LOGO RECOGNITION IN VIDEOS
AN AUTOMATED BRAND ANALYSIS SYSTEM

by

Murat DURUŞ

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

August 2008

LOGO RECOGNITION IN VIDEOS
AN AUTOMATED BRAND ANALYSIS SYSTEM

APPROVED BY

Prof. Dr. AYTÜL ERÇİL (Thesis Supervisor)	....
Assist. Prof. Dr. GÖZDE ÜNAL	....
Assist. Prof. Dr. HAKAN ERDOĞAN	....
Assoc. Prof. Dr. UĞUR SEZERMAN	....
Assist. Prof. Dr. YÜCEL SAYGIN	....

DATE OF APPROVAL: ....

to all Electrical and Electronics Engineers

&

who interested in Computer Vision & Pattern Recognition

Acknowledgments

I would like to thank Prof. Dr. Aytül Erçil for her kindness, understanding, and support throughout my academic life, as well as the individual who introduced me to computer vision and pattern recognition. I owe many thanks to her for providing me with new perspectives in this field.

I also owe much appreciation to faculty members Asst. Prof. Dr. Müjdat Çetin and Asst. Prof Dr. Gözde Ünal for their endless help and support during my academic life. I am also grateful to Asst. Prof Dr. Hakan Erdoğan, Asst. Prof. Dr. Yücel Saygın and Assoc. Prof. Dr. Uğur Sezerman for their participation in my thesis committee.

I have been supported by Sabancı University and the Technological and Research Council of Turkey (TÜBİTAK) during my M.Sc. research; I owe a debt of gratitude to these foundations for their valuable support. I would like to thank to the founders of this wonderful university, especially the late Sakip Sabanci for providing all of us, faculty, staff, and students, with such a comfortable and creative atmosphere .

I wish to express my gratitude to the members of VPA Laboratories and graduate students in Sabanci University. I also owe many thanks to Cenker Öden for helping me to tackle some problems throughout the thesis. I have learnt a lot from our discussions.

Lastly, I would like to express my deepest gratitude to my family. Their constant encouragement, unfailing support and boundless love have always been the true sources of strength and inspiration throughout my life. Thank you with all my heart.

Special thanks go to Zuhal Temel for all the encouragement and support she has provided throughout the thesis.

August 2008 Murat DURUŞ

LOGO RECOGNITION IN VIDEOS
AN AUTOMATED BRAND ANALYSIS SYSTEM

Murat DURUŞ

Electronics Engineering, M.Sc. Thesis, 2008

Thesis Supervisor: Aytül Erçil

Keywords: invariant object recognition, logo recognition, shape based matching, scale invariant feature transform (SIFT)

Abstract

Every year companies spend a sizeable budget on marketing, a large portion of which is spent on advertisement of their product brands on TV broadcasts. These physical advertising artifacts are usually emblazoned with the companies' name, logo, and their trademark brand. Given these astronomical numbers, companies are extremely keen to verify that their brand has the level of visibility they expect for such expenditure. In other words advertisers, in particular, like to verify that their contracts with broadcasters are fulfilled as promised since the price of a commercial depends primarily on the popularity of the show it interrupts or sponsors. Such verifications are essential to major companies in order to justify advertising budgets and ensure their brands achieve the desired level of visibility. Currently, the verification of brand visibility occurs manually by human annotators who view a broadcast and annotate every appearance of a companies' trademark in the broadcast.

In this thesis a novel brand logo analysis system which uses shape-based matching and scale invariant feature transform (SIFT) based matching on graphics processing unit (GPU) is proposed developed and tested. The system is described for detection and retrieval of trademark logos appearing in commercial videos. A compact representation of trademark logos and video frame content based on global (shape-based) and local (scale invariant feature transform (SIFT)) feature points is proposed. These representations can be used to robustly detect, recognize, localize, and retrieve trademarks as they appear in a variety of different commercial video types. Classification of trademarks is performed by using shaped-based matching and matching a set of SIFT feature descriptors for each trademark instance against the set of SIFT features detected in each frame of the video.

Our system can automatically recognize the logos in video frames in order to summarize the logo content of the broadcast with the detected size, position and score. The output of the system can be used to summarize or check the time and duration of commercial video blocks on broadcast or on a DVD. Experimental results are provided, along with an analysis of the processed frames. Results show that our proposed technique is efficient and effectively recognizes and classifies trademark logos.

VİDEO ÇERÇEVELERİNDE LOGO TANIMA
OTOMATİK MARKA ANALİZİ SİSTEMİ

Murat DURUŞ

Elektronik Mühendisliği, Yüksek Lisans Tezi, 2008

Tez Danışmanı: Aytül Erçil

Anahtar Kelimeler: bağımsız obje tanıma, logo tanıma, şekil tabanlı eşleştirme, bağımsız öznitelik dönüşümü (SIFT)

Özet

Her yıl şirketler pazarlamaya büyük miktarda bütçe harcarlar; ki bunun geniş bir bölümü ürünlerinin televizyonlardaki reklamlarına harcanmaktadır. Bu fiziksel reklam yapıları genellikle şirketin ismi, logosu ve ticari markalarıyla donatılır. Verilen astronomik rakamlar düşünüldüğünde şirketler markalarının bu masraf için bekledikleri seviyede görünürlüğe sahip olduğunu doğrulamaya oldukça yatkındırlar. Diğer yönden reklamın fiyatı öncelikle yayına girdiği ya da sponsoru olduğu programın popülaritesine bağlı olduğu için reklam veren firmalar özellikle reklamcılarla kontratlarının söz verildiği gibi yerine getirilmesini sağlamayı isterler. Böyle doğrula malar büyük şirketlerin reklam bütçelerini kesinleştirmeleri ve markalarının istenilen seviyede görünürlüğe ulaştığından emin olmaları için gereklidir. Günümüzde markanın görünürlüğünün doğrulanması , reklamı izleyen ve reklamda şirketin ticari markasının her görüntüsünü not eden yorumcu kişiler tarafından elle oluşturulmaktadır.

Bu tezde şekile dayalı eşleştirme ve ölçekten bağımsız öznitelik dönüşümünün (SIFT) bir grafik işlemci ünitesinde (GPU) uygulanması yöntemi kullanılarak yeni bir logo tanıma sistemi önerilmiş, geliştirilmiş ve test edilmiştir. Bu sistem reklam videolarında görünen ticari marka logolarının belirlenmesi ve bu logolara erişilmesi için tanımlanmıstır. Ticari marka logolarının global (şekil tabanlı) ve yerel (SIFT)) zellik noktalarına dayalı olarak video çerçevelerinde içeriğinin tanınması sistemi sunulmaktadır. Bu gösterimler, ticari markalar farklı çeşitlerde reklam video şekillerinde göründüklerinden dolayı onları gürbüz bir şekilde saptamak, tanımak, sınırlamak ve onlara erişmek için kullanılabilir. Ticari marka logolarının sınıflandırılması şekil tabanlı eşleştirmelerle ve videonun her bir çerçevesi için belirlenen SIFT öznitelik noktalarına karşı her ticari marka örneği için bulunnan SIFT öznitelik noktalarının eşleştirilmeleriyle oluşturulur.

Bu çalışma sonucunda yayının logo içeriğini istenilen boyut, konum ve başarı oranıyla özetlemek için video çerçevelerindeki logoları otomatik olarak tanıyan bir sistem tasarlanmıstır. Sistemin çıktısı yayındaki veya DVD ye kaydedilmiş reklam video bloklarının süresini, zamanını ve içeriğini kontrol etmek ya da özetlemek için kullanilabilir. Deneylere dayanan sonuçlar işlenmiş çerçevelerin analiziyle tedarik edilmiştir. Sonuçlar göstermektedir ki sunduğumuz teknik etkilidir ve ticari logoları etkin bir şekilde tanımakta ve sınıflandırmaktadır.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents a novel automated trademark logo recognition system on commercial videos, based on global and local features of the trademark logos. The proposed method is invariant to occlusion, scale, clutter, affinity, translation and rotation as well as to illumination. The system is designed to recognize the logos appearing in commercial video frames in order to summarize logo content of the commercials with the goal of detecting the content of the commercial videos.

1.1 Motivations

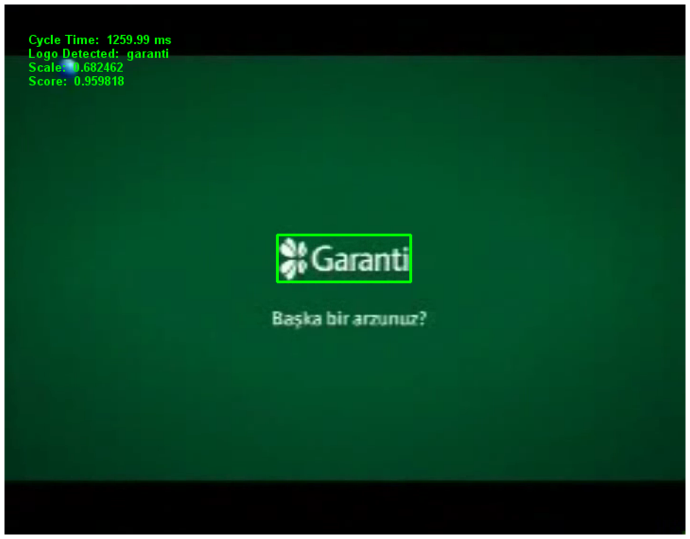
A large portion of the sizeable budget that international firms often spend annually is allocated on the advertisement of their product brands on TV broadcasts. These physical advertising artifacts are usually emblazoned with the companies' name, logo, and their trademark brand. Given these astronomical numbers, companies are extremely keen to verify that their brand has the level of visibility they expect for such expenditure. In other words advertisers, in particular, like to verify that their contracts with broadcasters are fulfilled as promised since the price of a commercial depends primarily on the popularity of the show it interrupts or sponsors. Such verifications are essential to major companies in order to justify advertising budgets and ensure their brands achieve the desired level of visibility. Currently, the verification of brand visibility occurs manually by human annotators who view a broadcast and annotate every appearance of a companies' trademark in the broadcast. [4].

