\mathbf{P}_m$ is the matrix of first m eigenvectors where each column is an eigenvector, and $\mathbf{R}$ is the reduced representation of logo images, and $\mathbf{v}(i),\ i = 1, \ldots, M$ denotes the eigenvalues. The sizes of matrices are $\mathbf{R}$ = mxN, $\mathbf{P}_m^T$ = mxM, and $\tilde{\mathbf{X}}$ = MxN.

In the **Training** phase, the classifier is trained with reduced representation of

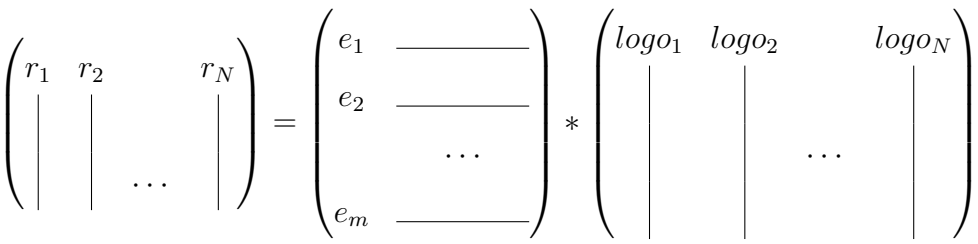


Figure 3.5. Matrix representation of $\mathbf{R} = \mathbf{P}_m^T \tilde{\mathbf{X}}$

logo images (i.e. **R**). In the **Test** phase, the reduced representation of query image is obtained by two sequential steps. First, mean image is subtracted from query image as shown in Eq. 3.9. Then mean centered image is projected on to principle axes as shown in Eq. 3.10.

$$\tilde{\mathbf{x}}_{test} = \mathbf{x}_{test} - \mathbf{u} \tag{3.9}$$

$$\mathbf{r}_{test} = \mathbf{P}_m^T \tilde{\mathbf{x}}_{test} \tag{3.10}$$

3.1.3. Non-Negative Matrix Factorization (NMF)

Non-Negative Matrix Factorization (NMF) is an unsupervised data reduction method that is used in decomposing multivariate data into parts-based representation. The pioneer work by Lee and Seung [17] shows that NMF can be used in learning parts of faces and semantic of text. Both PCA and NMF are matrix factorization techniques, but there is a difference between NMF and PCA, that is, NMF is additive (i.e. parts-based, see Figure 3.6), PCA is both additive and subtractive (i.e. holistic, see Figure 3.3). NMF has the non-negativity constraint (PCA has an orthogonality

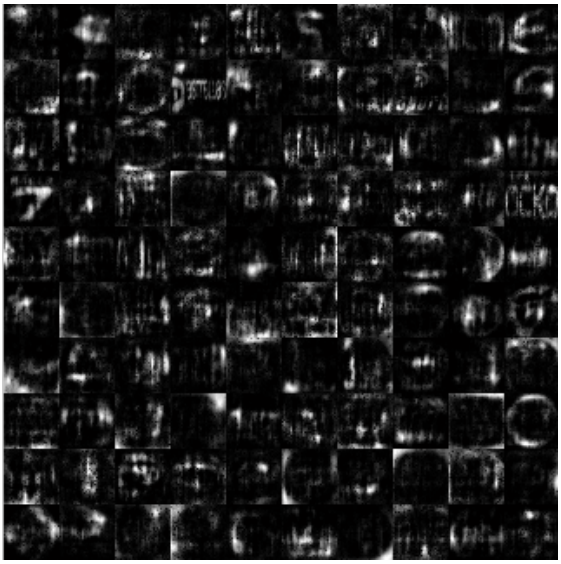


Figure 3.6. NMF basis images

constraint) which yield to a parts-based representation because it only allows additive combinations. For a better understanding of parts-based representation, we can think about a face image example, a face is composed of nose, eyes, mouth, and so on. And we can represent a face by combining those parts in an additive manner.

Given an initial data set represented by an MxN matrix $\mathbf{X}$, where each column is a M-dimensional non-negative vector (i.e. a logo image in our case) from N observations, the decomposition yields to two non-negative matrices $\mathbf{W}$ and $\mathbf{H}$ such that,

$$\mathbf{X} \approx \mathbf{WH} \tag{3.11}$$

where $\mathbf{W}$ is a Mxr basis matrix, and $\mathbf{H}$ is a rxN coefficient matrix. Each column of $\mathbf{W}$ is a basis vector (i.e. a part of the whole), and each column of $\mathbf{H}$ is the reduced representation of the corresponding column in $\mathbf{X}$. NMF basis images (i.e. columns of $\mathbf{W}$) for gray-scale logo images are given in Figure 3.6, due to the non-negativity constraint of NMF each basis image corresponds to a part of logo images. Usually r is

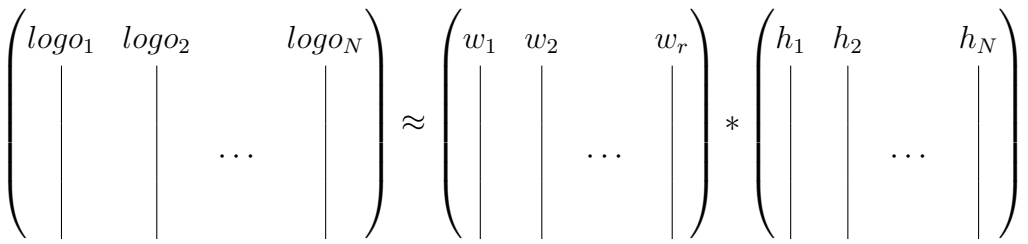


Figure 3.7. Matrix representation of $\mathbf{X} \approx \mathbf{WH}$

chosen such that, $r < MN/(M+N)$. Matrix representation of Eq. 3.11 is shown in Figure 3.7.

The NMF method starts with an initialization of $\mathbf{W}$ and $\mathbf{H}$ matrices with random values. The initialized matrices are iteratively updated to minimize a divergence objective function. The square of the Euclidean distance (Eq. 3.12) can be used as an objective function. By updating values of $\mathbf{W}$ and $\mathbf{H}$ we try to minimize the error between original data ($\mathbf{X}$) and its approximation($\mathbf{WH}$). The pseudocode of NMF algorithm is shown below in Figure 3.8.

$$||\mathbf{X} - \mathbf{WH}||^2 \tag{3.12}$$

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} \frac{(\mathbf{W}^T\mathbf{X})_{a\mu}}{(\mathbf{W}^T\mathbf{WH})_{a\mu}} \tag{3.13}$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia} \frac{(\mathbf{XH}^T)_{ia}}{(\mathbf{WHH}^T)_{ia}} \tag{3.14}$$

Require: X = MxN matrix (Each column is a logo image),
maxIteration, r (reduced dimension).
1: InitializeRandomly (W,H)
2: $i \leftarrow 0$
3: **while** (not Converged and $i <$ maxIteration) **do**
4: H $\leftarrow$ Update(H) //See Eq. 3.13
5: W $\leftarrow$ Update(W) //See Eq. 3.14
6: $i \leftarrow i + 1$
7: checkConvergence(X, WH)//See Eq. 3.12
8: **end while**
9: **return** W, H

Figure 3.8. NMF Pseudocode

For the **Training** of classifier, we use reduced representation of training logo images(i.e. $\mathbf{h}_i,\ i = 1, \ldots, N$). For the **Test** phase, when a new query image ($\mathbf{x}_{test}$) arrives, it is projected onto the pseudo-inverse of basis vectors ($\mathbf{W}$) by the following Eq. 3.15 to obtain NMF representation of query image.

$$\mathbf{h}_{test} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{x}_{test} \tag{3.15}$$

3.1.4. Independent Component Analysis (ICA)

Independent Component Analysis (ICA) is a computational method for separating a multivariate signal into additive subcomponents assuming the mutual statistical independence of the non-Gaussian source signals. It is a special case of blind source separation. A simple application of ICA is the 'cocktail party problem', where the underlying speech signals are separated from a sample data consisting of people talking simultaneously in a room. All voice records in a room($\mathbf{X}$) is generated by mixing all independent microphone records($\mathbf{S}$) with some weights($\mathbf{A}$), as shown in Eq. 3.16,

$$\mathbf{X} = \mathbf{AS} \tag{3.16}$$

where $\mathbf{X}$ is the data matrix, $\mathbf{A}$ is the mixing matrix, and $\mathbf{S}$ is the matrix of independent sources.

Typical algorithms for ICA use centering, whitening (usually with the eigenvalue decomposition), and dimensionality reduction as preprocessing steps in order to simplify and reduce the complexity of the problem for the actual iterative algorithm. Whitening and dimension reduction can be achieved with principal component analysis or singular value decomposition. Whitening ensures that all dimensions are treated equally a priori before the algorithm is run. Algorithms for ICA include infomax, FastICA, and JADE, but there are many others also. We preferred to use FastICA[20] algorithm in our work.

There are two different architectures that are proposed by Bartlett *et al.* [21]. In the first architecture, ICA1, statistically independent basis vectors are obtained. In this architecture basis images are spatially local and sparse. In the second architecture, ICA2, statistically independent coefficients are obtained, where basis images have a global appearance. Details of each architecture is explained in subsequent sections.

3.1.4.1. ICA Architecture 1 (ICA1). The ICA1 architecture and that of PCA have analogous structures, as they are both based on the statistical properties of the basis images. In ICA1, one tries to find statistically independent basis images, whereas in PCA one extracts uncorrelated basis images [23]. ICA1 basis images of gray-scale logo images are shown in Figure 3.9, which are spatially local and sparse.

We can organize our logo images into a data matrix, where each row vector is a different image. In this approach, images are random variables and pixels are trials.

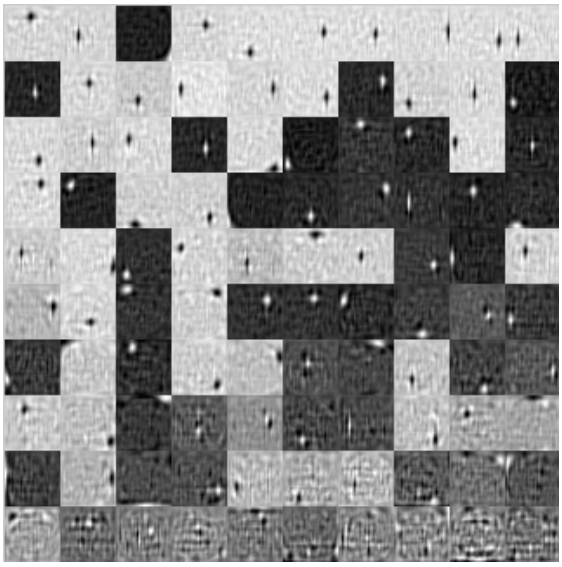


Figure 3.9. ICA1 basis images

In this approach, it makes sense to talk about independence of images or functions of images. Two images i and j are independent if when moving across pixels, it is not possible to predict the value taken by the pixel on image j based on the value taken by the same pixel on image i [22].

The pseudocode of ICA1 is given in Figure 3.10. ICA1 method is started with a PCA on the data matrix $\mathbf{X}$ (each row of $\mathbf{X}$ is a logo image) as shown in Eq. 3.17. Note that ICA is not a dimension reduction technique, thus dimension reduction is achieved by applying PCA in the first step. As the output of PCA first m eigenvectors, $\mathbf{P}_m$, are obtained according to given energy level. ICA is performed on the transpose of $\mathbf{P}_m$ as shown in Eq. 3.18. Here $\mathbf{W}$ is the inverse of $\mathbf{A}$, and is called as unmixing matrix, and is a square matrix(i.e. invertible). $\mathbf{U}$ is the approximation of independent source matrix, $\mathbf{S}$, and contains basis images in its rows.

$$\mathbf{R}_m = \mathbf{X}\mathbf{P}_m \tag{3.17}$$

Require: X = NxM matrix (Each row is a logo image), energy (for PCA dimension reduction)

1: //Apply PCA for dimension reduction.
2: $[R_m, P_m] \leftarrow$ PCA(energy, X) //See Eq. 3.17.
3: //Apply ICA on the transpose of first m eigenvectors.
4: [W, U] $\leftarrow$ ICA(P_m^T) //See Eq. 3.18. U: Basis Images
5: //Calculate ICA coefficients by using reconstruction of X formula.
6: B $\leftarrow$ CalculateICACoefficients() //See Eqs. 3.19 - 3.21.
7:
8: **return** B, U //Return ICA Coefficients, and Basis Images.

Figure 3.10. ICA1 Pseudocode

$$\mathbf{P}_m^T = \mathbf{W}^{-1}\mathbf{U} \qquad (\mathbf{W}^{-1} = \mathbf{A}, \quad and \quad \mathbf{U} \approx \mathbf{S}) \tag{3.18}$$

For reconstruction of $\mathbf{X}$, we can write Eq. 3.19,

$$\hat{\mathbf{X}} = \mathbf{R}_m \mathbf{P}_m^T \tag{3.19}$$

and by substituting Eq. 3.18 into Eq. 3.19 we have

$$\hat{\mathbf{X}} = \mathbf{R}_m \mathbf{W}^{-1}\mathbf{U} \tag{3.20}$$

finally ICA1 coefficients, $\mathbf{B}$, are obtained by Eq. 3.21.

$$\begin{pmatrix} logo_1 & \_\_\_\_\_ \\ logo_2 & \_\_\_\_\_ \\ & \cdots \\ logo_N & \_\_\_\_\_ \end{pmatrix} = \begin{pmatrix} b_1 & \_\_\_\_\_ \\ b_2 & \_\_\_\_\_ \\ & \cdots \\ b_N & \_\_\_\_\_ \end{pmatrix} * \begin{pmatrix} u_1 & \_\_\_\_\_ \\ u_2 & \_\_\_\_\_ \\ & \cdots \\ u_m & \_\_\_\_\_ \end{pmatrix}$$

Figure 3.11. Matrix representation of $\hat{\mathbf{X}} = \mathbf{BU}$

$$\mathbf{B} = \mathbf{R}_m \mathbf{W}^{-1} \tag{3.21}$$

The matrix representation of $\hat{\mathbf{X}}$, $\mathbf{B}$, and $\mathbf{U}$ are shown in Figure 3.11. $\hat{\mathbf{X}}$ has logo images in its rows, and corresponding ICA1 representation of logo images are located in each row of $\mathbf{B}$. Basis images can be seen in the rows of $\mathbf{U}$. The sizes of matrices are $\hat{\mathbf{X}} = N\text{x}M$, $\mathbf{B} = N\text{x}m$, and $\mathbf{U} = m\text{x}M$.

For the **Training** of classifier, we use ICA1 representation of the training logo images(i.e. $\mathbf{B}$). For the **Test** phase, when a new query image ($\mathbf{x}_{test}$) arrives, it is projected onto the principle components ($\mathbf{P}_m$) by the Eq. 3.22 (assuming $\mathbf{x}_{test}$ is a zero-mean image, to obtain a zero-mean image see Eq. 3.9), and then ICA1 representation is obtained by Eq. 3.23.

$$\mathbf{r}_{test} = \mathbf{x}_{test} \mathbf{P}_m \tag{3.22}$$

$$\mathbf{b}_{test} = \mathbf{r}_{test} \mathbf{W}^{-1} \tag{3.23}$$

Figure 3.12. ICA2 basis images

3.1.4.2. ICA Architecture 2 (ICA2). The ICA2 architecture aims to obtain statistically independent ICA coefficients rather than statistically independent basis images. This time logo images are located in the columns of the data matrix. In this approach, pixels are random variables and images are trials. Here, it makes sense to talk about independence of pixels or functions of pixels. For example, pixel i and j would be independent if when moving across the entire set of images it is not possible to predict the value taken by pixel i based on the corresponding value taken by pixel j on the same image.

Require: X = NxM matrix (Each row is a logo image), energy (for PCA dimension reduction)

1: //Apply PCA for dimension reduction.
2: $[R_m, P_m] \leftarrow$ PCA(energy, X) //See Eq. 3.17.
3: //Apply ICA on the transpose of PCA representation of X.
4: [W, U] $\leftarrow$ ICA(R_m^T) //See Eq. 3.24.
5: A = W^{-1}
6: **return** U, A //Return ICA Coefficients, and Basis Images.

Figure 3.13. ICA2 Pseudocode

At the beginning of the method, PCA is applied to reduce dimension as in ICA1 (Eq. 3.17). Then ICA is performed on the transpose of PCA representation of **X**

(i.e. $\mathbf{R}_m^T$) as shown in Eq. 3.24 where each column of $\mathbf{R}_m^T$ is PCA representation of a logo image. Each column of $\mathbf{A} = \mathbf{W}^{-1}$ is a basis image, and each column of $\mathbf{U}$ is the coefficients. ICA2 basis images of gray-scale logo images are given in Figure 3.12, where each basis images have a global appearance. The pseudocode of ICA2 is given in Figure 3.13.

$$\mathbf{R}_m^T = \mathbf{W}^{-1}\mathbf{U} \tag{3.24}$$

The matrix representation of Eq. 3.24 is shown in Figure 3.14. The sizes of matrices are $\mathbf{R}_m^T = m\text{x}N$, $\mathbf{W}^{-1} = m\text{x}m$, and $\mathbf{U} = m\text{x}N$.

$$\begin{pmatrix} | & | & & | \\ r_1 & r_2 & \cdots & r_N \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_m \\ | & | & & | \end{pmatrix} * \begin{pmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_N \\ | & | & & | \end{pmatrix}$$

Figure 3.14. Matrix representation of $\mathbf{R}_m^T = \mathbf{W}^{-1}\mathbf{U}$

For the **Training** of classifier, we use ICA2 representation of the training logo images(i.e. $\mathbf{U}$). For the **Test** phase, when a new query image ($\mathbf{x}_{test}$) arrives, it is first projected onto the principle components ($\mathbf{P}_m$) as shown in the Eq. 3.22 (assuming $\mathbf{x}_{test}$ is a zero-mean image, to obtain a zero-mean image see Eq. 3.9), and then ICA2 representation, $\mathbf{u}_{test}$, is obtained by Eq. 3.25.

$$\mathbf{u}_{test} = \mathbf{W}\mathbf{r}_{test}^T \tag{3.25}$$

3.1.5. Discrete Cosine Transform (DCT)

Discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCT is often used in signal and image processing, especially for lossy data compression, because it has a strong 'energy compaction' property, that is, most of the signal information tends to be concentrated in a few low-frequency components of the DCT (where small high-frequency components can be discarded).

The 2D DCT can be applied by the following Eq. 3.26

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2N}\right] \tag{3.26}$$

where f denotes the 2D form of $\mathbf{x}_i$ (macropixel representation of each logo image), $u, v = 0, \ \ldots \ , N-1$. $\alpha(u)$, and $\alpha(v)$ are defined as follows

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{for } u = 0 \\ \sqrt{\frac{2}{N}}, & \text{for } u \neq 0 \end{cases} \tag{3.27}$$

The image in Figure 3.15 shows combination of horizontal and vertical frequencies for an 8x8 (N = 8) two-dimensional DCT. Each step from left to right and top to bottom is an increase in frequency by 1/2 cycle. For example, moving right one from the top-left square yields a half-cycle increase in the horizontal frequency (goes from white to black). Another move to the right yields two half-cycles (white to black to white). A move down yields two half-cycles horizontally and a half-cycle vertically. The source data (8x8) is transformed to a linear combination of these 64 frequency squares (basis

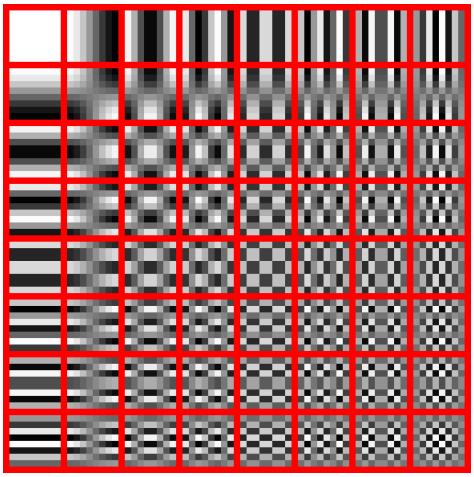


Figure 3.15. DCT basis images for 8x8 blocks

images).

The algorithm for applying DCT on images is shown in Figure 3.16. For each NxN block in the image, DCT of the block is obtained. We have again a NxN matrix but all low-frequency components are located in the upper left corner, and the top-left element is the DC (zero-frequency) component. The low-frequency components contain most of the energy of the NxN block. Thus we just consider the low-frequency components and omit the small high-frequency components. Then the DCT coefficients are sorted into a 1D array by a zig-zag scan as shown in Figure 3.17. As the result of zig-zag scan, we obtain a 1xN^2 array, and the low-frequency components are located at the beginning of the array. We can select the first n_{low} coefficients as shown in Eq. 3.28, thus dimension reduction is achieved.

$$B_{low}(i) \leftarrow B_{zz}(i) \qquad i = 0, \ \ldots \ , n_{low} - 1. \tag{3.28}$$

Require: I = MxM image, N (block size),
n_{low} (number of low freq. coeff.)
numBlocks $\leftarrow (M/N) * (M/N)$
for $i = 1$ to numBlocks **do**
  $B \leftarrow$ getNxNBlock(I, i)
  $B_{DCT} \leftarrow$ DCT(B) //See Eq. 3.26
  $B_{zz} \leftarrow$ ZigZagScan (B_{DCT})
  $B_{low} \leftarrow$ GetLowFrequencyCoefficients (B_{zz}, n_{low})//See Eq. 3.28.
  $I_{DCT}(i) \leftarrow B_{low}$
end for
return I_{DCT} //Return DCT Coefficients of I.

Figure 3.16. DCT Pseudocode

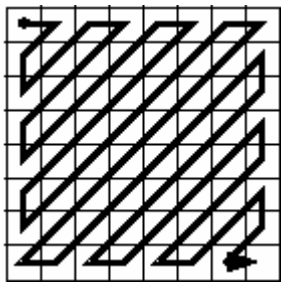


Figure 3.17. Zig-zag scan

3.2. Classification with Support Vector Machines (SVMs)

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Viewing input data as two sets of vectors in an n-dimensional space, an SVM will construct a separating hyperplane in that space, one which maximizes the *margin* between the two data sets. To calculate the margin, two parallel hyperplanes are constructed, one on each side of the separating hyperplane, which are 'pushed up against' the two data sets. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the neighboring data points of both classes, since in general the larger the margin the better the generalization error of the classifier.

Suppose some given data points each belong to one of two classes, and the goal is to decide which class a *new* data point will be in. In the case of SVMs, a data point is viewed as a p-dimensional vector, and we want to know whether we can separate such points with a $p-1$-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. However, we are additionally interested in finding out if we can achieve maximum separation (margin) between the two classes. By this we mean that we pick the hyperplane so that the distance from the hyperplane to the nearest data point is maximized. That is to say that the nearest distance between a point in one separated hyperplane and a point in the other separated hyperplane is maximized. Now, if such a hyperplane exists, it is clearly of interest and is known as the maximum-margin hyperplane and such a linear classifier is known as a maximum margin classifier.

3.2.1. Linear SVM

3.2.1.1. Separable Case. We are given a set of training data, in the form of

$$\{\mathbf{x}_i, y_i\} = \mathbf{x}_i \in \mathbf{R}^d, y_i \in \{-1, 1\}\,, \text{ for } \quad i = 1, \ldots, l. \tag{3.29}$$

where the y_i is either 1 or -1, indicating the class to which the point $\mathbf{x}_i$ belongs. We want to find the maximum-margin hyperplane which divides the points having y_i = 1 (i.e. positive examples) from those having y_i = -1 (i.e. negative examples). The points $\mathbf{x}$ which lie on the hyperplane satisfy

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{3.30}$$

where $\mathbf{w}$ is normal to hyperplane, and the perpendicular distance from hyperplane to origin is $|b|/||\mathbf{w}||$ ($||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$). Let d_+ (d_-) be the shortest distance from the separating hyperplane to the closest positive (negative) example. Define the *margin* of a separating hyperplane to be $d_+ + d_-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin. This can be formulated as follows: suppose that all the training data satisfy the following constraints:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1, \quad \text{for } y_i = +1 \tag{3.31}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1, \quad \text{for } y_i = -1 \tag{3.32}$$

Now consider the points for which the equality in Eq. 3.31 holds. These points lie on the hyperplane $H1 : \mathbf{x}_i \cdot \mathbf{w} + b = 1$ with normal $\mathbf{w}$ and perpendicular distance from the origin $|1 - b|/||\mathbf{w}||$. Similarly, the points for which the equality in Eq. 3.32 holds lie on the hyperplane $H2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$, with normal again $\mathbf{w}$, and perpendicular

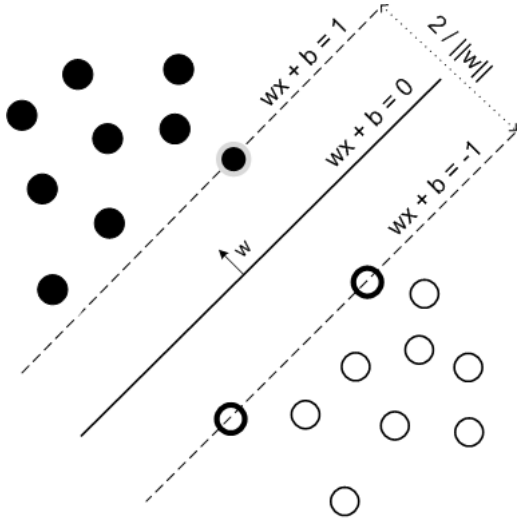


Figure 3.18. A linear SVM classifier (separable case)

distance from the origin $|-1-b|/||\mathbf{w}||$. Hence $d_+ = d_- = |1|/||\mathbf{w}||$ and the margin is simply $|2|/||\mathbf{w}||$. Note that $H1$ and $H2$ are parallel (they have the same normal) and that no training points fall between them. Our aim is finding H1, H2 hyperplanes that have the following two properties

- Classify negative and positive examples correctly (by satisfying Eq. 3.31 and Eq. 3.32)
- Have a maximum margin (by minimizing $\frac{1}{2}||\mathbf{w}||^2$)

It becomes an optimization problem with the given two items above. The optimization problem can be rewritten by combining the Eq. 3.31 and Eq. 3.32 together

$$\text{minimize } \frac{1}{2}||\mathbf{w}||^2, \text{ subject to } y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \tag{3.33}$$

And the optimization problem can be solved by quadratic programming techniques and programs.

The illustration of the classifier on a typical two dimensional case is shown in Fig-

ure 3.18. The training samples located on the hyperplanes are called support vectors, which are shown differently from other samples.

Writing the classification rule in its unconstrained dual form reveals that the maximum margin hyperplane and therefore the classification task is only a function of the support vectors, the training data that lie on the margin. The dual of the SVM can be shown to be:

$$\max \sum_{i}^{l} \alpha_i - \frac{1}{2} \sum_{i}^{l} \sum_{j}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \quad \text{subject to } \alpha_i \geq 0, \text{ and } \sum_{i}^{l} \alpha_i y_i = 0 \quad (3.34)$$

and $\mathbf{w}$ can be defined as

$$\mathbf{w} = \sum_{i}^{l} \alpha_i y_i \mathbf{x}_i \quad (3.35)$$

Support vector training (for the separable, linear case) therefore amounts to maximizing Eq. 3.34 with respect to the α_i, subject to the given constraints, with solution given by Eq. 3.35. In the solution, those points for which $\alpha_i > 0$ are called 'support vectors', and lie on one of the hyperplanes H1, H2. All other training points have $\alpha_i = 0$.

3.2.1.2. Non-Separable Case. What we have stated so far is for the separable case of data (noise-free data). How can we find hyperplanes for a training data that includes noisy samples as shown in Figure 3.19. In 1995, Cortes and Vapnik suggested a modified maximum margin idea that allows for mislabeled examples [30]. If there exists no hyperplane that can split the *yes* and *no* examples, the Soft Margin method will choose

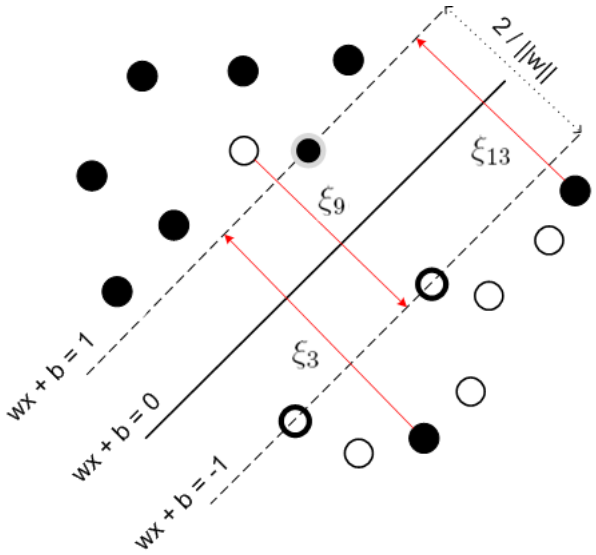


Figure 3.19. A linear SVM classifier with noisy data (non-separable case)

a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces slack variables, ξ_i, which measure the degree of misclassification of the datum $\mathbf{x}_i$

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i, \quad \text{for } y_i = +1 \tag{3.36}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i, \quad \text{for } y_i = -1 \tag{3.37}$$

The objective function is then increased by a function which penalizes non-zero ξ_i, and the optimization becomes a trade off between a large margin, and a small error penalty. The new objective function is defined as

$$\text{minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \tag{3.38}$$

where C is a parameter to be chosen by the user, a larger C corresponding to assigning a higher penalty to errors. And the only difference from the optimal hyperplane case is that the α_i now have an upper bound of C (i.e. $0 \leq \alpha_i \leq C$), and we can rewrite Eq. 3.34 as following

$$\max \sum_i^l \alpha_i - \frac{1}{2}\sum_i^l \sum_j^l \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \quad \text{subject to } 0 \leq \alpha_i \leq C, \text{ and } \sum_i^l \alpha_i y_i = 0 \tag{3.39}$$

The solution is the same as given in Eq. 3.35.