In this thesis we describe a system for detection and retrieval of trademarks appearing in commercial videos. Here a compact representation of trademarks and video frame content based on global shape based object features and local scale invariant feature transform SIFT feature points is proposed. These representations can be used to robustly detect, localize, and recognize trademark logos as they appear in a variety of different commercial video types. Classification of trademarks is performed by two methods, to introduce;

1-) matching a set of shape feature descriptors for each trademark logo instance against the set of shape features detected in each frame of the video (Figure 1.1).

2-) matching a set of SIFT feature descriptors for each trademark instance against the set of SIFT features detected in each frame of the video (Figure 1.2).

Experimental results are provided, along with an analysis of the processed frames. Results show that our proposed technique is time efficient and effectively recognizes and classifies trademark logos.



```
Cycle Time:  1259.99 ms
Logo Detected:  garanti
Scale:  0.682462
Score:  0.959818
```

Figure 1.1: Logo Recognition using Shape Based Matching

1.2 Contributions

In this thesis, a novel brand analysis system for the recognition of trademark logos in TV commercials is proposed, developed and tested. The system is identified as a novel one due to the implementation of the shape-based method [5] and the SIFT algorithm [3] on a graphics processing unit (GPU)for trademark logo recognition. Both of the above methods have been simultaneously developed simultaneously and tested

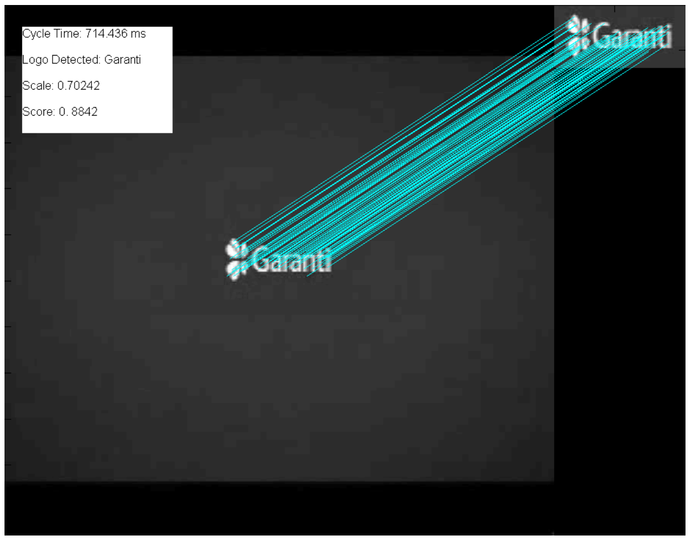


Figure 1.2: Logo Recognition using SIFT based Matching.

for a dataset of 50 different trademark logos. Results and benchmarking demonstrate that this system posseses developed has a large performance advantages, regarding recognition and timing results, over those of previous methods. Furthermore, the implementation of the SIFT algorithm on a GPU card speeds up the process while reserving the recognition performances. The results show that our system has a very good performance of recognition and time in comparison to previous methodologies.

1.3 Organization of the Thesis

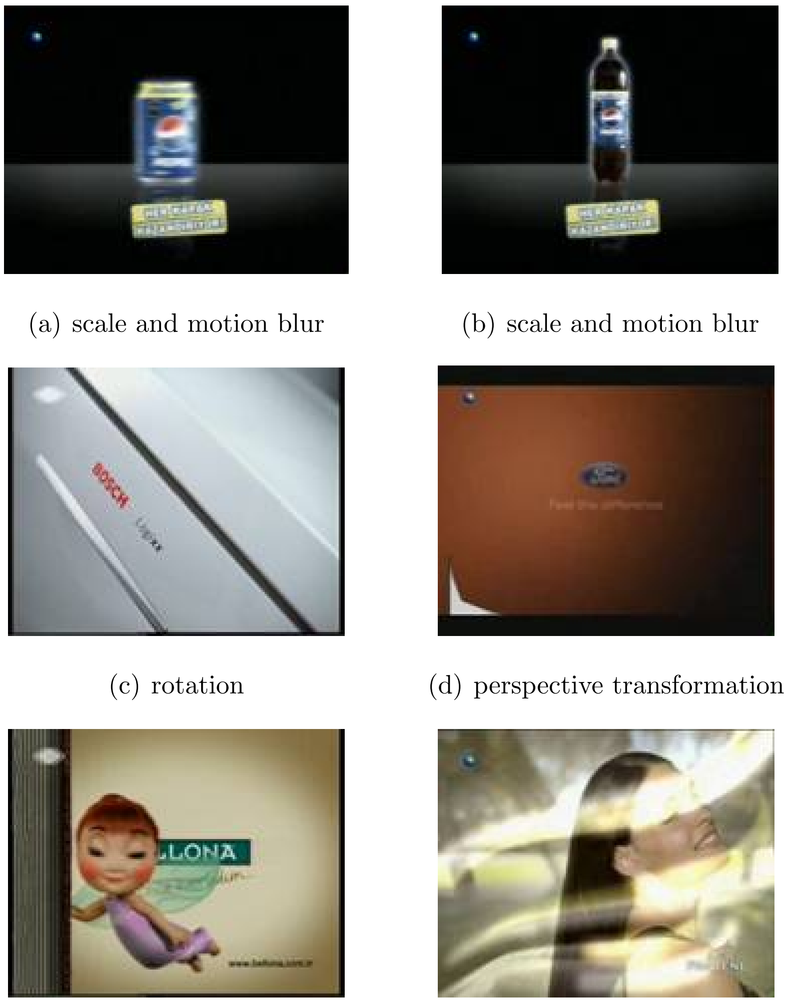
In chapter 2 related works regarding trademark logo recognition are presented. Chapter 3 provides the techniques for the detection of commercial blocks in TV broadcast and the particular methodologies involved. Chapter 4 and Chapter 5 explain the methods for trademark logo recognition, namely shape based matching and scale invariant feature transform (SIFT). Chapter 6 explains the experiments implemented with the results of the techniques employed. Chapter 7's conclusion suggests probable, future work on trademark logo recognition in the light of our research.

Chapter 2

Related Work

The problem of automatic trademark and logo detection and recognition has been a subproblem of object recognition, an issue that has been examined following many different approaches in recent decades. The two primary types of features which have been used are geometric and photometric object features: the former rely on properties of objects such as lines, vertices, curves and shapes ([6],[7]), while the latter are computed from pixel values (luminance or color) of the imaged object ([8],[9],[10]). Object detection and recognition using photometric features has been the subject of much recent research due to the fact that if these features are computed locally, they can cope with the problem of occlusion and are able to more accurately distinguish similar objects [8]. However, they are generally not robust to illumination as well as the motion blur in video frames. Some of the widely seen logo examples are seen in Figure 2.1.

Most of the work related to trademark logo recognition deals with the problem of content-based indexing and retrieval in logo databases, with the goal of assisting in the detection of trademark infringement by comparing a newly designed trademark with archives of already registered logos ([11],[12],[13],[14]). In this scenario, it is generally assumed that the image acquisition and processing chain is controlled so that the images are of acceptable quality and are not distorted. The problem of trademark recognition in videos is inherently more difficult, since the entire process is not controlled and several limitations of the imaging equipment introduce considerable distortion and loss of quality of the original logos, namely;



(a) scale and motion blur (b) scale and motion blur

(c) rotation (d) perspective transformation

(e) occlusion (f) illumination

Figure 2.1: Various logo examples.

1-) video interlacing,

2-) color sub-sampling

3-) motion blur, etc.

In [15] the problem of detecting and tracking billboards in soccer videos was studied, with the goal of superimposing different advertisements according to different audiences. Billboards are detected using colour histogram back projection and represented using a PD in an invariant color space estimated from manually annotated video frames. The focus of this work is on detection and tracking rather than recognition. In [16] logo appearance is detected by analyzing sets of significant edges and applying heuristic techniques to discard small or sparsely populated edge regions of the image. The logo recognition method proposed in [17] extends the work presented in [18] and deals with logos appearing on rigid planar surfaces that have a homogeneously colored background; the video frame is binarized and logo regions are combined using heuristics. The Hough transform space of the segmented logo is then searched for large values to find the image intensity profiles along lines. Logo recognition is performed by matching these lines with the line profiles of the models. In [19] candidate logo regions are detected using color histogram back-projection and then are tracked. Multidimensional receptive field histograms are then used to perform logo recognition. For every candidate region the most likely logo is computed, and thus if a region does not contain a logo the precision of identification is reduced. In [20] the architecture for a system for media monitoring is presented. The system provides logo detection and recognition functionalities, and the authors briefly discuss a variation of the SIFT algorithm to select and track keypoints in videos. The points are used for trademark recognition, but the logo matching algorithm is not described, and very few results of the proposed variation are provided.