3.2.2. Non-Linear SVM

In the previous section, the original optimal hyperplane algorithm was given for the data that can be separated by a linear function of data. How can we classify a non-linear data that can not be classified by a linear function? In 1992, Boser, Guyon and Vapnik suggested a way to create non-linear classifiers by applying the *kernel trick* to maximum-margin hyperplanes [31]. The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in the transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be non-linear in the original input space. We can map the data to some other Euclidean space H as shown below:

$$\Phi : \mathbf{R}^d \rightarrow H \tag{3.40}$$

Then the training algorithm would only depend on the data through dot products in H, i.e. on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Now if there were a *kernel function* K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, we would only need to use K in the training algorithm. Some common kernels include:

- Polynomial (degree p)

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

- Gaussian Radial Basis Function (RBF)

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}\right)$$

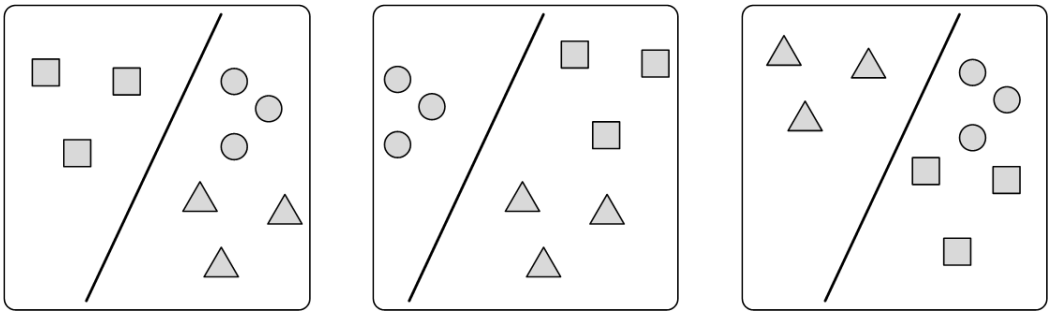
- Sigmoid Function

$$K(\mathbf{x}, \mathbf{y}) = tanh(\kappa \mathbf{x} \cdot \mathbf{y} - \delta)$$

3.2.3. Multiclass SVM

Support Vector Machines (SVMs) were originally designed for binary classification. However multiclass classification problems can be solved by using multiple binary classifiers. We can write the training data from k different classes in the form of

$$\{\mathbf{x}_i, y_i\} = \mathbf{x}_i \in \mathbf{R}^d, y_i \in \{1, \cdots, k\}, \text{ for } \quad i = 1, \ldots, l \tag{3.41}$$



(a) SVM1 (b) SVM2 (c) SVM3

Figure 3.20. One-versus-all method

where y_i is the class of $\mathbf{x}_i$.

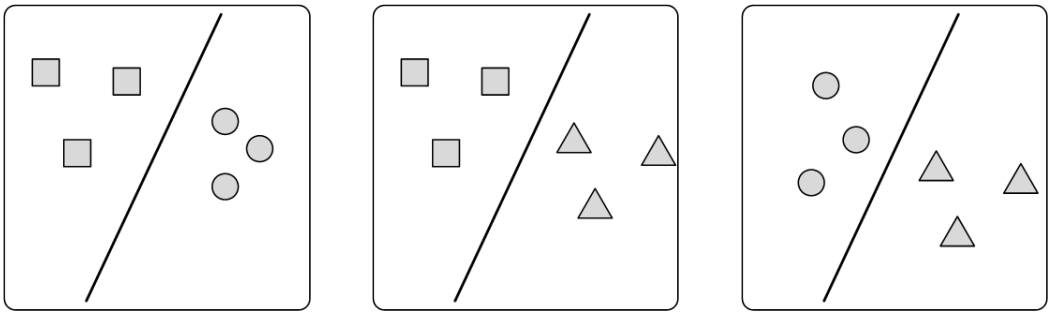
There are two common methods to build binary classifiers to solve multiclass classification problems:

- One-versus-All
- One-versus-One

which are described in the following subsections.

3.2.3.1. One-versus-All. The earliest method used for multiclass SVM classification is probably one-versus-all. For a data of k classes it constructs k SVMs. The i^{th} SVM is trained with all the data of i^{th} class as positive examples, and the rest of the data as negative examples. Thus we can say that, i^{th} SVM gives the score for the query data to be in i^{th} class. The class with the highest score wins the competition(i.e. query data belongs to that class). This is called winner takes-all strategy. A simple example of one-versus-all method is shown in Figure 3.20. There are 3 classes, and 3 SVMs. Each class is trained against the rest of the classes in each SVM.

3.2.3.2. One-versus-One. In the one-versus-one method, one SVM is trained for each pair of classes. Thus each class constructs an SVM with each of the other classes. Totally for k classes, $k(k-1)/2$ SVMs are trained, where each SVM is trained with data of two classes only. To decide winner class, a voting mechanism is used. For each



(a) SVM1 (b) SVM2 (c) SVM3

Figure 3.21. One-versus-one method

SVM the winner class takes one additional vote. And the final winner is found by adding up all votes coming from all SVMs. The class that has the maximum number of votes is the winner(i.e. query data belongs to that class). This is called max-wins voting strategy. A simple example of one-versus-one method is shown in Figure 3.21. For each pair of classes one SVM is trained.

A comparison of the results of the two methods is given in [33]. For the implementation, we have used the LIBSVM [35] software library, which works robustly, and have been used commonly for many classification problems. LIBSVM uses one-vs-one method for multiclass classification.

4. TV LOGO IDENTIFICATION SYSTEM

In the previous two chapters we have explained TV logo detection and classification methods. Now we try to combine those two parts together to obtain a TV logo identification system. A simple block diagram of a typical TV logo identification system can be seen in Figure 4.1. A broadcast video is given as input to the system, and the name of the channel in the given video is expected as the output.

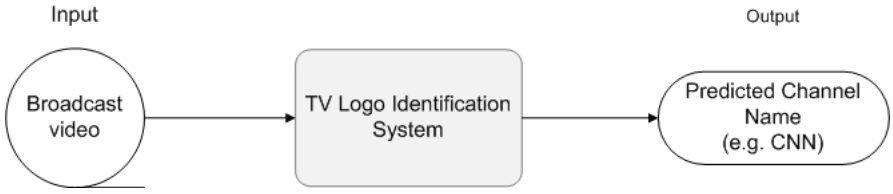


Figure 4.1. A logo identification system

Details of the logo identification system can be seen as a flowchart in Figure 4.2. The system starts with logo detection by processing one frame per second from a given broadcast video. The output of the logo detection process is a binary logo mask, L_i, which contains the information of position of logo candidate on the video frame. By using that logo mask we obtain the logo candidate and it (i.e. an rgb logo image) is given as input to the SVM classifier, which is trained previously on our logo DB (Note that, feature extraction is performed before making query to SVM, this process is embedded in SVM block in the diagram for simplicity). We get best three predictions from SVM with their probabilities (e.g. 1-CNN, probability:0.15; 2-CNBC-E, probability:0.07; 3-TRT1, probability:0.04) after filtering inconsistent SVM predictions by a corner verification operation. Since expected corner is known for each channel (ground truth) we can easily check consistency of the SVM predictions with the corner information. For example, TRT1 logos always appear in the upper left corner, if SVM makes a TRT1 prediction for a logo candidate in one of the other three corners, that TRT1 prediction will not be taken into consideration. Then we check the probability values if they are high enough to be sure we have a good logo mask. If probability values are not high enough we obtain new logo masks from current logo mask and make new SVM queries with the logo candidates resides under them. We

obtain 18 new logo masks from current logo mask by **shifting** Δk pixels (up, down, left, and right), **enlarging** Δk pixels (up, down, left, right, horizontally, vertically, and both directions) and **shrinking** Δk pixels (up, down, left, right, horizontally, vertically, and both directions) bounding box of the logo mask.

On the other hand, we make another query to SVM with using best logo mask, L_{best} which is saved in each iteration of logo detection according to best answers of SVM (i.e. the logo mask that gets highest probability from SVM is the best logo mask). This best logo mask supplies a protection from unexpected faulty results due to the instantaneous changes in current logo mask. We compare the results of L_i and L_{best} to decide which is better, and use the results of the winner to make final decision. In the last step of the logo identification system we have a decision system. The decision system collects the SVM predictions, then give the final decision of the system if the predictions are consistent for a period of time. The consistency checks makes the system more robust. The details of decision system will be given in next section.

4.1. Decision System

After the first experiments on our logo identification system, we have observed that small changes in the logo masks may lead to different SVM results (i.e. inconsistent predictions of the system). Then we decided to construct a robust decision mechanism which can not be affected by instantaneous changes in SVM results. The main logic of such a decision mechanism is waiting for a period of time to be sure system produces consistent results before making a prediction. For this purpose we decided to use a 'Time Windowing' mechanism, which is illustrated in Figure 4.3. We select a sliding time window, W, in a sequence of frames. For each frame, we get three predictions with highest probabilities from SVM. Then all results in the time window is combined and cumulative results are found. The winner is the channel, which gets the maximum cumulative result. For example in Figure 4.3, according to the given SVM results for three frames CNN has the maximum cumulative result. So we can say winner is CNN. Note that in the second frame CNN does not get the highest probability but

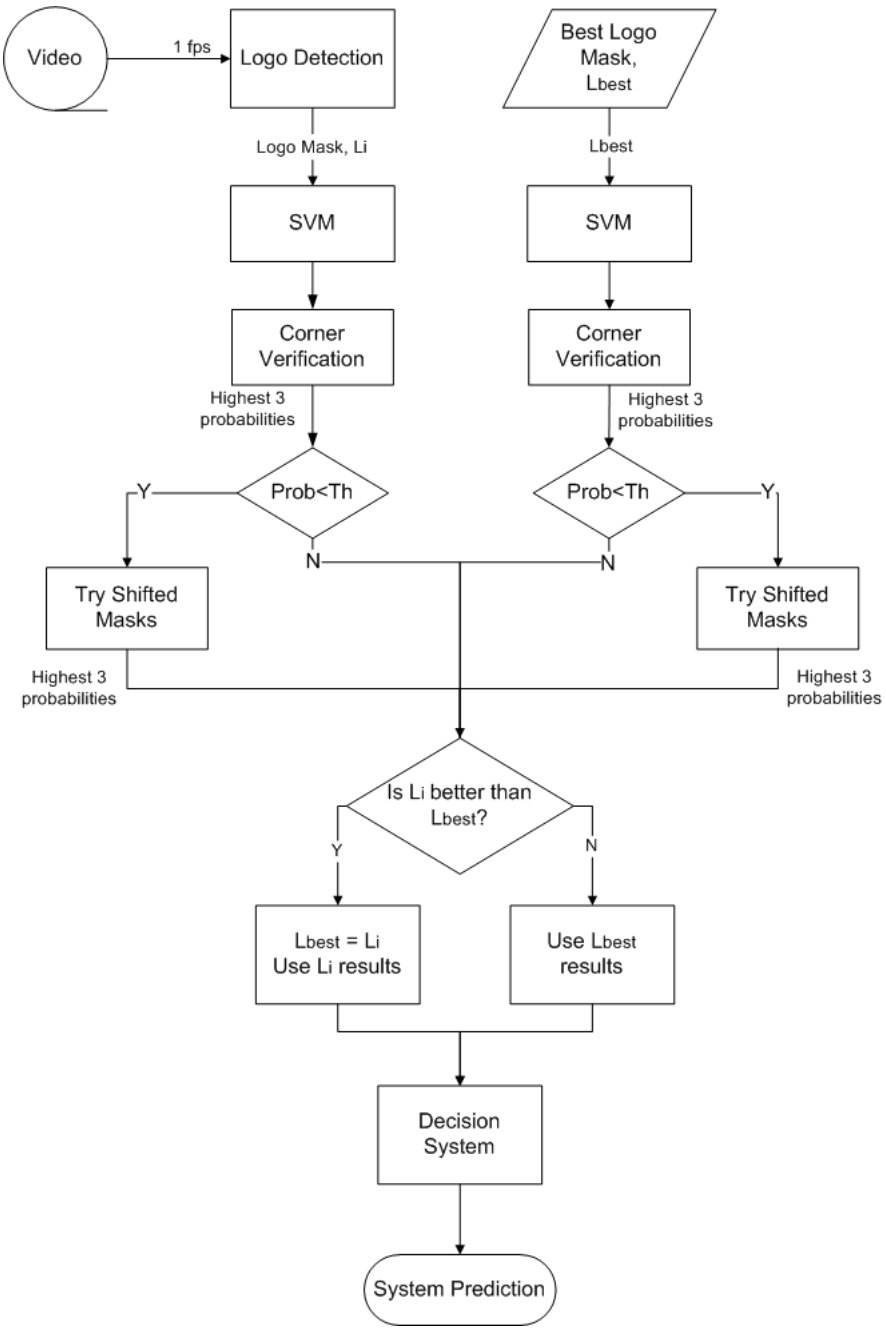


Figure 4.2. Flowchart of logo identification system

final decision is not changed due to high probabilities coming from other frames. Thus robustness is achieved.

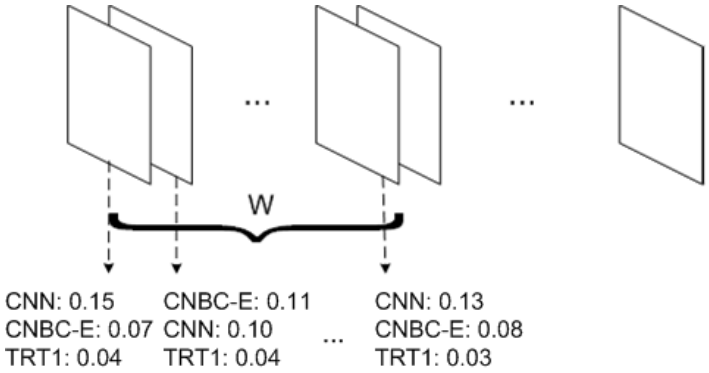


Figure 4.3. Time windowing

There is one more step after the time windowing mechanism to make the system more robust. That is, we expect the results of the time windowing system are consistent for n times consecutively. For example, while processing the frame i, we calculate the cumulative SVM probabilities of last W frames(i.e. frames $i, i-1, \ldots, i-W-1$) and predict the channel name, say CNN, as having the maximum cumulative probability. We record this as the first win of CNN, and then starts to process frame $(i+1)$. We put forward W one step, and calculate the cumulative probabilities of last W frames (i.e. frames $i+1, i, \ldots, i-W$). If CNN has maximum cumulative probability again, it reaches to two consecutive wins. And if it repeats the wins for n consecutive times without any break, the system is said to be ready to make a prediction. And this will be the system output which is shown in the last step of Figure 4.2.