In this thesis we propose a system for automatically detecting and retrieving trademark logo appearances in commercial videos. Figure 2.2 presents an overview of the entire system. In brief, a TV broadcast video is recorded directly to DVD or it is directly taken from the broadcast. A collection of static commercial video frames in this video are then processed to extract a compact representation or the summarization of the logo content of broadcast. The trademark logo images are firstly trained by using the logo images gathered from the Google image search. The results of this processing may be stored in a file for later retrieval. In other words, all of the trademarks are then matched against the content extracted from every frame of the video to compute a "match score" indicating the likelihood that the trademark logo occurs at any given point in the video frame. The black frames are used to detect the intervals of the video likely to contain the commercial blocks, i.e. trademark logo image. Retrieved segments are used to drive a user interface used by a human annotator who can then validate this automatic annotation.

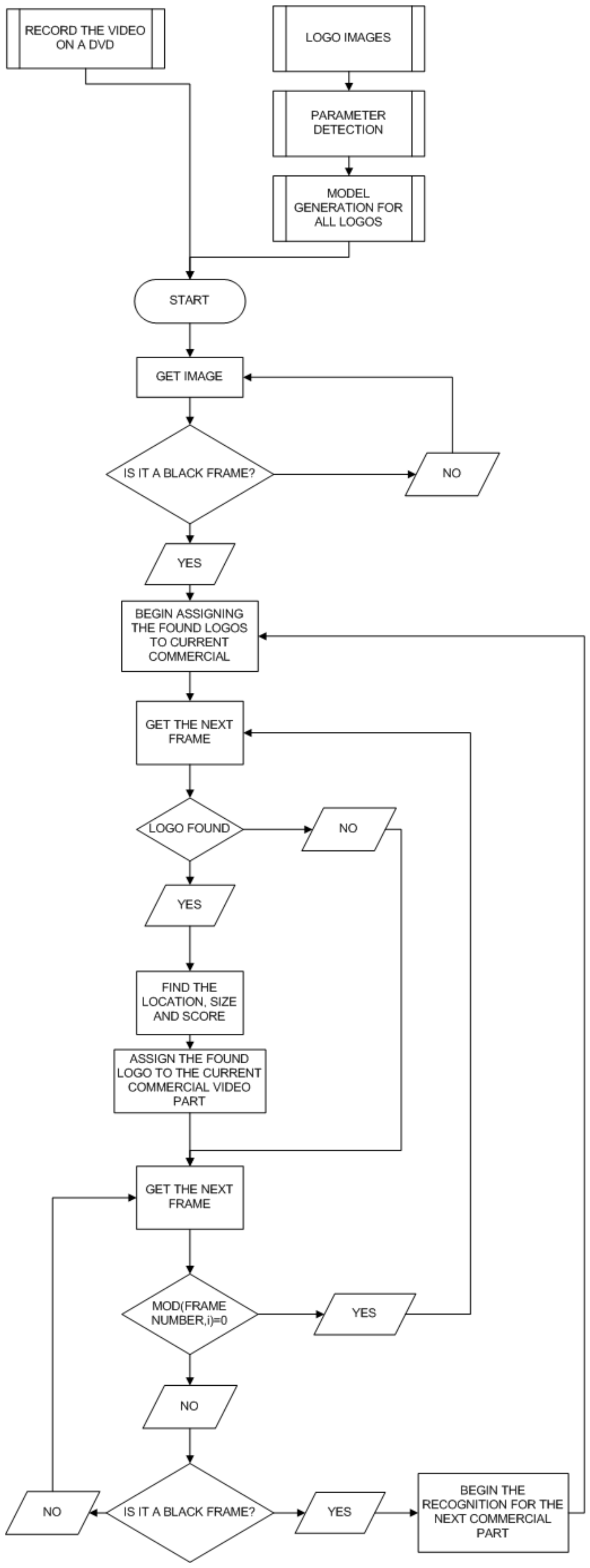

RECORD THE VIDEO ON A DVD
LOGO IMAGES
PARAMETER DETECTION
MODEL GENERATION FOR ALL LOGOS
START
GET IMAGE
IS IT A BLACK FRAME?
NO
YES
BEGIN ASSIGNING THE FOUND LOGOS TO CURRENT COMMERCIAL
GET THE NEXT FRAME
LOGO FOUND
NO
YES
FIND THE LOCATION, SIZE AND SCORE
ASSIGN THE FOUND LOGO TO THE CURRENT COMMERCIAL VIDEO PART
GET THE NEXT FRAME
MOD(FRAME NUMBER,i)=0
YES
NO
NO
IS IT A BLACK FRAME?
YES
BEGIN THE RECOGNITION FOR THE NEXT COMMERCIAL PART

Figure 2.2: Overview of the entire system.

Chapter 3

Commercial Detection

For many companies, TV commercials provide critical marketing tools. Their interspersion within regular broadcast television programming can be entertaining, informative, annoying, or a sales goldmine - depending on one's viewpoint. As a result, the detection or deletion of commercial segments within television broadcasts has long been a research focus. Interestingly enough, the goals of these two applications-at least indirectly-are at odds with each other [4].

One application seeks to identify and track when specific commercials are broadcast. Advertisers, in particular, like to verify that their contracts with broadcasters are fulfilled as promised, since the price of a commercial depends primarily on the popularity of the show it interrupts. The more individuals (the product's demographics) watching the program, the higher the cost, usually . Thus, all advertisers desire to ensure that a particular commercial runs during a specified program [4].

The other group, those who want to detect commercials for the purpose of elimination, from their recordings is composed of viewers who desire to watch their recorded television shows without the annoyance of commercials. Apart from individuals, video database maintainers also appreciate the ability to automatically edit out commercials in stored shows and thereby decrease storage requirements. Advertisers are naturally strongly opposed to such devices as they defeat the commercial's intended financial gains such commercials will provide. This section will discuss several algorithms that have been experimentally used to detect commercials, as well as devices that are currently available for this purpose [4].

3.1 Characteristics of Commercials

The problem of detecting commercials within television broadcasts is related to several - more general - problems in video processing. These issues include scene break detection, video segmentation, and video indexing and retrieval. However, commercial segments have certain characteristics that make their identification easier than of general video segments. These characteristics make it possible to use detection algorithms suitable for feature extraction from a general video database.

To start with, commercials are almost always grouped into blocks, typically consisting of four to 10 commercials each. As shown in Figure 3.1, at the beginning and end of each commercial block and between each commercial in the block, several frames of monochrome black are displayed. On many stations, the observation has been that the last two to three commercials of a block are commercials promoting upcoming shows. Also, some countries (e.g. Germany, Turkey) have laws requiring that every commercial block begin with a standard "commercial block introduction" sequence. In Turkey, it is a TV broadcast policy that at least 3 black frames have to be introduced before going into the commercial block.

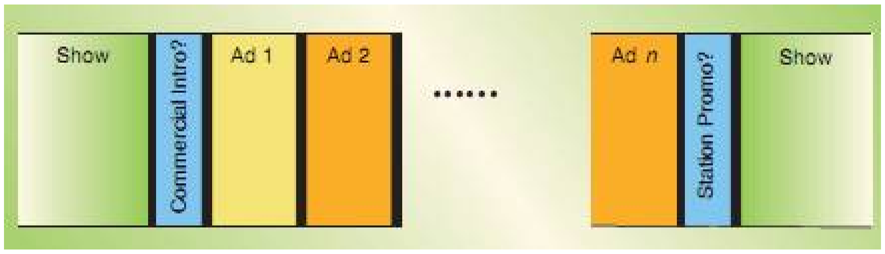


Figure 3.1: Structure of typical commercial block.

Many television stations also have a practice of displaying a network logo in the corner of the screen during regular programming and then removing this logo during commercial breaks. Within a given television series, all episodes generally have commercial breaks scheduled at approximately the same time in the episode. Also, many commercials are repeated on a frequent basis, particularly for a given station. Several other characteristics relate to the individual commercials. The duration of individual commercials is a minimum of 15 seconds and in order to capture viewers' attention in the small amount of time available to convey a message, commercials tend to be high in “action”, typified by a high number of cuts between frames among other factors. (In [1] it is noted that the average “hard” cut rate in a sample of 200 commercials from German television was 20.9 cuts per minute, while the rate in the accompanying movie clips was only 3.7 cuts per minute.) There are usually a large number of frames with text containing the product or company’s name. Also, to leave the product in the viewer’s mind, the last few seconds of many ads consist of “still” shots of the product or slogan. These still images generally give us the advantage to catch the current trademark logo and assign it to the current commercial block.

Other characteristics beyond the visual information are often present as well. The most noticeable characteristic, and the one most irritating to viewers, is the tendency of broadcasters to increase the volume level of the audio track during commercials. Another audio clue to the presence of commercials is that the delimiting black frames at the beginning and end of commercials are accompanied by silence in the audio track. Also, the dialogue on the audio track generally contains the product or company’s name. Finally, when closed captioning is available for a television show, it is generally discontinued during commercial breaks. No currently proposed detection algorithm utilizes all of these clues. Most algorithms, however, do detect various combinations of them in order to improve detection rates.

In our work we used the arithmetic mean of gray values of the current frame in order to detect whether it is a black frame. We also check the variance of the gray values of the image. For our application, only the mean of the gray values was sufficient to detect the black frames; however, we also employed the variance of the gray values in order to re-check the detection.

$$\mu_X = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{3.1}$$

$$\sigma_X^2 = \sum_{i=1}^{n}(X_i - \mu_X)^2 \tag{3.2}$$

where n is the number of pixels, X_i is the gray value of the pixel, μ_X and σ_X^2 mean and variance of the image gray values, respectively. By using these features and implementing the algorithm given by Figure 3.1 above, we were able to detect the black frames with 100% accuracy.

3.2 Detection Schemes

There are two main categories of methods to detect commercials. Feature-based detection relies on general characteristics of commercials to detect their presence. Any of the commercial characteristics mentioned earlier could be used to indicate the (possible) presence of a commercial. Recognition-based detection attempts to identify individual commercials in the broadcast as matching commercials it has already learned.

3.2.1 Black Frames and Silences

The most common characteristics used in commercial detection are the delimiting black frames and silences. In locating black frames, the simplest method is to look at the average intensity value of the pixels in the image. The average intensity is determined easily in the analog domain [21] and is the basis for most current commercial applications. The determination that a given frame is "black" is based on the average being below a pre-determined threshold value. Improved black frame detection can be accomplished by requiring that the standard deviation of the intensity values also be below a threshold according to [1]. Some work has also been done by [22] regarding a method to detect black frames in an MPEG-encoded bit stream, without the computational cost of decoding.

Silent audio frames may be similarly detected by examining the average volume level on the audio track. Most applications couple these two functions to decrease the likelihood of a false detection of a black frame or the detection of an irrelevant black frame within a program. With this rule, a black frame is only detected if accompanied by a silence. To further reduce the chance of random black frame detection, most algorithms require that a certain number of least black frames be detected together, usually three or five. Here we used the black frames that have at least two consecutive comings for entering to the recognition algorithm. Once black frames can be reliably detected, the timing aspects of the commercial breaks can be exploited. Two black frame series detections may indicate a commercial segment is between them. Most algorithms establish a maximum time between black frame sequences for a segment to be considered as a possible commercial. If the time between black frame sequences is greater than this, the segment is considered to be part of the program. The algorithm proposed by [22] sets this maximum commercial length at ninety seconds (2250 frames at 25 frames per second). This algorithm also looks at how many consecutive commercial segments occur to determine if the candidate commercial is in fact part of a commercial block. The video block detection algorithm requires that if a potential block does not contain at least three individual commercials, then it must be part of the program. (That is, if at least four black frame sequences occur with a maximum separation between each of ninety seconds, it is classified as a commercial break.)

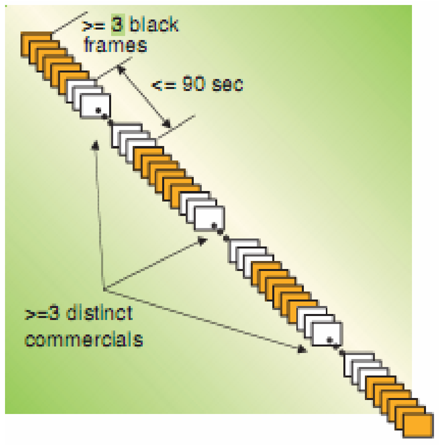


Figure 3.2: An example commercial block.

In this work we trigger the system by the detection of the first black frame while also checking the next frames for approval. After this triggering the system enter into the recognition mode as indicated in the block diagram of the system in Figure 2.2. Similar to our approach, using 10 broadcast clips, [22] evaluated the algorithm 3.2. The total amount of time was 315 minutes and included various genres such as sports, news and talk shows. The 10 broadcast clips contained a total of 11 commercial breaks as determined by human inspection. The algorithm detected all 11 commercial breaks, and none of the programming content was missed. However, the algorithm did fail to detect parts of the last commercial in some of the blocks, incorrectly including them with the programming instead. Still, the algorithm is said to perform reasonably well. Calculation of a performance measure called "recall" which is the percentage of commercial time correctly identified as such, showed that only one of the 11 clips had a recall rate below 85. Eight of the clips had a recall rate greater than 98%.

Our application tested the black frame detection algorithm on a 7 different videos lasting 36 minutes and 24 seconds. We have tested the algorithm on our dataset and achieved a recognition rate of 100%.

3.2.2 High Cut Rate and Action

Another characteristic used in feature-based detection is the high cut rate typically observed in commercials. The problem of determining the cut rate of a video segment is basically the same as the problem of determining shot changes (where the video switches from one shot to another). Once shot changes have been located, determining the cut rate is merely a matter of counting.

A number of methods have been proposed to locate shot changes; most use statistics on differences in the color histogram from one frame to the next. Another method proposed by [23] uses a wavelet-based distance metric to quantify the difference between two frames and identify cuts. One algorithm for using the cut rate in commercial detection, proposed by [1], had two basic rules:

1) a candidate sequence must have a cut rate above five cuts per minute for its entirety,

2) the cut rate must go above 30 cuts per minute at some point.

This algorithm had a recall rate of 93.43% and a false detection rate of 0.09%, confirming the suitability of using strong hard cuts as a pre-filter for commercial blocks. Some algorithms incorporate other editing techniques used frequently in commercials, such as fades and dissolves, to indicate the possible presence of commercials. Lienhart's group uses two additional metrics related to the high level of action in commercials [1]. First, the "edge change ratio" describes the number of edge pixels (as found by an edge detection algorithm) entering and leaving a frame. The second metric, called motion vector length, describes the motion of objects in the image. It is similar to the motion vectors calculated in MPEG encoding. Detection methods based on these two metrics both had recall rates around 96% when used on their test database.

Naturally, feature-based detection is most effective when multiple characteristics are considered together. In [1] a combined system that has two steps is created. First, the black frame sequence detector and the cut-rate detector are used to find candidate commercial segments. Then those candidate segments are passed to the action detectors (edge change ratio and motion vector length) to find the exact commercial block limits. The advantage of this two-step system is that the more computationally expensive operations can be reserved for the second step.

3.2.3 Recognition-Based Methods

Recognition-based detection methods are specialized video database systems that maintain a database of known commercials. To determine if the current segment of a television broadcast is a commercial, the segment is compared to known commercials using a query-by- example type operation. If a match is found, then the segment is almost certainly a commercial (depending on the precision of the matching algorithm). This process is somewhat similar to the trademark logo recognition but this precision is much less that the trademark matching we have performed in our analysis since the goal is only to detect commercial blocks. Also, apart from the recognition of logos, another indicator, such as the changed channel logo, may be used here in order to classify commercial parts of a TV broadcast.

Because of the computational expense involved in searching through a video database, most recognition-based algorithms use at least a simple feature-based detector to determine candidate video segments, i.e. a shot segmentation algorithm or a black frame sequence detector. Their purpose is to determine the start point for the video segment to be sent to the database. Since the black frames or cuts are already being located for that purpose, it is convenient to look at their timing to perform a feature-based pre-selection.

Recognition-based systems are susceptible to problems in matching a segment from a broadcast to the same one in the database because of the variations caused by irregularities in the broadcast. Color levels of the same commercial, for example, can vary from station to station. Also, commercials are sometimes edited to shorten their length, which make them somewhat more difficult to match. Thus, any recognition-based system must be flexible enough in its search algorithm to allow for such variations. There is some evidence that, because of broadcasting variations, the color histogram techniques that are prevalent in video database indexing may not be ideally suited for recognizing commercials. The wavelet-based approach of [23] and the gradient method of [24] are examples of algorithms that use non-color based indices to overcome this problem.

The recognition-based algorithm proposed by [1] uses a database-matching scheme that can match subsequences within video segments. This ability makes it possible to recognize edited commercials. This algorithm searches the database in two steps. The algorithm uses an index of color coherence vectors (CCV). These vectors are similar to color histograms but offer give some spatial information by indicating how many pixels are contained in “monochromatic” regions in the image.

As shown in Fig. 6, [1] used a sliding window to indicate the segment of the current broadcast to send to the database for a possible match. In the first step, a window of L seconds is compared to the first L+S seconds of the commercials in the database. If a potential match is found in the database, the comparison window is expanded to the full length of the candidate commercial. Because of the lower number of frames compared, the first step in this algorithm is markedly shorter than a search using the entire commercial. This first step weeds out enough non-matches to provide a net decrease in computation time, even though two searches are required to detect a single commercial. Experimentally, this algorithm is said to correctly identify all 125 commercials in three hours of video when given a 200 commercial database in which to search. On the average, the beginning and end frames of the commercials were detected to within 5 frames of the actual.

Recognition-based systems face the drawback that commercials must be known (and indexed in the database) before they can be recognized in a broadcast. There are three possible modes of operation that have been proposed to accomplish this necessary commercial “learning”. First, the user could indicate to the algorithm when a new commercial is encountered. The system would then store that new commercial in the database. Second, companies could compile databases of the most frequently

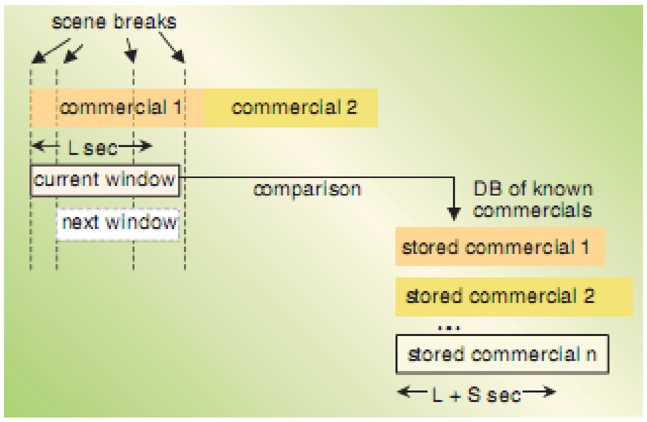


Figure 3.3: First step of recognition-based algorithm proposed by [1].

aired commercials and sell such databases to users. The third, most useful, option is for the system to automatically learn new commercials as it encounters them. In [1] a system for such automated commercial learning has been proposed. It assumes that most commercials are already known. New commercials are entered into the database if they are surrounded by two previously known commercials (and are less than ninety seconds). Of course, this method will tend to miss ads such as station promos that generally appear at the end of commercial blocks.