The final step of the decision system is cumulative SVM probability thresholding. That is, if the cumulative SVM probability value of predicted channel is lower than a threshold, system does not make a prediction. This thresholding mechanism is used to avoid predictions of logo candidates which are not TV logos (e.g. Program name texts, program logos, or other stable contours on the frame).

5. EXPERIMENTS AND RESULTS

We can categorize our experiments into two different categories. The first experiments are on the TV logo classification. We worked on captured frames, and compared classification results of different features such as GD, ICA, PCA, NMF, and DCT on SVM to obtain the best classifier. The second set of experiments are on the complete TV logo identification system (combining TV logo detection and TV logo classification together), and we worked on broadcast videos to test the system. The details of each experiments will be given in subsequent sections.

5.1. TV Logo Classification Experiments

5.1.1. Data Set

For the TV logo classification experiments, we first constructed a logo DB that consists of logos of almost all Turkish channels and some European channels. We collected 20 sample for each TV logo, each sample has different background as in Figure 5.1. The whole database (DB) consisted of 152 channels x 20 instances = 3040 logo images. One instance for each logo in the DB is shown in Figure 5.2.

For the construction of logo DB, we utilized a TV card on a PC and a digital satellite receiver. The captured frames are in PAL BG (i.e. 720x576) format. All logos are manually cropped from the captured frames. While collecting the logo samples we tried to select samples that have different backgrounds. The different backgrounds of the same logo helps to construct a classifier which is robust against background variations. The samples with different backgrounds was given in Figure 5.1.

In this logo DB, since all samples are cropped manually and have therefore different sizes they are affected by geometric noise. The logo samples are extracted manually based on their bounding boxes, but due to the manual operation the logo sizes are not identical. The geometric noise is apparent in logo samples in Figure 5.1 in the form of

Figure 5.1. Some logo samples in logo DB. Each row illustrates six different instances of a logo with different background.

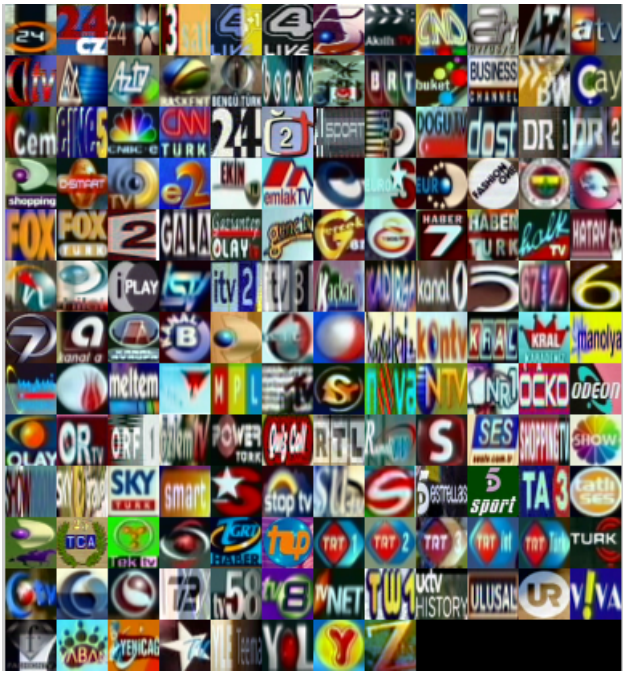


Figure 5.2. All logos in logo DB

slightly changing scale or displacement of the center. Since automatic extraction of logo ROI will also be noisy, we have not post-processed these images for exact alignment or scaling so that the classifier can be robust against these perturbations.

For each channel, out of 20 samples we used 15 images for the training phase, and 5 images for testing. We have worked on both gray-scale and rgb images, to observe the contribution of colour in logo classification.

5.1.2. Experiments

To select best feature for the logo classification problem, we perform experiments by using the features GD, PCA, NMF, ICA1, ICA2, and DCT. A snapshot from Graphical User Interface(GUI) of our application, which is written in MATLAB, can be seen in Figure 5.3. Since PCA, ICA, NMF and DCT are subspace analysis methods and takes fixed size inputs, we first normalize the size of the ROI by applying GD, which converts all logo images into fixed-size macro-pixel representation. Then subspace analysis methods can be applied. The simple illustration of this process was given in Figure 3.2 in the TV Logo Classification section. For any grid size, there is a corresponding resolution level on number of native pixels covered by the macropixel window. We select three different grid sizes such as 8x8, 16x16, and 32x32, which is shown in the left most column of Table 5.1. The native sizes of logo images that cropped manually from PAL BG frames (i.e. 720x576) change in a range of 20 pixel to 120 pixel for each dimension. For example, one of the TRT1 samples has a size of 40x102.

We perform experiments on both gray-scale and rgb images of logos. We have used multi-class SVM as the classifier. W used LibSVM [35] software package, where *one-vs-one* method is used for the multi-class classification. We have trained SVM in *linear* mode. Totally 2280 logo images are used for training (15 samples from each logo).

In the test phase, total 760 logo images are used for testing (5 samples from each logo). For each test image in the test set, one result is recorded as *true classification* or

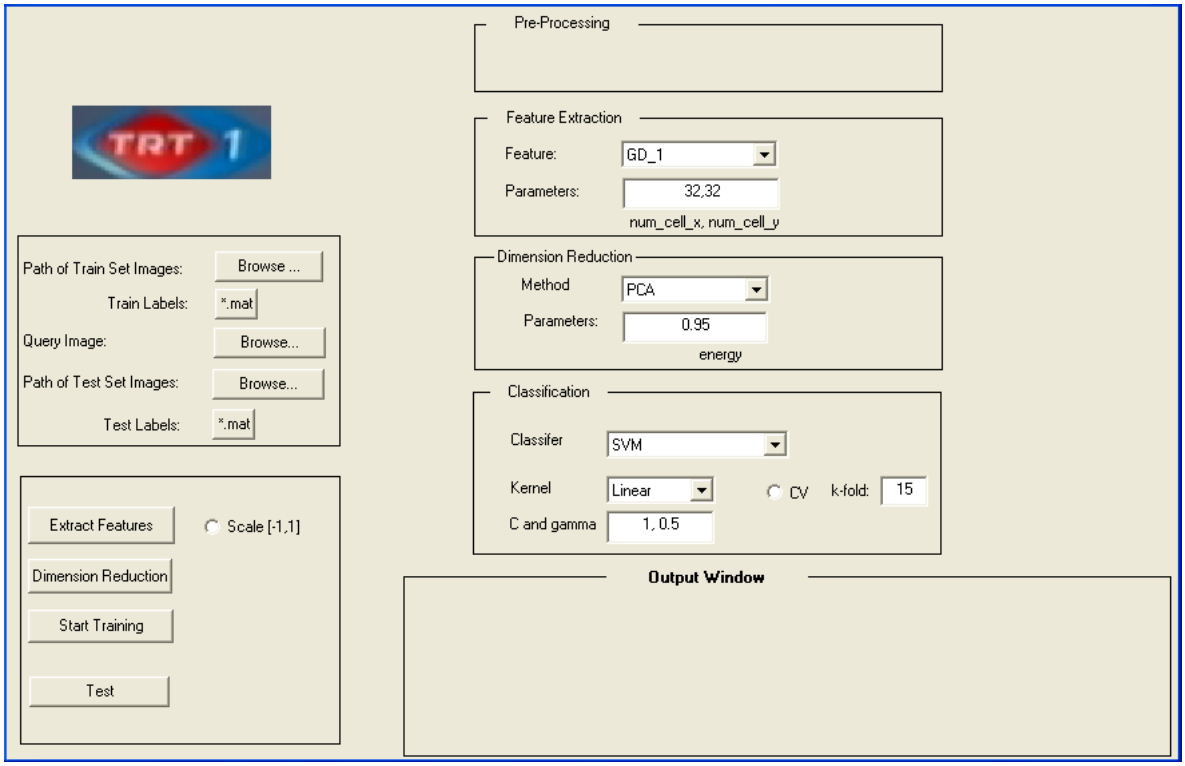


Figure 5.3. A snapshot of logo classification application GUI

false classification according to the result of SVM. *True classification* rate is calculated by the following formula

$$\frac{\#\text{ true classifications}}{\#\text{ test samples}}$$

where # test samples = 760 in our experiment. When we look at the test results in Table 5.1, we can see that **ICA2** outperforms other features with 99.21% (6 misses in 760 test samples) accuracy rate on colour logo images. According to that results, it is obvious that using a bigger grid gives better results. And we can say that 32x32 would be a good choice for grid size. The other factor that effects the result of experiment is the colour factor. We perform experiments on both gray-scale and colour images and we see that colour logo images gives better results than gray-scale logo images. For ICA2, there is nearly 2% percent difference between results of gray-scale images, and colour images for a grid size of 32x32.

Table 5.1. Classification test results for gray-scale (first 3 rows) and colour images (last 3 rows)

Image Size	GD	PCA	NMF	DCT	ICA1	ICA2
8x8	95.39	93.68	86.71	91.45	91.45	95.26
16x16	97.24	96.71	91.58	96.84	94,34	97.37
32x32	97.76	97.63	94.74	97.24	96.05	97.37
8x8	96.32	95.13	92.37	95.66	95.66	97.50
16x16	98.03	97.50	97.50	97.37	96.97	98.68
32x32	98.29	98.29	97.63	97.50	97.63	**99.21**

Table 5.2. Sizes of feature vectors for gray-scale (first 3 rows) and colour images (last 3 rows)

Image Size	GD	PCA, NMF, ICA1, and ICA2	DCT
8x8	64	27	64
16x16	256	61	128
32x32	1024	100	128
8x8	192	43	192
16x16	768	82	384
32x32	3072	121	384

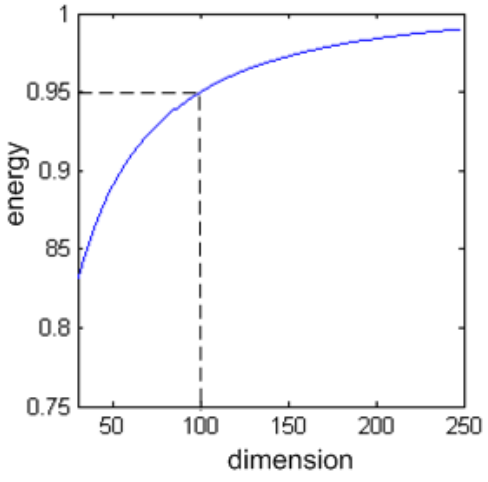


Figure 5.4. PCA energy graph for 32x32 gray-scale images

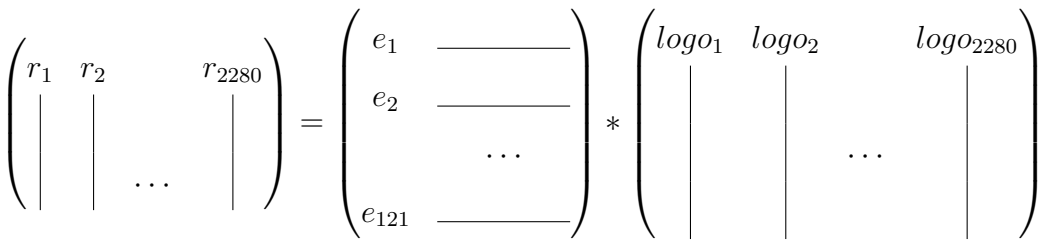


Figure 5.5. Matrix representation of PCA projection ($\mathbf{R} = \mathbf{P}_m^T \tilde{\mathbf{X}}$) for 32x32 colour images in the training phase. ($\mathbf{R}$ = 121x2280, $\mathbf{P}_m^T$ = 121x3072, $\tilde{\mathbf{X}}$ = 3072x2280).

The sizes of feature vectors are given in Table 5.2. For GD, size of feature vector is the same as size of images. For PCA we select the first m eigenvectors that include 95% energy. Cumulative energy distribution of principle components for 32x32 gray-scale images is given in Figure 5.4. The matrices used for the PCA projection in the training are illustrated in Figure 5.5, where N = 2280 (number of logo images), M = 3072 (number of macropixels), and m = 121 (number of eigenvectors corresponds to 95% of the energy). Thus, a dimension reduction from 3072 to 121 is achieved by using PCA projection. We assume size of PCA feature vectors as reference, and use the same size for ICA and NMF. Since PCA is used as a preprocessing step in ICA for dimension reduction, the ICA feature vectors are constructed in the same size as the PCA case. For the calculation NMF feature vectors we need to determine two parameters. The first parameter, 'size of feature vector' is used as the same as size of PCA feature vectors, and the second parameter is 'number of iterations = 100'. For calculation of DCT feature vectors, we used $N = 8$ (block size) and applied DCT to each 8x8 subblocks. And then, DCT feature vectors are formed by concatenating the first n_{low} low-frequency coefficients of each subblock. n_{low} is selected according to the image size, for example $n_{low} = 8$ for 32x32 images, and $n_{low} = 32$ for 16x16 images.

5.2. TV Logo Detection and Identification Experiments

As explained in the Logo Identification System section, we combine TV logo detection and TV logo classification sub systems in order to construct a TV identification system. A snapshot of our TV logo identification application GUI is shown in Figure

5.6. An 'avi' file is given as input to the system. In the lower left corner of the GUI, currently processed frame is shown. The processed corner images are shown from left to right for each step of the algorithm. The resulting ROI logo candidates are shown in upper right corner, and the prediction of the system with the cumulative SVM probability value is shown in red bold letters.