3.3 Applications

As noted above, there are two major areas of application for commercial detection algorithms: “commercial trackers” and “commercial killers”. Commercial trackers are designed to automatically audit the broadcast of commercials so advertisers can verify fulfillment of their “air play” contracts. Clearly, this application must use recognition-based methods because specific commercials are being sought out. If feature-based indicators are used within such recognition-based devices, it is desirable to adjust any threshold values to minimize false negatives. This way the chance of missing commercials will be minimal [4].

Commercial killers try to remove commercials from the recordings so that viewers do not have to watch them on playback. Devices for this purpose started showing up in the mid-90s. Today, some of the most major VCR brands offer an option, generally called “Commercial Advance”, to accomplish this. All major brands rely on the same technology developed by [25]. The algorithm is a simple one based on detecting black frame sequences and analyzing the timing between them. As a broadcast is recorded on the VCR, the algorithm keeps track of when the black frames occur. When the recording stops, it performs the necessary computations to determine the location of commercial blocks. This information is then encoded on the videotape. When the tape is subsequently viewed, the VCR automatically fast-forwards past commercial blocks.

Chapter 4

Shape Based Object Recognition

Object recognition is a broad area of computer vision and image processing which is useful for matching an image with a model of the object. Generally the model of the image is generated from an image of a template object.

4.1 Theoretical Background on Object Recognition using Shape Based Matching

For our application the model of the object is generated from logos which are searched over the image search facility of *Google*®. Here the selection of the logo is determined regarding some features such as;

- the size
- the shape characteristics
- the clutter on the logo model image.
- the noise in the model image etc.

The details of these requirements are explained in the following sections. Several methods have been proposed to recognize objects in images by 2D matching of the models and objects. From now on matching means 2D image matching throughout the thesis. A survey of matching approaches is given in [26]. In most matching approaches the model image is systematically compared to the test image using all degrees of freedom of the chosen class of transformations. The comparison in these approaches is based on a suitable similarity measure. To speed up the recognition process, the search is done in a coarse-to-fine manner using image pyramids [27].

The simplest class of object recognition methods is based on the gray level values of the model image and test image itself and uses normalized cross correlation or the sum of squared or absolute differences as a similarity measure [26]. Normalized cross correlation is invariant to linear brightness changes but is very sensitive to clutter and occlusion as well as nonlinear contrast changes. The sum of gray value differences is not robust to any of these changes, but can be made robust to linear brightness changes by explicitly incorporating them into the similarity measure, and to a moderate amount of occlusion and clutter by computing the similarity measure in a statistically robust manner [28].

A more complex class of object recognition methods do not use the gray values of the model or object itself, but use the object's edges for matching. Two example representatives of this class are [29] and [30]. In most of the existing approaches, the edges are segmented, i.e., a binary image is computed for both the model and the search image. Usually, the edge pixels are defined as the pixels in the image where the magnitude of the gradient is maximum in the direction of the gradient. Various similarity measures can then be used to compare the model to the image. The similarity measure in [29] computes the average distance of the model edges and the image edges. The disadvantage of this similarity measure is that it is not robust to occlusions because the distance to the nearest edge increases significantly if some of the edges of the model are missing in the image.

The Hausdorff distance similarity measure used in [30] tries to remedy this shortcoming by calculating the maximum of the $k-th$ largest distance of the model edges to the image edges and the $l-th$ largest distance of the image edges to the model edges. If the model contains n points and the image contains m edge points, the similarity measure is robust to $100k/n\%$ occlusion and $100l/m\%$ clutter. Unfortunately, an estimate for m is needed to determine l, which is usually not available [5].

All of these similarity measures have the disadvantage that they do not take into account the direction of the edges. In [31] it is shown that disregarding the edge direction information leads to false positive instances of the model in the image. The similarity measure proposed in [31] tries to improve this by modifying the Hausdorff distance to also measure the angle difference between the model and image edges. Unfortunately, the implementation is based on multiple distance transformations, which makes the algorithm too computationally expensive for faster inspection.

In all of the above approaches, the edge image is binarized. This makes the object recognition algorithm invariant only against a narrow range of illumination changes. If the image contrast is lowered, progressively fewer edge points will be segmented, which has the same effects as progressively larger occlusion.

In [5] an object recognition system which works close to real time and uses novel similarity measures that is inherently robust against occlusion, clutter, and nonlinear illumination change is described. In this work the matching is performed based on the maxima of the similarity measure in the transformation space. Here there are also options where subpixel-accurate poses are obtained by extrapolating the maxima of the similarity measure from discrete samples in the transformation space. In addition, for very high accuracy requirements, least-squares adjustment is used to further refine the extracted pose.

In addition, another class of edge based object recognition algorithms is based on the generalized Hough transform (GHT) [32]. Approaches of this kind have the advantage that they are robust to occlusion as well as clutter. Unfortunately, the GHT requires extremely accurate estimates for the edge directions or a complex and expensive processing scheme, e.g., smoothing the accumulator space, to determine whether an object is present and to determine its pose. This problem is especially grave for large models. The required accuracy is usually not obtainable, even in low noise images, because the discretization of the image leads to edge direction errors that already are too large for the GHT.

In all approaches above except [5], the edge image is binarized. This makes the object recognition algorithm invariant only against a narrow range of illumination changes. If the image contrast is lowered, progressively fewer edge points will be segmented, which has the same effects as progressively larger occlusion. The similarity measures proposed in [5] overcome all of the above problems and result in an object recognition strategy robust against occlusion, clutter, and nonlinear illumination changes. In the shape-base implementation, we used the similarity features defined in [5] in order to match the trademark logo images with the selected video frames. The details of this implementation are explained in detail in the following sections.

4.2 The Similarity Measures Used

In this approach the model of an object consists of a set of edge points $p_i = (x_i, y_i)^T$ and associated direction vectors $d_i = (t_i, u_i)^T, i = 1, ...n$ [5]. The direction vectors can be generated by a number of different image processing operations, e.g., edge, line, or corner extraction, as discussed in the previous section. Typically, the model is generated from a training image of the object that provides the required shape features of the model. Here it is better to define the model from the template of the object with low noise and scale, size, illumination differences. It is also advantageous to specify the coordinates p relative to the center of gravity i of the ROI of the model or to the center of gravity of the points of the model.

The test image in which a given object to be found can be transformed into a representation in which a direction vector $e_{x,y} = (v_{x,y}, w_{x,y})^T$ is obtained for each image point. In the matching process, a transformed model must be compared to the image at a particular location. In the most general case considered here, the transformation is an arbitrary affine transformation. It is useful to separate the translation part of the affine transformation from the linear part. Therefore, a linearly transformed model is given by the points $p_i^{'} = Ap_i$ and the accordingly transformed direction vectors $d_i^{'} = Ad_i$ where;

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

As discussed above, the similarity measure by which the transformed model is compared to the image must be robust to occlusions, clutter, and illumination changes. One such measure is to sum the (unnormalized) dot product of the direction vectors of the transformed model and the image over all points of the model to compute a matching score at a particular point $q = (x, y)^T$ the image, i.e., the similarity measure of the transformed model at the point q, which corresponds to the translation part of the affine transformation, is computed as follows:

$$
\begin{aligned}
s &= \frac{1}{n}\sum_{i=1}^{n}\langle d_i', e_{q+p'}\rangle \\
&= \frac{1}{n}\sum_{i=1}^{n} t_i' v_{x+x_i', y+y_i'} + u_i' w_{x+x_i', y+y_i'}
\end{aligned}
\tag{4.1}
$$

If the model is generated by edge or line filtering, and the image is preprocessed in the same manner, this similarity measure fulfills the requirements of robustness to occlusion and clutter. If parts of the object are missing in the image, there are no lines or edges at the corresponding positions of the model in the image, i.e., the direction vectors will have a small length and hence contribute little to the sum. Likewise, if there are clutter lines or edges in the image, there will either be no point in the model at the clutter position or it will have a small length, which means it will contribute little to the sum.

The similarity measure in equation 4.1 is not truly invariant against illumination changes, however, since usually the length of the direction vectors depends on the brightness of the image, e.g., if edge detection is used to extract the direction vectors. However, if a user specifies a threshold on the similarity measure to determine whether the model is present in the image, a similarity measure with a well defined range of values is desirable. The following similarity measure achieves this goal:

$$
\begin{aligned}
s &= \frac{1}{n}\sum_{i=1}^{n}\frac{\langle d_i', e_{q+p'}\rangle}{\parallel d_i' \parallel \cdot \parallel e_{q+p'} \parallel} \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{t_i' v_{x+x_i', y+y_i'} + u_i' w_{x+x_i', y+y_i}}{\sqrt{t_i'^2 + u_i'^2}\cdot\sqrt{v^2_{x+x_i', y+y_i'} + w^2_{x+x_i', y+y_i'}}}
\end{aligned}
\tag{4.2}
$$

Because of the normalization of the direction vectors, this similarity measure is additionally invariant to arbitrary illumination changes since all vectors are scaled to a length of 1. What makes this measure robust against occlusion and clutter is the fact that if a feature is missing, either in the model or in the image, noise will lead to random direction vectors, which, on average, will contribute nothing to the sum.