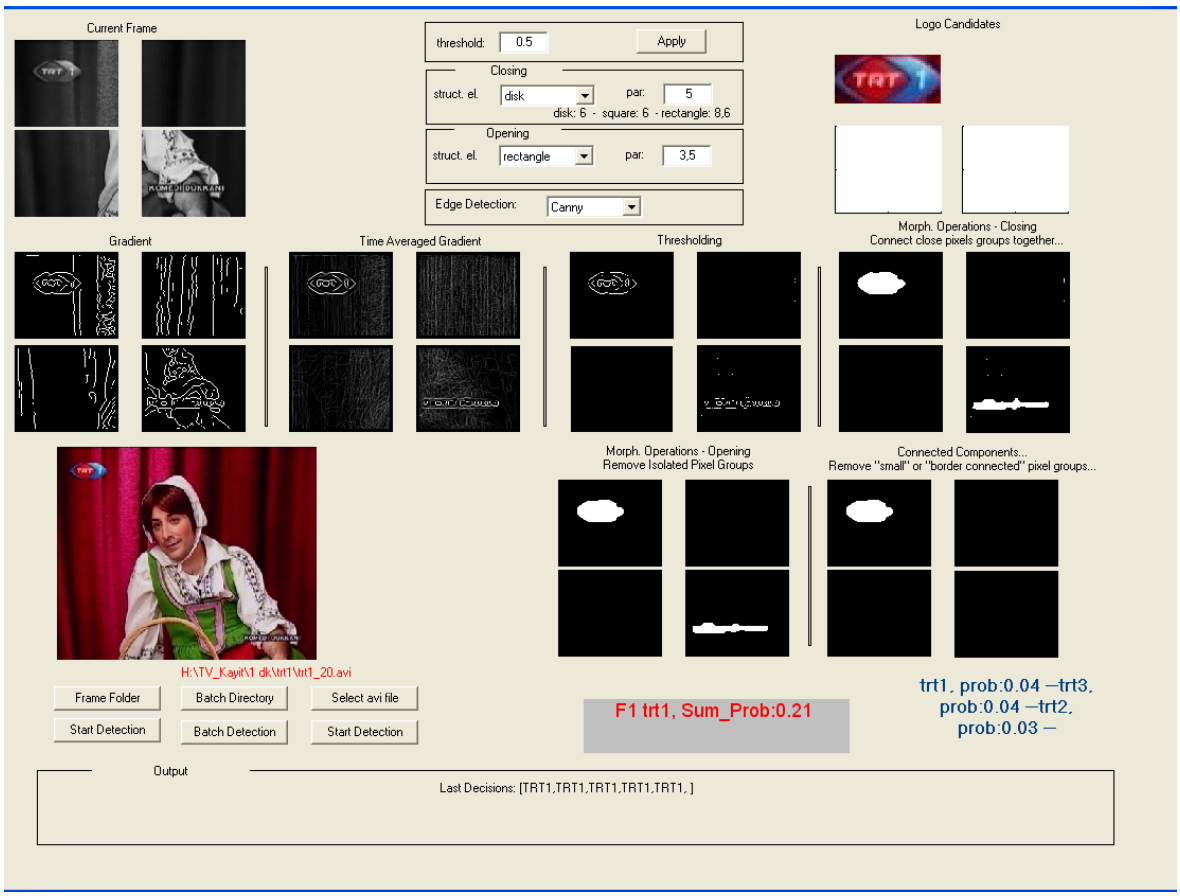


Figure 5.6. A snapshot of logo identification application GUI

5.2.1. Data Set

In order to test our logo identification system we have collected sample video sequences from 12 most popular TV channels of Turkey. We have recorded 20 sample sequences for each TV channel and created a video DB of 240 video sequences. Duration of each video is approximately 60 seconds, hence the total DB amounts to 4 hours of video. Logo identification problem becomes a more challenging problem with this short duration of sequences. Many of the works in literature perform experiments in long duration video records (e.g. 30 minutes or longer) with few number of samples. We have

kept duration of videos short enough to increase number of sample sequences to 240. Therefore we are able to apply our logo identification algorithm to a variety of video records that have different characteristics (e.g. No motion in the scene, overlapping TV logos, text and program logos in other corners, and so on).

Some of the challenging video examples are shown in Figure 5.7. Figure 5.7a shows a video scene which has almost no motion; Figure 5.7b shows a video scene which has no motion and has some static contours (i.e. texts, logos) in each corner; Figure 5.7c shows a video scene which has horizontal and vertical text lines almost connected to the TV logo; Figure 5.7d shows a video scene which has text lines connected to the TV logo; Figure 5.7e shows a video scene where both the TV logo colour and background colour are the same; Figure 5.7f shows a video scene that has a transparent TV logo on a complex background (i.e. having many edges in background); Figure 5.7g shows a video scene in which the original TV logo is overlapping with another TV logo; Figure 5.7h shows a video scene which is similar to Figure 5.7e where TV logo and background have the same colour; Figure 5.7i shows a video scene in which TV logo is connected to a text line.

We have utilized a TV card on a PC and a digital satellite receiver to record these broadcast videos. The recorded videos are in CIF format (i.e. 352x288). Some of the important properties of the videos:

- Container: AVI
- Codec: Intel Indeo (R) Video R5.10
- Frame Sampling Rate: 25 fps
- Frame Size: 352x288
- Duration: 60 seconds
- Size On Disk: 30-40 MB

In order to read recorded ‘avi’ files and convert to frames we have used dxAvi library [36].

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 5.7. Some of the challenging videos in video DB

5.2.2. Experiments

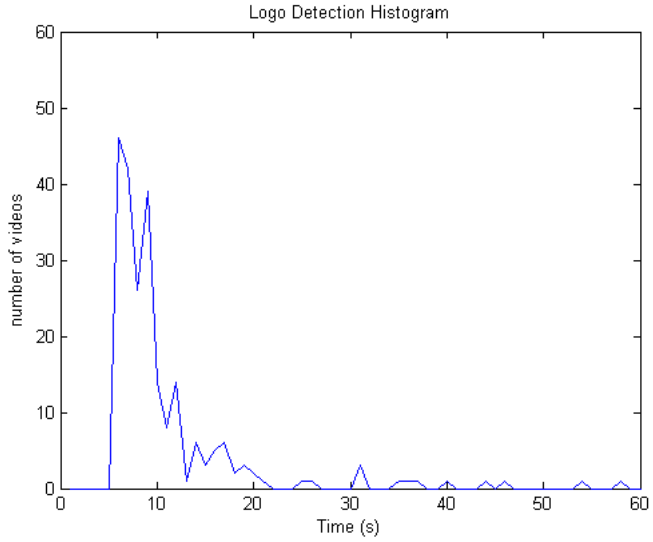
The performance evaluation of TV logo detection and TV logo identification systems in the literature is not always consistent. Some of the works [1, 8] in the literature give TV logo tracking results, some of the works do not give performance evaluation methods, and some of them do not even give information about the dataset. In this work, we try to fill these gaps in the literature. Since the time spent for the logo detection is very important we have used one minute duration videos for realistic experiments. A successful logo detection system should detect TV logos within this period of time in order to be useful for real time broadcast videos.

We evaluate both TV logo detection and TV logo identification algorithms in our experiments. We have conducted five different types of experiments. Each experiment is conducted to show contribution of each proposed method, finally the best system is achieved by adding all methods.

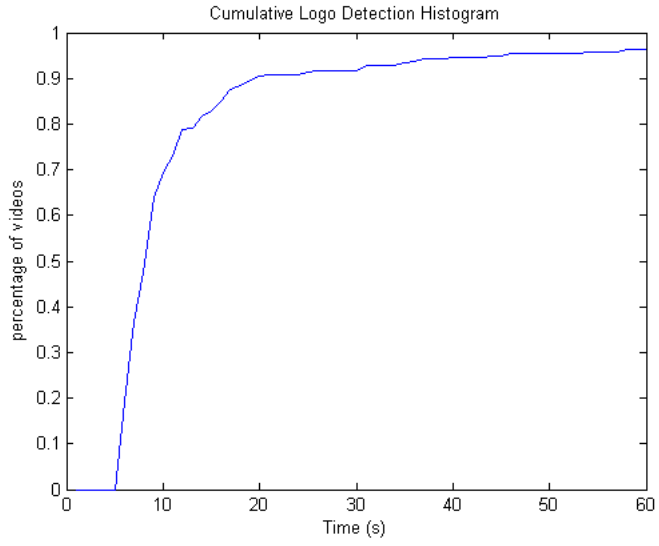
In order to measure logo detection process, we check if any logo mask is obtained in the correct corner. Detection is considered as successful if logo detection system obtains a logo mask in the expected corner of the frame, and the logo mask passes the constraints (e.g. stability constraint, shape constraints, etc.). To measure performance of the logo identification system on each video sample we used the following formula

$$\frac{\text{\# True Matches}}{\text{\# Total Predictions}} \tag{5.1}$$

5.2.2.1. Experiment-1. The first experiments are conducted with simple thresholding by using $Th = 0.5$ value. We have used a disk structuring element with radius r=5 for morphological closing, and a rectangle structuring element with width=5, height=3 for morphological opening. Transition mask method is not used in this experiment. Logo detection results for Experiment-1 are given in Figure 5.8, where Figure 5.8a



(a)



(b)

Figure 5.8. Logo detection results of experiment-1. a) Logo detection histogram, b) Cumulative logo detection histogram.

Figure 5.9. Problem of simple thresholding method. Images left to right: Time avg. edges, after thresholding, after closing, after opening, obtained logo candidate.

shows histogram of logo detection, and Figure 5.8b shows cumulative histogram of logo detection according to time. The first detection epochs of TV logos are use for logo detection histogram. Once can see that in 70% percent of videos logos are detected within ten seconds, and 90% percent of them within 20 seconds. And if we use all of the time allowed, i.e. 60 seconds, the detection rate for this experiment reaches to 96.25% (231/240), with nine missed videos.

Identification results of Experiment-1 are shown in Table 5.3. Each cell of the table includes identification performance of the proposed system on a video record according to the evaluation criteria given in Eq. 5.1. The average accuracy rate for Experiment-1 is 85.67%. One of the typical problems of simple thresholding method is shown in Figure 5.9. Edges of logo image are not obtained correctly in thresholding step, and broken edges lead to deformation in morphological opening step, thus ideal logo mask can not be extracted. Note that, if the edges were obtained without any broken parts, hole filling operation would protect the logo mask from the deformation in opening step. This example is captured from record 11 of ATV. The same problem is also seen very frequently in STAR videos.

5.2.2.2. Experiment-2. In order to alleviate the thresholding problems as illustrated in Figure 5.9, we perform experiments using hysteresis thresholding method. In this method two levels of thresholds are applied, namely high threshold and low threshold. We use $Th_{low} = 0.35$ and $Th_{high} = 0.5$ for this experiment. We have used the same structuring elements as the ones we used in Experiment-1 (Closing: disk, r=5; Opening: rectangle, w=5, h=3). Logo detection results for Experiment-2 are given in Figure 5.10. Detection results are similar to the results of the previous experiment with a small

Table 5.3. Identification results of experiment-1

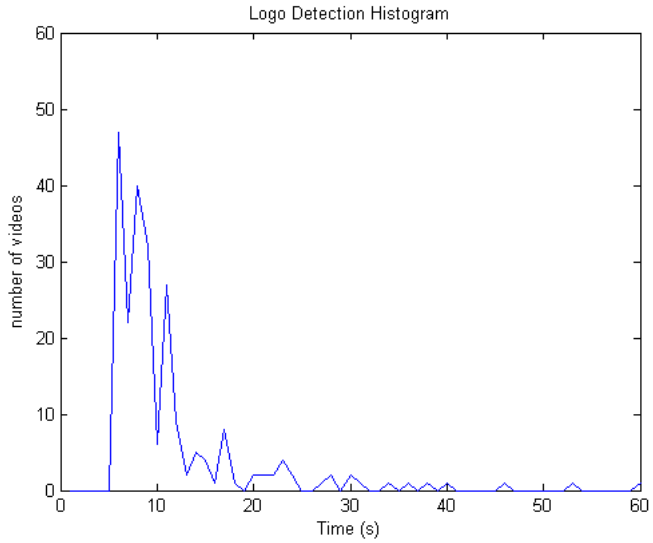
	atv	cine5	cnbce	cnnturk	ch. d	fox	ch. 7	ntv	show	star	trt1	tv8
record 1	0.84	0.92	1	0.96	0.45	0.62	1	1	1	0	1	1
record 2	1	1	1	1	0.86	0.36	1	1	1	0.74	1	1
record 3	0.97	1	1	1	1	1	1	1	1	1	0.25	1
record 4	1	1	1	1	1	0.42	1	0.18	1	0.76	1	1
record 5	0.95	1	1	0.79	1	1	1	1	1	0.79	1	1
record 6	0.98	1	1	1	1	0.95	0	0	1	1	0	1
record 7	1	1	1	0	0	1	1	1	1	1	0.9	1
record 8	1	1	1	0	1	1	0.98	1	1	0.31	1	1
record 9	1	1	1	1	1	1	1	1	1	1	0.5	1
record 10	1	0.6	1	0.68	1	1	1	0	0	0	0.92	1
record 11	0	1	0.83	1	1	1	1	1	1	0.98	1	0.63
record 12	1	1	1	0.08	1	1	1	1	1	1	1	1
record 13	1	1	0.94	0.64	1	1	1	0	1	1	1	1
record 14	1	1	1	1	1	1	1	1	1	1	0.6	1
record 15	1	1	1	1	1	1	1	0	1	1	0.9	1
record 16	1	1	1	1	1	1	0.94	0	1	0.45	0.95	1
record 17	1	0.85	1	0	0.96	1	1	1	1	0.22	0	1
record 18	0.93	1	1	1	1	1	1	1	1	0.55	0.89	0.93
record 19	1	0.6	1	1	1	1	1	0	0.88	1	0.14	1
record 20	0.93	0.36	1	0	0.96	0	0.86	0	0.95	1	1	1
Avg:	0.9300	0.9165	0.9885	0.7075	0.9115	0.8675	0.9390	0.6090	0.9415	0.7400	0.7525	0.9780
Overall Avg: 0.8567												

decrement in maximum detection rate, which is 95.42% (229/240) for this experiment.

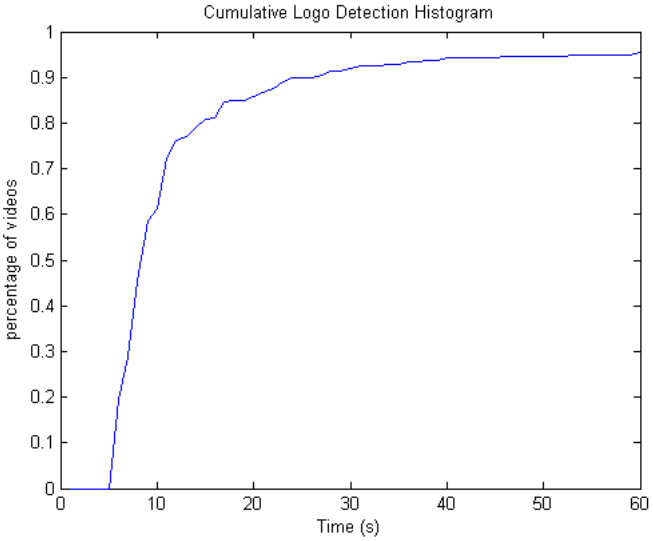
Identification results of Experiment-2 are given in Table 5.4. By using hysteresis thresholding the average accuracy rate is increased from 85.67% to 86.34%. Note that record 11 of atv is increased from 0 to 0.96. From the results, it is clear that something goes wrong for NTV videos due to very low accuracy rates. The main problem with NTV videos is structure the of channel logo. There is a text band under the logo and most of the time this band leads to deformation of logo mask in the closing step. A typical example of that overclosing problem is given in Figure 5.11. We will try to find a solution to that problem in Experiment-4 by introducing adaptive structuring element method.

5.2.2.3. Experiment-3. In the third experiments, in order to combat small corruptions in logo masks we considered new logo masks by shifting, enlarging, and shrinking the original one by 2 pixels in different directions (i.e. Up, Down, Left, Right, etc.). Consequently, we obtain 18 different logo masks. This method is expected to improve classification performance. Since the other experiment parameters are the same as the ones used in Experiment-2 (Thresholding: hysteresis, [0.35 0.50]; Closing: disk, r=5; Opening: rectangle w=5, h=3) the logo detection result is the same as the one in Experiment-2. Logo identification results of Experiment-3 are shown in Table 5.5. However, the average accuracy rate is surprisingly decreased from 86.34% to 83.09%. The main reason of the decrement is the low accuracy rate of TRT1 which decreases to 34.1% from 81.35%. With the mask perturbation method, most of the SVM predictions of TRT1 videos becomes to TRTInt. Figure 5.12 shows the TRT logos which are very similar to each other. They share the SVM probabilities because of the similarity. With the mask perturbation method, TRTInt always gets higher SVM probability as compared to TRT1.

5.2.2.4. Experiment-4. In the fourth experiment we use adaptive structuring element method. In this method there are two different groups of structuring elements for morphological operations, namely, big and small structuring elements. For each cor-



(a)



(b)

Figure 5.10. Logo detection results of experiment-2. a) Logo detection histogram, b) Cumulative logo detection histogram.

Figure 5.11. NTV overclosing problem. Images left to right: Corner region of frame, after thresholding, after morphological operations.

Table 5.4. Identification results of experiment-2

	atv	cine5	cnbce	cnnturk	ch. d	fox	ch. 7	ntv	show	star	trt1	tv8
record 1	1	0.65	1	1	1	1	1	1	1	0.13	1	1
record 2	1	1	1	1	1	1	1	1	1	0.65	1	1
record 3	0.94	1	1	1	1	1	1	1	1	1	0	1
record 4	1	0.88	1	0.8	1	0.97	1	0.04	1	1	1	1
record 5	0.95	1	1	1	1	1	1	0	1	0.75	1	1
record 6	1	1	1	1	1	1	0	0	1	1	0	1
record 7	1	1	1	0	0	1	1	1	1	1	0.67	1
record 8	0.61	1	1	0	1	1	1	1	1	0	1	1
record 9	1	1	1	1	1	1	1	1	1	1	1	0.8
record 10	1	0.25	0.94	1	1	1	1	0	0	0.48	0.97	1
record 11	0.96	0.98	0.79	0.98	1	1	1	1	1	0.94	1	0
record 12	1	0.9	1	0.12	1	0.21	1	0	1	1	1	1
record 13	1	0.89	1	0.8	1	1	1	0.13	1	1	1	1
record 14	1	1	1	1	1	1	1	1	1	1	0.88	1
record 15	1	1	1	1	0.98	1	0.98	0	0.97	1	0.75	1
record 16	1	1	1	1	1	1	0.97	0	1	1	1	1
record 17	0	0.97	1	0	1	1	1	1	1	1	0.88	1
record 18	1	1	1	0.97	1	1	1	1	1	1	0.8	0.96
record 19	1	0.92	1	0.98	1	1	1	0	0.67	1	0.97	1
record 20	0.84	0.52	1	0	0.98	0	0.78	0	0.92	1	0.35	1
Avg:	0.9150	0.8980	0.9865	0.7325	0.9480	0.9090	0.9365	0.5085	0.9280	0.8475	0.8135	0.9380
Overall Avg: 0.8634												

Table 5.5. Identification results of experiment-3

	atv	cine5	cnbce	cnnturk	ch. d	fox	ch. 7	ntv	show	star	trt1	tv8
record 1	1	1	1	1	1	0.96	1	1	1	0.13	1	1
record 2	1	1	1	0.97	0.98	1	1	1	1	0.7	1	1
record 3	0.95	1	1	1	1	1	1	1	1	1	0	1
record 4	1	1	1	1	1	1	1	0.05	1	1	0.82	1
record 5	0.95	1	1	1	1	1	1	0	1	0.74	1	1
record 6	0.95	0.98	1	1	1	1	0	0	1	1	0	1
record 7	1	1	1	0	0	1	1	1	1	1	0	1
record 8	0.91	1	1	0	1	1	0.95	0.24	1	0	0	1
record 9	1	1	1	0.98	1	1	1	1	1	1	0	1
record 10	1	0	1	1	1	1	1	0	0	0.76	0	1
record 11	1	1	0.82	0.98	1	1	1	1	0.98	0.98	1	0
record 12	1	0.93	1	0.05	1	1	1	0	1	0.91	1	1
record 13	1	0.89	1	0.97	1	1	1	0	1	1	1	1
record 14	1	1	1	1	1	0.76	1	1	1	1	0	1
record 15	1	1	1	1	1	1	0.98	0	0.97	1	0	1
record 16	1	1	1	1	1	1	0.65	0	1	1	0	1
record 17	1	1	1	0	1	1	1	1	1	1	0	1
record 18	1	1	1	0.97	1	1	1	1	1	0.89	0	0.96
record 19	1	1	1	1	1	1	1	0	0.86	0.98	0	1
record 20	0.88	0.74	1	0	0.98	0	0.64	0	0.64	1	0	1
Avg:	0.9820	0.9270	0.9910	0.7460	0.9480	0.9360	0.9110	0.4645	0.9225	0.8545	0.3410	0.9480
Overall Avg: 0.8309												

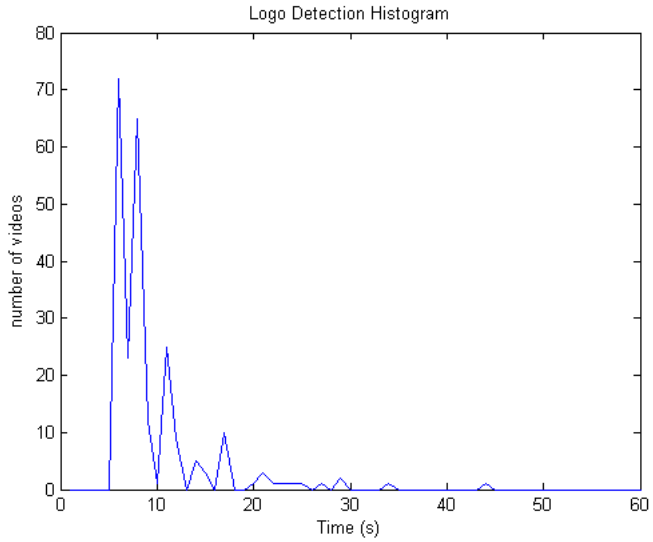
Figure 5.12. Similar TRT logos

ner, bigger structuring elements are used in the first iteration, and if no logo mask is obtained, then smaller structuring elements are applied in the second iteration. With this method we expect to solve the problems as shown in Figure 5.11. The structuring elements that we used in the experiment are

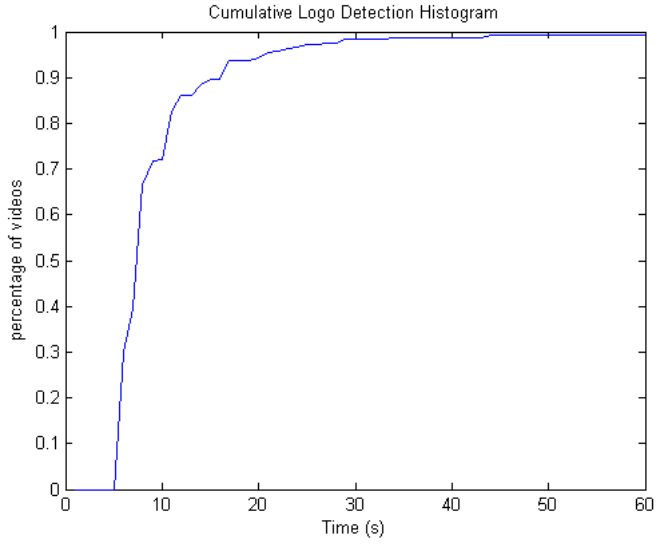
- Big Structuring Elements:
  - Closing: disk, r=5
  - Opening: rectangle, w=5, h=3
- Small Structuring Elements:
  - Closing: disk, r=3
  - Opening: square, a=3

Logo detection results of Experiment-4 are given in Figure 5.13. With the adaptive structuring element we reach to a maximum detection rate of **99.17%** (238/240) with two missed videos. In Figure 5.14 comparision of logo detection results for all experiments are shown. Note that since Experiment-3 has the same results with Experiment-2, and Experiment-5 has the same results with Experiment-4 their results are not included in the graph.

Logo identification results of Experiment-4 are given in Table 5.6. It is clear that there is a big increase in the NTV results(from 46.45% to 83.65%). The problem with NTV videos was previously depicted in Figure 5.11. With the use of adaptive structuring element method this problem is fixed. In the first iteration big structuring element does not obtain a logo mask for NTV logos but in the second iteration small structuring element extracts the logo mask. The average accuracy rate for Experiment-4 is 86.32%.



(a)



(b)

Figure 5.13. Logo detection results of experiment-4. a) Logo detection histogram, b) Cumulative logo detection histogram.

Table 5.6. Identification results of experiment-4

	atv	cine5	cnbce	cnnturk	ch. d	fox	ch. 7	ntv	show	star	trt1	tv8
record 1	1	0.98	1	1	0.98	0.89	1	1	1	0.17	1	1
record 2	0.87	1	1	1	1	1	1	1	1	0.7	0.97	1
record 3	0.04	1	1	1	1	1	0	1	1	1	0.23	1
record 4	1	1	1	1	1	1	0.74	0.16	1	1	0.82	1
record 5	0.91	1	1	1	1	1	1	0.97	1	0.91	1	1
record 6	1	0.98	1	1	1	1	0	1	1	1	0	1
record 7	1	1	1	0	1	1	1	1	1	1	0	1
record 8	1	1	1	1	1	1	0.93	0.68	1	0.29	0	1
record 9	1	1	1	1	1	1	1	0.92	1	0.98	0	1
record 10	1	0.67	1	1	1	1	1	1	0	0.9	0	1
record 11	0.76	1	0.82	0.98	1	1	1	1	0.98	1	1	0.22
record 12	1	0.9	1	0.97	1	1	1	1	0.98	0.89	1	1
record 13	1	0.91	0.96	1	1	1	0.98	0	0	1	1	1
record 14	1	1	1	1	1	0.56	0.86	1	1	1	0	1
record 15	1	1	1	0.98	1	1	1	0	0.97	1	0	1
record 16	1	1	1	1	1	1	0.65	1	1	1	0	1
record 17	1	1	1	0	1	1	1	1	1	1	0	0.95
record 18	1	1	1	0.97	1	1	1	1	1	0.98	0	0.89
record 19	1	0.97	1	1	1	1	1	1	0.78	1	0	1
record 20	0.95	0.43	1	0	0.98	0	0.79	1	0.42	1	0	1
Avg:	0.9265	0.9420	0.9890	0.8450	0.9980	0.9225	0.8475	0.8365	0.8565	0.8910	0.3510	0.9530
Overall Avg: 0.8632												

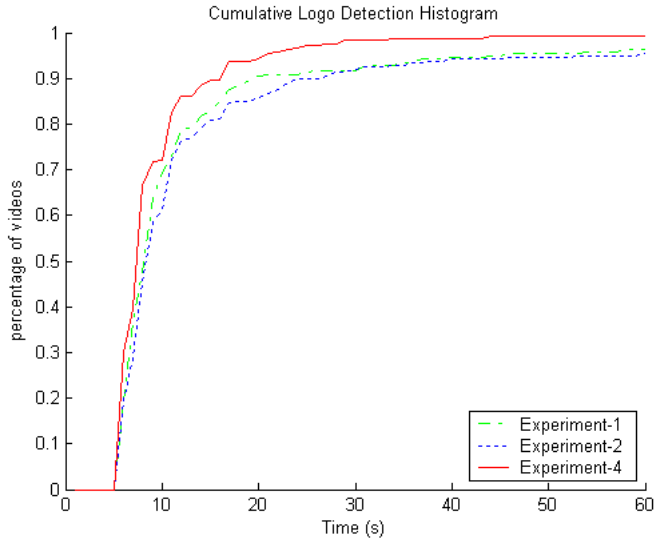


Figure 5.14. Comparison of logo detection results

5.2.2.5. Experiment-5. In this experiment a corner verification method is added to the system in order to decrease false SVM predictions. In this method, prediction of SVM is checked with corner information for consistency. Since expected corner is known for each channel (ground truth) we can easily check consistency of the SVM predictions. For example, TRT1 logos always appear in the upper left corner of frame, if SVM makes a TRT1 prediction for a logo candidate in one of the other three corners, that TRT1 prediction will not be taken into consideration.

Since there is no change in logo detection system, logo detection results of Experiment-5 is the same as Experiment-4. Logo identification results of Experiment-5 are shown in Table 5.7. Note that as explained before, mask perturbation method leads to an increment of TRTInt predictions in TRT1 videos. Since the logos are not totally different, we make an assumption that TRTInt predictions are also correct. The average accuracy rate for Experiment-5 is 94.03%. Logo identification results of all experiments are shown in Figure 5.15. Notice that the increase in the results of CNNTURK and NTV videos with the use of adaptive structuring element (Experiment-4). And the best performance is achieved in the Experiment-5 by the addition of corner verification to the system.

So far we do not apply any cumulative SVM probability threshold to the identifi-

Table 5.7. Identification results of experiment-5

	atv	cine5	cnbce	cnnturk	ch. d	fox	ch. 7	ntv	show	star	trt1	tv8
record 1	1	0.98	1	0.95	1	0.97	1	1	1	0.36	1	1
record 2	0.82	1	1	1	1	1	1	1	1	0.73	1	1
record 3	1	1	1	1	1	1	0.98	1	1	1	1	1
record 4	0.95	1	1	1	1	0.91	1	0.29	1	1	1	1
record 5	0.95	1	1	1	1	1	1	0.97	1	1	1	1
record 6	1	1	1	0.98	1	1	0	1	1	1	0.86	1
record 7	1	1	1	0.03	1	1	1	1	1	1	0.93	1
record 8	0.91	1	1	1	1	1	0.93	1	1	0.36	1	1
record 9	1	1	1	1	1	1	1	0.96	1	1	1	1
record 10	1	1	1	1	1	1	1	1	0	0.97	1	1
record 11	1	1	0.61	1	1	1	1	1	0.98	1	1	0.64
record 12	1	0.94	1	1	1	1	1	0.86	1	0.8	1	1
record 13	1	0.95	0.96	1	1	1	1	0.35	1	1	1	1
record 14	1	1	1	1	1	0.89	1	1	1	1	0.9	1
record 15	1	1	1	1	0.96	1	1	0	0.9	1	0.95	1
record 16	1	1	1	1	1	1	0.65	1	1	1	1	1
record 17	1	1	1	0	1	1	1	1	1	1	1	0.92
record 18	1	1	1	0.95	1	1	1	1	1	0.98	0.67	0.84
record 19	1	1	1	0.93	1	1	1	1	0.87	1	0.95	1
record 20	1	0.73	1	0	1	0	0.77	1	0.93	1	1	1
Avg:	0.9815	0.9800	0.9785	0.8420	0.9980	0.9385	0.9165	0.8715	0.9340	0.9100	0.9630	0.9700
Overall Avg: 0.9403												