The similarity measure in equation 4.2 returns a high score if all the direction vectors of the model and the image align, i.e., point in the same direction. If edges are used to generate the model and image vectors, this means that the model and image must have the same contrast direction for each edge. Sometimes it is desirable to be able to detect the object even if its contrast is reversed. This is achieved by:

$$s = \left|\frac{1}{n}\sum_{i=1}^{n}\frac{\langle d_i', e_{q+p'}\rangle}{\| d_i' \| \cdot \| e_{q+p'} \|}\right| \tag{4.3}$$

In rare circumstances, it might be necessary to ignore even local contrast changes. In this case, the similarity measure can be modified as follows:

$$s = \frac{1}{n}\sum_{i=1}^{n}\frac{|\langle d_i', e_{q+p'}\rangle|}{\| d_i' \| \cdot \| e_{q+p'} \|} \tag{4.4}$$

The above three normalized similarity measures are robust to occlusion in the sense that the object will be found if it is occluded. As mentioned above, the results from the fact that the missing object points in the instance of the model in the image will on average contribute nothing to the sum. For any particular instance of the model in the image, this may not be true, e.g., because the noise in the image is not uncorrelated. This leads to the undesired fact that the instance of the model will be found in different poses in different images, even if the model does not move in the images, because in a particular image of the model the random direction vectors will contribute slightly different amounts to the sum, and hence the maximum of the similarity measure will change randomly. To make the localization of the model more precise, it is useful to set the contribution of direction vectors caused by noise in the image to zero. The easiest way to do this is to set all inverse lengths $\frac{1}{\|e_{q+p'}\|}$ of the direction vectors in the image to 0 if their length $\| e_{q+p'} \|$ is smaller than a threshold that depends on the noise level in the image and the preprocessing operation that is used to extract the direction vectors in the image. This threshold can be specified easily by the user. By this modification of the similarity measure, it can be ensured that an occluded instance of the model will always be found in the same pose if it does not move in the images.

The normalized similarity measures above (equations 4.2, 4.3, 4.4) have the property that they return a number smaller than 1 as the score of a potential match. In all cases, a score of 1 indicates a perfect match between the model and the image.

Furthermore, the score roughly corresponds to the portion of the model that is visible in the image. For example, if the object is 50% occluded, the score (on average) cannot exceed 0.5. This is a highly desirable property because it gives the user a means to select an intuitive threshold for when an object should be considered as recognized. A desirable feature of the above similarity measures is that they do not need to be evaluated completely when object recognition is based on a threshold s_{min} for the similarity measure that a potential match must achieve. Let s_j denote the partial sum of the dot products up to the $j-th$ element of the model. For the match metric that uses the sum of the normalized dot products, this is:

$$s_j = \frac{1}{n}\sum_{i=1}^{j} \frac{\langle d_i', e_{q+p'} \rangle}{\parallel d_i' \parallel \cdot \parallel e_{q+p'} \parallel} \tag{4.5}$$

Obviously, all the remaining terms of the sum are all $\leqq 1$. Therefore, the partial score can never achieve the required score s_{min} if $s_j < s_{min} - 1 + \frac{j}{n}$, and hence the evaluation of the sum can be discontinued after the $j-th$ element whenever this condition is fulfilled. This criterion speeds up the recognition process considerably. Nevertheless, further speed-ups are highly desirable. Another criterion is to require that all partial sums have a score better than s_{min} , i.e. $s_j = s_{min}$. When this criterion is used, the search will be very fast, but it can no longer be ensured that the object recognition finds the correct instances of the model because if missing parts of the model are checked first, the partial score will be below the required score. To speed up the recognition process with a very low probability of not finding the object although it is visible in the image, the following heuristic can be used: the first part of the model points is examined with a relatively safe stopping criterion, while the remaining part of the model points are examined with the hard threshold s_{min}. The user can specify what fraction of the model points is examined with the hard threshold with a parameter g. If $g = 1$, all points are examined with the hard threshold, while for $g = 0$, all points are examined with the safe stopping criterion. With this, the evaluation of the partial sums is stopped whenever $s_j < min(s_{min} - 1 + fj/n, s_{min}j/n)$, where $f = (1 - gs_{min})(1 - s_{min})$. Typically, the parameter g can be set to values as high as 0.9 without missing an instance of the model in the image.

4.3 System Implementation

Shape-based matching enables us to find and localize objects based on a single model image, i.e., from a model. The method we have implemented is robust to noise, clutter, occlusion, and arbitrary non-linear illumination changes. Objects are localized and found even if they are rotated or scaled. Shape-based matching can be applied to standard 8bit gray value images, images with more than 8bit gray value depth, and to color (more generally: multi-channel) images. The system consists of two modules: an offline generation of the model and an online recognition. The model is generated from an image of the object to be recognized.

4.3.1 Model Creation

As mentioned in the beginning of this chapter the logo model images are chosen via image search facility of Google®. Here the selection of the logo is determined regarding some features such as;

- the size
- the shape characteristics
- the clutter on the logo model image.
- the noise in the model image etc.

i.e the model logo is specifed by the user. Alternatively, it can be generated by suitable segmentation techniques or manually from the frame samples.

A prerequisite for a successful matching process is, of course, a suitable model for the object you want to find. A model is suitable if it describes the signifcant parts of the object, i.e., those parts that characterize it and allow discriminating it clearly from other objects or from the background. On the other hand, the model should not contain clutter, i.e., points not belonging to the object (see, e.g., Figure 4.1).

When creating the model, the first step is to select a region of interest (ROI) in the image or select a logo model image that covers the whole image in order to determine the model. In this thesis we prefer to use the images that only contain logos for the sake of simplicity and in order to prevent the previous segmentation work to extract the logo. A region defines an area in an image or more generally a

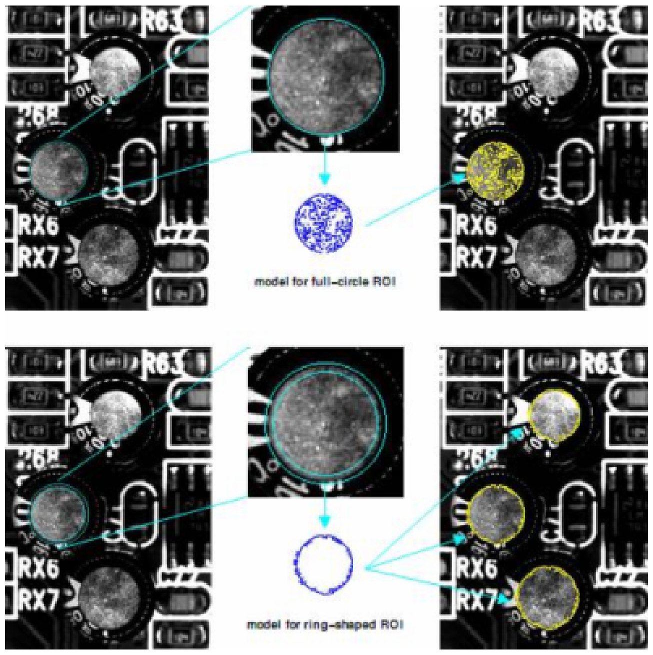


Figure 4.1: Masking the part of a region containing clutter [2].

set of points. A region can have an arbitrary shape; its points do not even need to be connected. Thus, the region of the model can have an arbitrary shape as well. Also we have to note that the ROI should not be too "thin", otherwise it vanishes at higher pyramid levels! As a rule of thumb, an ROI should be $2^{NumLevels-1}$ pixels wide [5]. Figure 4.2 shows logo forms inside the image and Figure 4.3 shows the result of matching for multiple logos.

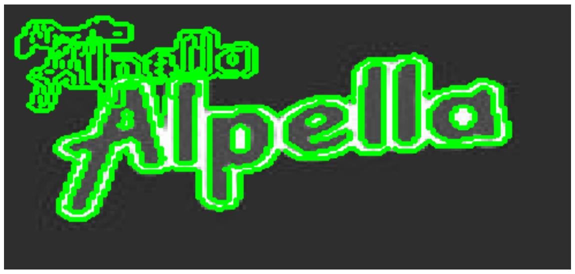
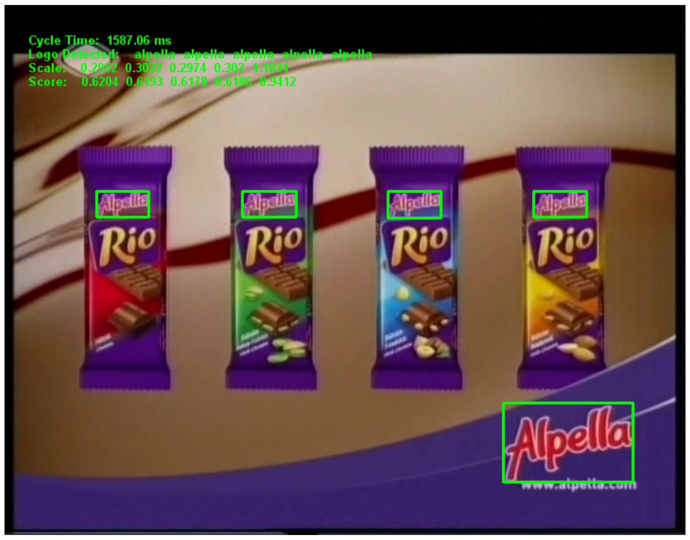


Figure 4.2: Logo forms inside the image.