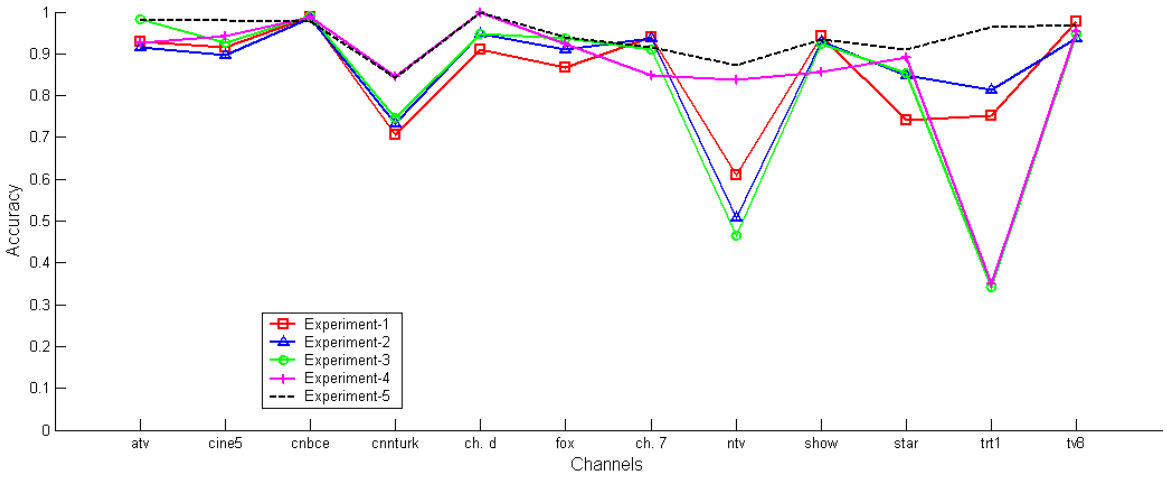


Figure 5.15. Comparison of logo identification results

cation results. As explained previously in section 4.1 Decision System, SVM probability values of W frames are added up and a cumulative SVM probability value is obtained. The one that has the highest cumulative SVM probability value for n consecutive times declared as the prediction of the system. In order to decrease false predictions we apply a cumulative SVM probability threshold, that is, logo identification system makes a prediction only if cumulative SVM probability is higher than a certain threshold value. This thresholding helps to suppress false predictions that have low cumulative SVM probability values. The cumulative SVM probability histograms for each TV channel are given in Figure 5.16. TM stands for True Match, and FM stands for False Match. In the figure, the low cumulative SVM probability values for TRT1 videos can be recognized easily. The reason of the low cumulative SVM probability values is the existence of many similar TRT logos in the data set as shown in Figure 5.12. Thus TRT logos shares SVM probabilities and each one gets a low probability value from SVM. Another interesting plot belongs to TV8, that is, cumulative SVM probability values for TV8 is very high, and most of the predictions has cumulative SVM probabilities higher than 0.8 (which is not included in the plot).

Figure 5.17 shows cumulative probability values for TM and FM results of all TV channels. With a proper cumulative SVM probability threshold value we aim to increase TM rate by reducing number of false matches. The average accuracy rates for different cumulative SVM probability threshold values are shown in Figure 5.18. The

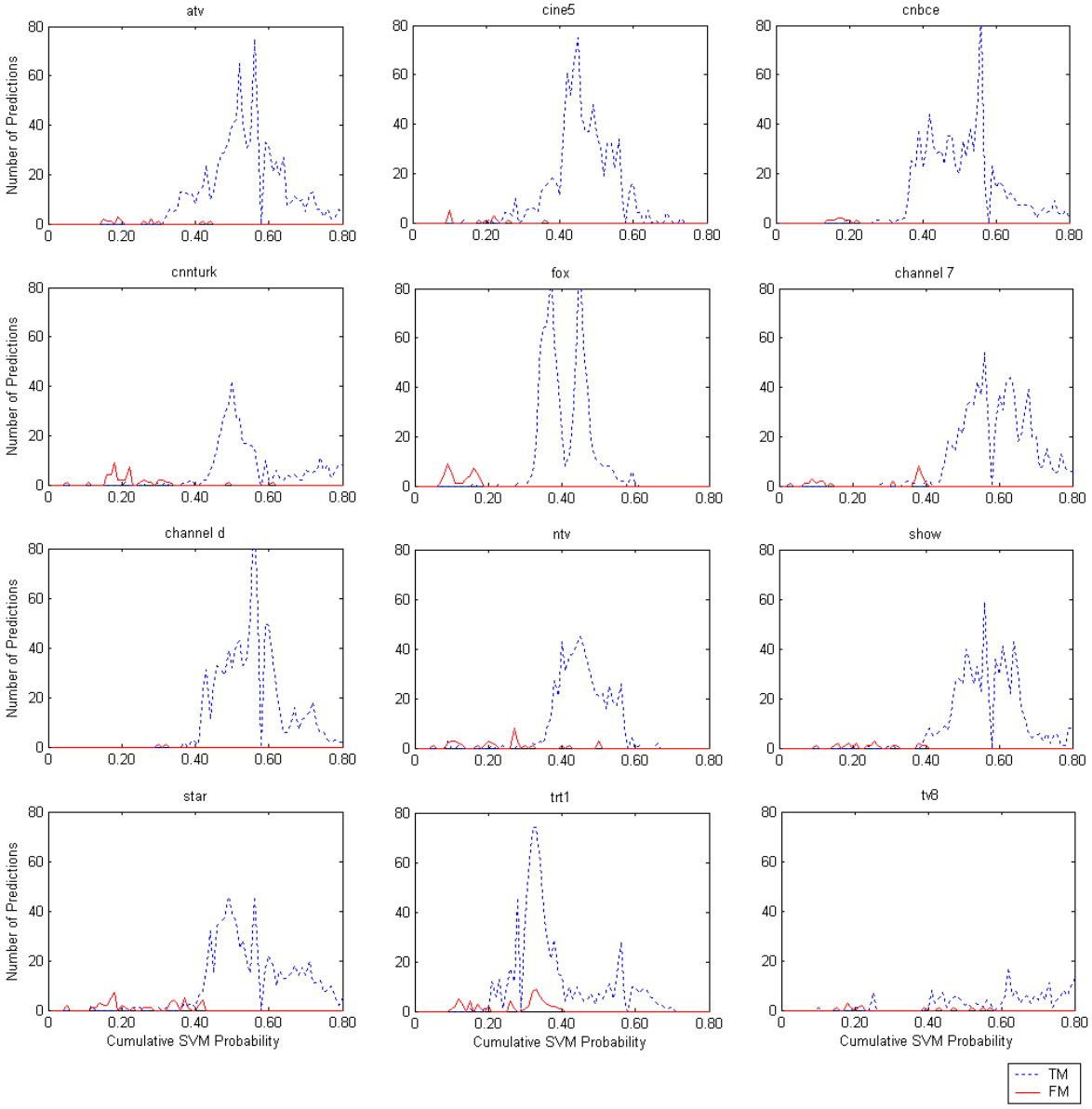
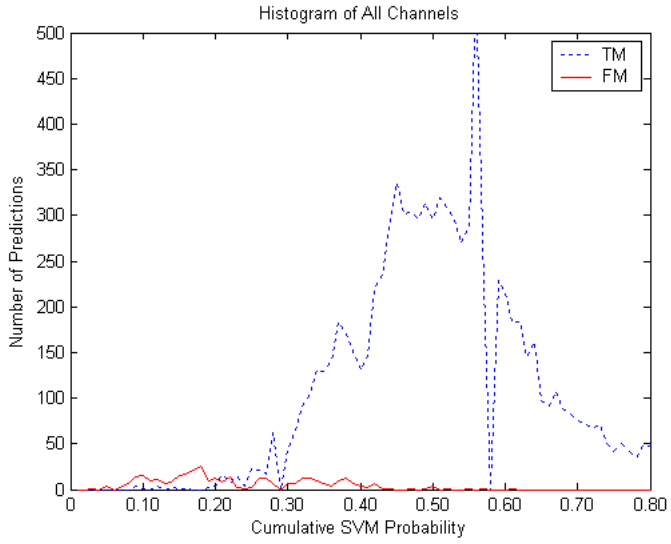


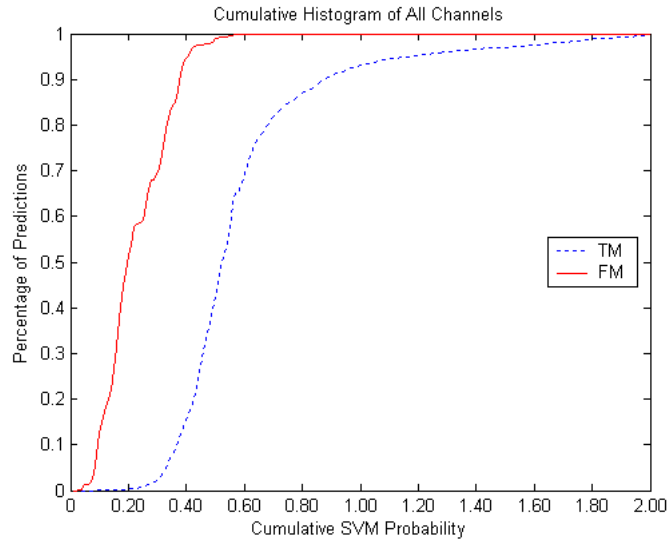
Figure 5.16. Cumulative SVM probability histograms for each TV channel

highest accuracy value (**96.03%**) is obtained by using a cumulative SVM probability value of 0.30. Identification results after cumulative SVM probability thresholding with the threshold value of 0.30 is shown in Table 5.8. Logo identification results of before and after cumulative SVM probability thresholding are given in Figure 5.19.

In Figure 5.20, some video examples when our algorithm fails are shown. Figure 5.20a shows a video example that has almost no motion (two scenes in the whole video) and has very complex background (forest). On the complex background TV logo edges are not extracted as expected and due to the static behavior of the video,



(a)



(b)

Figure 5.17. Cumulative SVM probability histograms for all TV channels a)Histogram of cumulative SVM probability for TM and FM results, b) Cumulative histogram of cumulative SVM probability for TM and FM results.

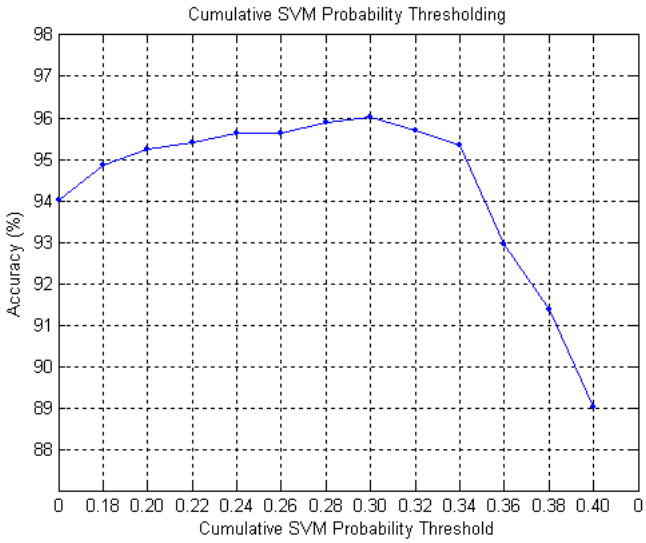


Figure 5.18. Avg. accuracy rates for different cumulative SVM prob. thresholds

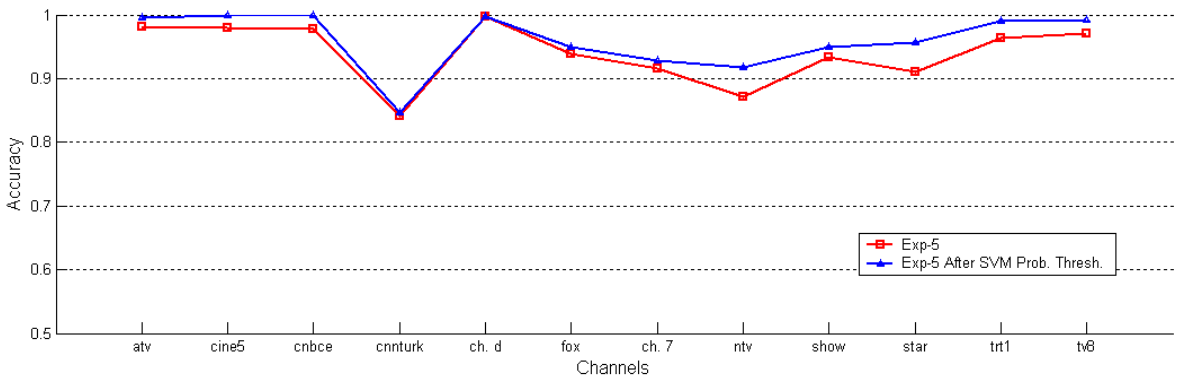
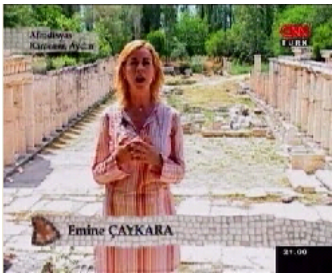


Figure 5.19. Results of experiment-5. Before and after cumulative SVM prob. thresholding

Table 5.8. Identification results of experiment-5 after cumulative SVM probability thresholding

	atv	cine5	cnbce	cnnturk	ch. d	fox	ch. 7	ntv	show	star	trt1	tv8
record 1	1	0.98	1	1	1	1	1	1	1	0.37	1	1
record 2	1	1	1	1	1	1	1	1	1	1	1	1
record 3	1	1	1	1	1	1	1	1	1	1	1	1
record 4	0.95	1	1	1	1	1	1	0.36	1	1	1	1
record 5	1	1	1	1	1	1	1	1	1	1	1	1
record 6	1	1	1	1	1	1	0	1	1	1	0.94	1
record 7	1	1	1	0	1	1	1	1	1	1	1	1
record 8	0.97	1	1	1	1	1	0.93	1	1	1	1	1
record 9	1	1	1	1	1	1	1	1	1	1	1	1
record 10	1	1	1	1	1	1	1	1	0	0.97	1	1
record 11	1	1	1	1	1	1	1	1	0.98	1	1	1
record 12	1	1	1	1	1	1	1	1	1	0.8	1	1
record 13	1	1	1	1	1	1	1	1	1	1	1	1
record 14	1	1	1	1	1	1	1	1	1	1	0.87	1
record 15	1	1	1	1	0.96	1	1	0	1	1	1	1
record 16	1	1	1	1	1	1	0.65	1	1	1	1	1
record 17	1	1	1	0	1	1	1	1	1	1	1	1
record 18	1	1	1	1	1	1	1	1	1	0.98	1	0.84
record 19	1	1	1	0.93	1	1	1	1	1	1	1	1
record 20	1	1	1	0	1	0	1	1	1	1	1	1
Avg:	0.9960	0.9990	1.00	0.8465	0.9980	0.9500	0.9290	0.9180	0.9490	0.9560	0.9905	0.9920
Overall Avg: 0.9603												



(a) Cnnturk17

(b) Fox20

(c) Show10

(d) Details of Cnnturk17

(e) Details of Fox20

(f) Details of Show10

Figure 5.20. Images a) to c) show video examples when our algorithm fails. Images d) to f) show output of each step in algorithm for each video example(Images left to right: Corner region of frame, after thresholding, after morphological operations).

time averaged edges does not change by time. Figure 5.20b shows a video example where a text band is inserted near to the TV logo in the video. The edges of text band are connected to the edges of logo and ideal logo mask is not obtained. Figure 5.20c shows another example where some static texts are inserted close to the TV logo in the video. Consequently, logo mask is not obtained accurately due to those static texts.