Here we can also combine the interactive ROI specifcation with image processing. A useful method in the presence of clutter in the model image is to create a first model region interactively and then process this region to obtain an improved ROI. Figure 4.4 shows an example where the task is to locate the arrows. Here we can see



```
Cycle Time:  1587.06 ms
Logo Detected:   alpella  alpella  alpella  alpella  alpella
Scale:    0.2952  0.3022  0.2974  0.302  1.1834
Score:    0.6204  0.6193  0.6178  0.6186  0.9412
```

Figure 4.3: The result of matching for multiple logos.

the results for different threshold values and the model is extracted from the image using morphological technique opening (erosion + dilation), which eliminates small regions. Before this, we fill the inside of the model image since only the boundary points are part of the (original) model region.

Here we have to also note that the ROI used when creating the model also influences the results of the subsequent matching: By default, the center point of the ROI acts as the so-called point of reference of the model for the estimated position, rotation, and scale. The point of reference also influences the search itself: An object is only found if the point of reference lies within the image, or more exactly, within the domain of the image.

4.3.1.1 The Information Stored in the Model

As the name shape-based pattern matching implies, objects are represented and recognized by their shape. Multiple ways exist to determine or describe the shape of an object. Here, the shape is extracted by selecting all those points whose contrast exceeds a certain threshold; typically, the points correspond to the contours of the object.

For the model, those pixels whose contrast, i.e., gray value difference to neighboring pixels, exceeds a threshold specified by the user are selected. In order to obtain

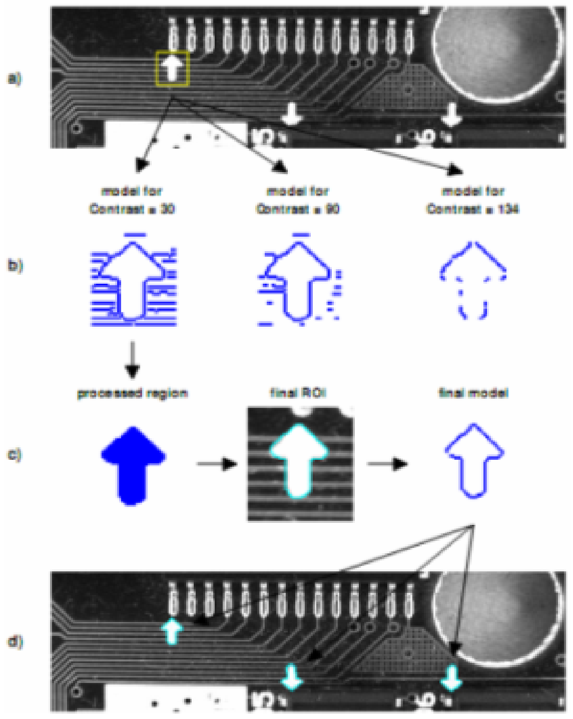


Figure 4.4: a) interactive ROI; b) models for different values of threshold (or contrast); c) processed model region and corresponding ROI and model; d) result of matching [2].

a suitable model, the contrast should be chosen in such a way that the significant pixels of the object are included, i.e., those pixels that characterize it and allow to discriminate it clearly from other objects or from the background. Obviously, the model should not contain clutter, i.e., pixels that do not belong to the object.

In some cases it is impossible to find a single value for threshold that removes the clutter but not also parts of the object. Figure 4.5 shows an example where the task is to create a model for the outer rim of a drill-hole: If the complete rim is selected, the model also contains clutter (Figure 4.5a); if the clutter is removed, parts of the rim are missing (Figure 4.5b).

To solve such problems, the threshold provides two additional methods: hysteresis thresholding and selection of contour parts based on their size. Hysteresis thresholding uses two thresholds, a lower and an upper threshold. For the model, first pixels that have a contrast higher than the upper threshold are selected; then, pixels that have a contrast higher than the lower threshold and that are connected

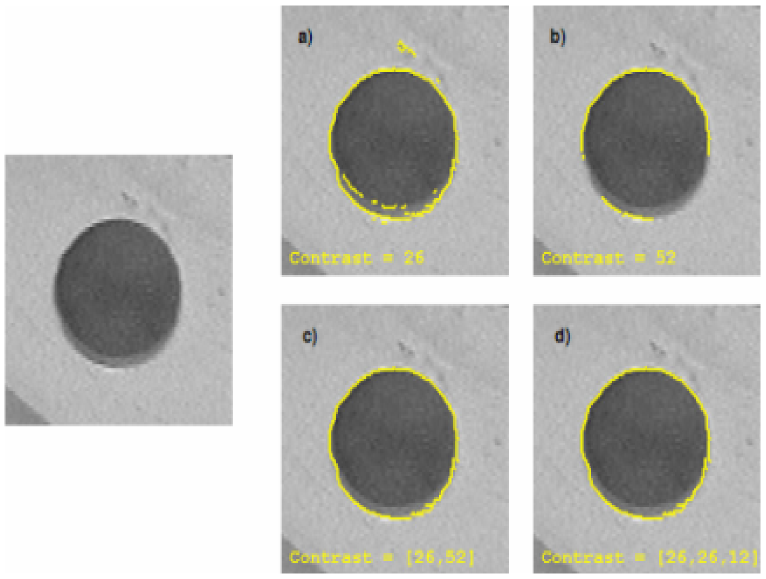


Figure 4.5: Selecting significant pixels via threshold (i.e. contrast): a) complete object but with clutter; b) no clutter but incomplete object; c) hysteresis threshold; d) minimum contour size [2].

to a high-contrast pixel, either directly or via another pixel with contrast above the lower threshold are added. This method enables us to select contour parts whose contrast varies from pixel to pixel. As seen in the example in Figure 4.1, with a hysteresis threshold we can create a model for the complete rim without clutter.

The second method to remove clutter is to specify a minimum size, i.e., number of pixels, for the contour components. Figure 4.5 d shows the result for the example task.

4.3.1.2 Using Subsampling to Speed up the Search

To speed up the recognition process, the model is generated in multiple resolution levels, which are constructed by building an image pyramid from the original image as shown in Figure 4.2. The number of pyramid levels l_{max} is chosen by the user.

Here the pyramid consists of the original, full-sized image and a set of down sampled images. For example, if the original image (first pyramid level) is of the size 600x400, the second level image is of the size 300x200, the third level 150x100, and so on. The object is then searched first on the highest pyramid level, i.e., in the smallest image. The results of this fast search are then used to limit the search in the next pyramid image, whose results are used on the next lower level until the lowest level is reached. Using this iterative method, the search is both fast and accurate. Figure 4.2 depicts 4 levels of an example image pyramid together with the corresponding model regions.

As a result of experiments it is recommended to choose the highest pyramid level at which the model contains at least 10-15 pixels and in which the shape of the model still resembles the shape of the object.

4.3.1.3 Allowing a Range of Orientation and Scale

As explained in the previous section, each resolution level consists of all possible rotations and scaling of the model, where thresholds ϕ_{min} and ϕ_{max} for the angle and σ_{min} and σ_{max} for the scale are selected by the user. The step length for the discretization of the possible angles and scales can either be done automatically by a method similar to the one described in [29] or be set by the user. In higher pyramid levels, the step length for the angle is computed by doubling the step length of the next lower pyramid level.

During the matching process, the model is searched for in different angles within the allowed range, at steps specified by the user. It is also possible to determine the angle step to obtain the highest possible accuracy by determining the smallest rotation that is still discernible in the image. The underlying algorithm is explained in Figure 4.6: The rotated version of the cross-shaped object is clearly discernible from the original if the point that lies farthest from the center of the object is moved by at least 2 pixels. Therefore, the corresponding angle ϕ_{opt} is calculated as follows:

$$d^2 = l^2 + l^2 - 2 \cdot l \cdot l \cdot cos\phi \Rightarrow \phi_{opt} = arccos(1 - \frac{d^2}{2 \cdot l^2}) = arccos(1 - \frac{2}{l^2}) \qquad (4.6)$$

with l being the maximum distance between the center and the object boundary and $d = 2$ pixels. For some models, such estimated angle step size is still too large. In these cases, it is divided by 2 automatically.

Also in order to get better results, the value chosen for angle step should not deviate too much from the optimal value ($1/3\phi_{opt} \leqslant \phi \leqslant 3\phi_{opt}$). We have to also

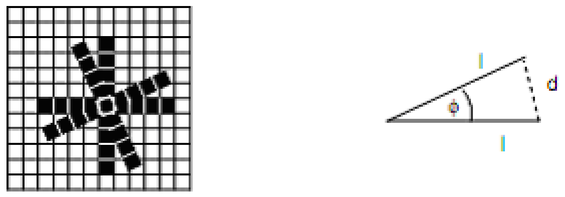


Figure 4.6: Determining the minimum angle step size from the extent of the model [2].

note here that choosing a very small step size does not result in increased angle accuracy [5].