6. CONCLUSIONS

In this study, we have developed a fully automatic TV logo identification system. In the first part of the work, we have concentrated on detection of logo images in a sequence of frames. In the second part, logo classification is handled. We have used SVM in multiclass classification and compared performances of some well known subspace analysis techniques such as PCA, NMF, ICA, and DCT to find the best feature to describe TV logos. GD method is used as preprocessing step before applying the subspace analysis methods to obtain a fixed size representation of logo images.

Most of the previous works in the literature were on the logo detection, in this work we add logo classification part on the top of logo detection part and combine them to obtain a logo identification system. In the logo detection, we use time averaged edges method with some important additions such as hysteresis thresholding and adaptive structuring element method. In the logo identification system, mask transition method is used by obtaining new logo masks from current logo mask in order to increase SVM scores. And to increase robustness, a decision system which uses time windowing mechanism is designed.

Two different categories of experiments are performed in this work. First experiments are performed to find the best feature describing TV logos. For this purpose, we have constructed a logo DB of 3040 images from 152 different Turkish and European TV channels. In these experiments, ICA2 outperforms other features with an accuracy rate of 99.21% (754/760). Second group of experiments are conducted to test performance of the logo detection and identification system. We have collected 240 one minute duration broadcast videos from most popular 12 TV channels of Turkey. And the proposed system achieves to 99.17% (238/240) logo detection rate and 96.03% average accuracy rate for the logo identification.

6.1. Future Work

We have used time averaged edges to detect logos in a video sequence. In the case of no motion and complex background in the sequence, a secondary method can be applied to detect logos. For example "logoness" features can be used; i.e. there must be commonality of features that separate all logos from any non-logo program video. An ANN or SVM can be trained with two classes, i.e. positive and negative examples, and corner regions can be scanned to detect regions that have logoness properties.

We have tested our system on a variety of broadcast videos and achieved very encouraging results. In order to cross-validate the system performance, it can be tested on a totally different set of broadcast videos.

In the proposed logo detection algorithm, frames are processed sequentially from start to end. As an alternative method, logo detection can be performed by starting logo search process at arbitrary points in the video and wrapping around the sequences. Therefore, logo detection process can be accomplished in a less amount of time.

APPENDIX A: LIST OF TV CHANNEL LOGOS

Table A.1: TV channel logos in logo DB

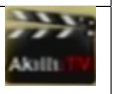
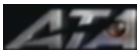
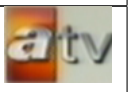
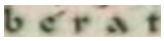
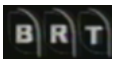
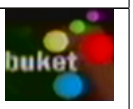
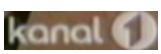
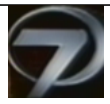
#	Name	Type	Logo	#	Name	Type	Logo
1	24	anim.		2	24cz	opaq.	
3	24n	transp.		4	3sat	opaq.	
5	4+1live	transp.		6	4live	transp.	
7	5telec.	transp.		8	akilli	anim.	
9	ans	opaq.		10	art	opaq.	
11	ata	opaq.		12	atv	opaq.	
13	atvcan.	opaq.		14	ayna	opaq.	
15	azadazer.	opaq.		16	azeritv	opaq.	
17	baskent	opaq.		18	bengutu.	anim.	
19	berat	opaq.		20	bjk	opaq.	
21	brt	opaq.		22	buket	opaq.	
23	business	opaq.		24	bw	opaq.	
25	cay	opaq.		26	cem	opaq.	
27	ch. 1	transp.		28	ch. 5	opaq.	
29	ch. 67	opaq.		30	ch. 6	opaq.	
31	ch. 7	opaq.		32	ch. a	opaq.	

Table A.1 – Continued

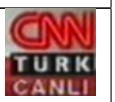
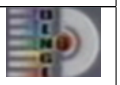
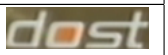
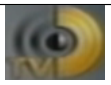
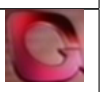
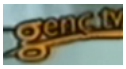
#	Name	Type	Logo	#	Name	Type	Logo
33	ch. av.	opaq.		34	ch. b	opaq.	
35	ch. d	opaq.		36	ch. t	opaq.	
37	cine5	transp.		38	cnbce	opaq.	
39	cnnturk	opaq.		40	cnnturkc.	opaq.	
41	ct24	transp.		42	ct2	opaq.	
43	ctsport	transp.		44	dingi	opaq.	
45	dogu	opaq.		46	dost	opaq.	
47	dr1	transp.		48	dr2	transp.	
49	dshopp.	opaq.		50	dsmart	opaq.	
51	duzgun	opaq.		52	e2	opaq.	
53	ekin	opaq.		54	emlak	opaq.	
55	etv	opaq.		56	euroStar	opaq.	
57	eurod	opaq.		58	fashion	opaq.	
59	fb	opaq.		60	flash	opaq.	
61	fox	opaq.		62	foxcan.	opaq.	
63	foxturk	opaq.		64	foxyeni	opaq.	
65	france2	transp.		66	gala	opaq.	
67	gantepo.	opaq.		68	genc	opaq.	

Table A.1 – Continued

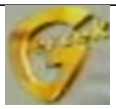
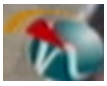
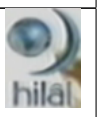
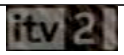
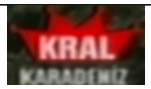
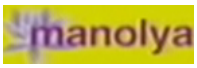
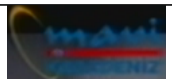
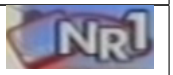
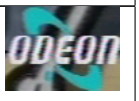
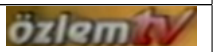
#	Name	Type	Logo	#	Name	Type	Logo
69	gercek	opaq.		70	gs	opaq.	
71	haber7	opaq.		72	habert.	anim.	
73	halk	opaq.		74	hatay	opaq.	
75	hayat	opaq.		76	hilal	opaq.	
77	iplay	opaq.		78	ist	opaq.	
79	itv2	transp.		80	itv3	transp.	
81	kackar	opaq.		82	kadirga	opaq.	
83	kanalt.	opaq.		84	karaden.	opaq.	
85	kontv	opaq.		86	kral	opaq.	
87	kralkar.	opaq.		88	manolya	opaq.	
89	mavikar.	opaq.		90	mehtap	opaq.	
91	meltem	opaq.		92	mesaj	opaq.	
93	mpl	opaq.		94	mtv	transp.	
95	mzefirst	opaq.		96	nova	opaq.	
97	ntv	opaq.		98	n. one	transp.	
99	ocko	opaq.		100	odeon	opaq.	
101	olay	opaq.		102	ordu	opaq.	
103	orf1	transp.		104	ozlem	opaq.	
105	powert.	opaq.		106	quizcall	opaq.	
107	rtl	transp.		108	rumeli	opaq.	

Table A.1 – Continued

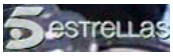
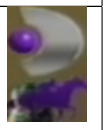
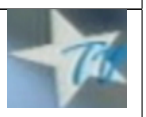
#	Name	Type	Logo	#	Name	Type	Logo
109	samany.	opaq.		110	ses	opaq.	
111	shopping	opaq.		112	show	opaq.	
113	showroom	opaq.		114	skytrav.	opaq.	
115	skyturk	opaq.		116	smart	opaq.	
117	star	opaq.		118	starcan.	opaq.	
119	stop	opaq.		120	su	opaq.	
121	susport	opaq.		122	t5estr.	opaq.	
123	t5sport	opaq.		124	ta3	opaq.	
125	tatlises	opaq.		126	tay	opaq.	
127	tca	opaq.		128	teknolo.	opaq.	
129	tempo	opaq.		130	tgrtHab.	opaq.	
131	top	opaq.		132	trt1	opaq.	
133	trt1can.	opaq.		134	trt2	opaq.	
135	trt3	opaq.		136	trtint	opaq.	
137	trtturk	opaq.		138	turkce	opaq.	
139	turkmen.	opaq.		140	turkshop	opaq.	
141	turkshow	opaq.		142	tv2	transp.	
143	tv58	opaq.		144	tv8	opaq.	

Table A.1 – Continued

#	Name	Type	Logo	#	Name	Type	Logo
145	tvnet	opaq.		146	tw1	opaq.	
147	uktvhis.	transp.		148	ulusal	opaq.	
149	ur	transp.		150	viva	opaq.	
151	wfashion	opaq.		152	yaban	opaq.	
153	yenicag	opaq.		154	yildiz	opaq.	
155	yleteema	transp.		156	yol	opaq.	
157	yumurcak	opaq.		158	ztv	opaq.	

REFERENCES

1. Albiol, A., M. J. C. Fulla, A. Albiol, and L. Torres, "Detection of TV Commercials", Proc. IEEE ICASSP, Montreal, Canada, May 2004.

2. Yan, W.Q., J. Wang, and M.S. Kankanhalli, "Automatic video logo detection and removal", Multimedia Systems 10(5): pp. 379-391, 2005.

3. Meisinger, K., T. Troeger, M. Zeller, and A. Kaup, "Automatic TV Logo Removal Using Statistical Based Logo Detection and Frequency Selective Inpainting", Proc. European Signal Processing Conference, September 2005.

4. Santos, A.R. and H.Y. Kim, "Real-Time Opaque and Semi-Transparent TV Logos Detection", Proc. 5th Int. Information and Telecommunication Technologies Symposium (I2TS), Cuiaba, 2006.

5. Cózar, J.R., N. Guil, J.M. Gonzlez-Linares, and E.L. Zapata, "Video Cataloging Based On Robust Logotype Detection", IEEE International Conference on Image Processing (ICIP), 2006.

6. Ekin, A. and R. Braspenning, "Spatial detection of tv channel logos as outliers from the content", Proceedings of SPIE – Volume 6077 Visual Communications and Image Processing, 2006.

7. Duffner, S. and C. Garca, "A neural scheme for robust detection of transparent logos in TV programs", Lecture Notes in Computer Science – II vol. 4132, pp. 14–23, Springer, Berlin 2006.

8. Wang, J., L. Duan, Z. Li, J. Liu, H. Lu, and J. Jin, "A Robust Method for TV Logo Tracking in Video Streams", icme, pp.1041–1044, 2006 IEEE International Conference on Multimedia and Expo, 2006.

9. Wang, J., Q. Liu, L. Duan, H. Lu and C. Xu, "Automatic TV Logo Detection, Tracking and Removal in Broadcast Video", MMM 2007, LNCS 4352, Part II, pp.63–72, 2007.

10. Günsel, B., A. Ferman, and A.M. Tekalp, "Temporal video segmentation using unsupervised clustering and semantic object tracking", Journal of Electronic Imaging, Volume 7, pp.592–604, 1998.

11. Millerson, G., The technique of television production, 12th Ed., NewYork, March 1990.

12. Canny, J., "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, pp. 679–698, 1986.

13. "Principal Components Analysis on Wikipedia", http://en.wikipedia.org/wiki/Principal_components_analysis, 2008.

14. Pearson, K., "On Lines and Planes of Closest Fit to Systems of Points in Space", Philosophical Magazine 2 (6): pp. 559-572, 1901.

15. Shlens, J., "A Tutorial on Principal Component Analysis", http://www.snl.salk.edu/~shlens/pub/notes/pca.pdf, 2005.

16. Smith, L. I., "A Tutorial on Principal Component Analysis", http://kybele.psych.cornell.edu/~edelman/Psych-465-Spring-2003/PCA-tutorial.pdf, 2002.

17. Lee, D.D. and Seung, H., "Learning the parts of objects by non-negative matrix factorization", Nature, 401, pp. 788–791, 1999.

18. Lee, D.D. and Seung, H., "Algorithms for Non-negative Matrix Factorization", Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference: pp. 556-562, MIT Press, 2001.

19. Okun, O. and Priisalu H., "Nonnegative matrix factorization for pattern recognition", Proceedings of the 5th IASTED International Conference on Visualization, Imaging and Image Processing, pp. 546-551, Benidorm, Spain, 7-9 September 2005.

20. "The FastICA Package for Matlab", http://www.cis.hut.fi/projects/ica/fastica.

21. Bartlett, M. S., Lades, H. M., and Sejnowski, T. J., "Independent component representations for face recognition," in Proc. SPIE Symp. Electon. Imaging: Science Technology - Human Vision and Electronic Imaging III, vol. 3299, pp. 528-539., San Jose, CA, 1998.

22. Bartlett,M. S., Movellan, J. R., and Sejnowski, T. J., "Face Recognition by Independent Component Analysis", IEEE Transaction on Neural Networks, Vol 13, pp. 1450-1464, 2002.

23. Ekenel, H.K. and Sankur, B., "Feature selection in the independent component subspace for face recognition", Pattern Recognition Letters(25), No. 12, pp. 1377-1388, September 2004.

24. Hyvärinen, A. and Oja E., "Independent Component Analysis. A Tutorial", http://www.cis.hut.fi/projects/ica, 1999.

25. Hyvärinen, A. and Oja, E., "Independent component analysis: Algorithms and applications", Neural Networks 13 (45), 411-430, 2000.

26. "Independent Component Analysis on Wikipedia", http://en.wikipedia.org/wiki/Independent_component_analysis, 2008.

27. Khayam, S. A., "The Discrete Cosine Transform (DCT): Theory and Application", seminar 1 – ECE 802-602: Information Theory & Coding, March 2003.

28. "Discrete Cosine Transform on Wikipedia", http://en.wikipedia.org/wiki/Discrete_cosine_transform, 2008.

29. Burges, C. J. C., “A Tutorial On Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery 2, 121–167, 1998.

30. Cortes, C. and Vapnik, V., “Support-Vector Networks”, Machine Learning, 20, http://www.springerlink.com/content/k238jx04hm87j80g, 1995.

31. Boser, B. E., Guyon, I. M., and Vapnik.,V. N. “A training algorithm for optimal margin classifiers”, In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, ACM Press, 1992.

32. “Support Vector Machine on Wikipedia”, http://en.wikipedia.org/wiki/Support_vector_machine, 2008.

33. Hsu, C. W. and Lin, C. J., “A Comparison of methods for multiclass support vector machines”, IEEE Trans. on Neural Networks, 2002.

34. Ding, C. H. Q. and Dubchak, I., “Multi-class protein fold recognition using support vector machines and neural networks”, Bioinformatics 17, pp 349–358, 2001.

35. Chang, C. C. and Lin, C. J., “LIBSVM: a library for support vector machines”, http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

36. Thangali, A., “A Directshow based AVI read interface for Matlab”, http://cs-people.bu.edu/tvashwin/dx_avi/index.html.