Similar to the range of orientations, it is possible to specify an allowed range of scale. We can allow for scaling in two forms:

- identical scaling in row and column direction (isotropic scaling)
- different scaling in row and column direction (anisotropic scaling)

For isotropic scaling, we specify the range of scales with the required parameters as minimum scale, maximum scale, and scale step of the operator. For anisotropic scaling, we can determine the six scale parameters separately for rows and columns instead of the three above.

Again, it is better to reduce the limit the allowed range of scale as much as possible in order to speed up the search process.

Similar to the angle step value, scale step (or the equivalents for anisotropic scaling), can be determined by using the smallest scale change that is still discernible in the image. Similarly to the angle step size the center of the object is moved by at least 2 pixels. Therefore, the corresponding scale change $\Delta\sigma_{opt}$ is calculated as follows:

$$\Delta\sigma = \frac{d}{l} \Rightarrow \Delta\sigma_{opt} = \frac{2}{l} \tag{4.7}$$

with l being the maximum distance between the center and the object boundary and $d = 2$ pixels. For some models, the such estimated scale step size is still too large. In these cases, it is divided by 2 automatically.

Also in order to get better results the value chosen for scale step should not deviate too much from the optimal value ($1/3\sigma_{opt} \leqslant \sigma \leqslant 3\sigma_{opt}$). We have to also note here that choosing a very small step size does not result in increased angle accuracy [5].

As explained in detail above, the rotated and scaled models are generated by rotating and scaling the original image of the current pyramid level and performing the feature extraction in the rotated image. This is done because the feature extractors may be anisotropic, i.e., the extracted direction vectors may depend on the orientation of the feature in the image in a biased manner. If it is known that the feature extractor is isotropic, the rotated models may be generated by performing the feature extraction only once per pyramid level and transforming the resulting points and direction vectors. The feature extraction can be done by a number of different image processing algorithms that return a direction vector for each image point. One such class of algorithms are edge detectors, e.g, the Sobel or Canny [33] operators. Another useful class of algorithms are line detectors [34]. Finally, corner detectors that return a direction vector, e.g. [35], could also be used. Because of runtime considerations the Sobel filter is used in the current implementation of the object recognition system. One disadvantage using this filter maybe that in the video frames noise poses a significant problem due to the recording equipment. Therefore we try to select model images with less clutter and high resolution as possible. To recognize the model, an image pyramid is constructed for the image in which the model should be found. For each level of the pyramid, the same filtering operation that was used to generate the model, e.g., Sobel filtering, is applied to the image. This returns a direction vector for each image point. Note that the image is not segmented, i.e., thresholding or other operations are not performed. This results is robust to illumination changes.

4.3.2 Optimizing the Search Process

To identify potential matches, an exhaustive search is performed for the top level of the pyramid, i.e., all precomputed models of the top level of the model resolution hierarchy are used to compute the similarity measure for all possible poses of the model. A potential match must have a score larger than a user-specified threshold s_{min} and the corresponding score must be a local maximum with respect to neighboring scores. As described above, the threshold s_{min} is used to speed up the search by terminating the evaluation of the similarity measure as early as possible. With the termination criteria, this seemingly brute-force strategy actually becomes extremely efficient. On the average, about 9 pixels of the model are tested for every pose on the top level of the pyramid.

After the potential matches have been identified, they are tracked through the resolution hierarchy until they are found at the lowest level of the image pyramid. Various search strategies like depth-first, best-first, etc., have been examined. It turned out that a breadth-first strategy is preferable for various reasons, most notably because a heuristic for a best-first strategy is hard to define, and because depth-first search results in slower execution if all matches should be found.

Once the object has been recognized on the lowest level of the image pyramid, its position and rotation are extracted to a resolution better than the discretization of the search space, i.e., the translation is extracted with subpixel precision and the angle and scale with a resolution better than their respective step lengths. This is done by fitting a second order polynomial (in the four pose variables) to the similarity measure values in a $3 \times 3 \times 3 \times 3$ neighborhood around the maximum score. The coefficients of the polynomial are obtained by convolution with 4D facet model masks. The corresponding 2D masks are given in [34]. They generalize to arbitrary dimensions in a straightforward manner.

When comparing a part of a search image with the model, the matching process calculates the so-called score, which is a measure of how many model points could be matched to points in the search image (ranging from 0 to 1).

In addition the search algorithm is broken off the comparison of a candidate with the model when it seems unlikely that the minimum score will be reached. In other words, the goal is not to waste time on hopeless candidates. This is named as the “greediness”, however, can have unwelcome consequences. In some cases a perfectly visible object is not found because the comparison “starts out on a wrong foot” and is therefore classified as a hopeless candidate and broken off. The user can adjust the greediness of the search, i.e., how early the comparison is broken off, by selecting values between 0 (no break off: thorough but slow) and 1 (earliest break off: fast but unsafe).

4.3.3 Least-Squares Pose Refinement

While the pose obtained by the extrapolation algorithm is accurate enough for most applications, in some applications requiring an even higher accuracy the least-squares pose refinement is used. This can be achieved through a least-squares adjustment of the pose parameters. To achieve a better accuracy than the extrapolation, it is necessary to extract the model points as well as the feature points in the image with subpixel accuracy. If this would not be done, the image and model points would be separated radially by about 0.25 pixels on average if each model point is matched to its closest image point. However, even if the points are extracted with subpixel accuracy, an algorithm that performs least-squares adjustment based on closest point distances would not improve the accuracy much since the points would still have an average distance significantly larger that 0 tangentially because the model and image points are not necessarily sampled at the same points and distances. Because of this, the proposed algorithm finds the closest image point for each model point and then minimizes the sum of the squared distances of the image points to a line defined by their corresponding model point and the corresponding tangent to the model point, i.e., the directions of the model points are taken to be correct and are assumed to describe the direction of the object's border. If, for example, an edge detector is used, the direction vectors of the model are perpendicular to the object boundary, and hence the equation of a line through a model point tangent to the object boundary is given by $t_i(x - x_i) + u_i(y - y_i) = 0$. Let $q_i = (v_i, w_i)^T$ denote the matched image points corresponding to the model points p_i. Then, the following function is minimized to refine the pose a:

$$d(a) = \sum_{i=1}^{n} [t_i(u_i(a) - x_i) + u_i(w_i(a) - y_i)]^2 \Rightarrow min \tag{4.8}$$

The potential corresponding image points in the search image are obtained by non-maximum suppression only and are extrapolated to subpixel accuracy [36]. By this, a segmentation of the search image is avoided, which is important to preserve the invariance against arbitrary illumination changes. For each model point the corresponding image point in the search image is chosen as the potential image point with the smallest euclidian distance using the pose obtained by the extrapolation to transform the model to the search image. Because the points in the search image are not segmented, spurious image points may be brought into correspondence with model points. Therefore, to make the adjustment robust, only correspondences with a distance smaller than a robustly computed standard deviation of the distances are used for the adjustment.

Since 4.8 results in a linear equation system when similarity transformations are considered, one iteration suffices to find the minimum distance. However, since the point correspondences may change by the refined pose, an even higher accuracy can be gained by iterating the correspondence search and pose refinement. Typically, after three iterations the accuracy of the pose no longer improves.

4.3.4 Using The Results of Matching

The results of the trademark logo recognition provides us the;

- position (row and column) of the match,
- orientation angle,
- scaling factor
- matching score

of the trademark logo found in the current frame.

The matching score, which is a measure of the similarity between the model and the matched object, can be used "as it is", since it is an absolute value. Here we can determine the position, orientation, scale and the more important is the matching score with the current frame.

Chapter 5

Scale Invariant Feature Transform

5.1 Theoretical Background on Scale Invariant Feature Transform

Scale-invariant feature transform (SIFT) is one of several computer vision algorithms for extracting distinctive features from images [3]. SIFT interest points are based on a Difference of Gaussian detector. Its high-dimensional descriptor vector relies on gradient histograms. In addition to key point location, Lowe's method [3] also provides a way to extract features which are invariant to scale, rotation, and translation. The name Scale-invariant Feature Transform was chosen, as the algorithm transforms image data into scale-invariant coordinates. Following are the major stages of computation used to generate the set of image features:

- Scale-space extrema detection
- Keypoint localization
- Orientation assignment
- Keypoint descriptor

5.1.1 Detection of Scale-Space Extrema

The first stage of keypoint detection is to determine location and scales that can be repeatably assigned under differing views of the same object. Detection of these locations can be done by searching for stable features across all possible scales, using a continuous function scale known as scale space [37]. Using the Gaussian function as the scale-space kernel, the scale space of an image can be defined as a function $L(x, y, \sigma)$, which is produced from the convolution of the image, $I(x, y)$ with a variable-scale Gaussian, $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{5.1}$$

where $*$ is the convolution operator in x and y and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{5.2}$$

In order to detect stable keypoint locations in the scale-space, difference-of-Gaussian (DoG) function is used, which is computed from the difference of two nearby scales that are separated by a constant multiplicative factor k:

$$\begin{aligned} D(x, y, \sigma) &= (D(x, y, k\sigma) - D(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{5.3}$$

Figure 5.1 shows an efficient way to construct $D(x, y, \sigma)$. In the left column, the initial image is incrementally convolved by Gaussians of different scales, which are separated by a constant factor k in the scale space to build an image pyramid. Each octave of scale space is divided into an integer number s, of intervals, so $k = 2^{1/s}$. Adjacent image scales are subtracted to produce the DoG images as shown on the right. Once one octave is complete, the Gaussian image in the current octave that has a variance of 2σ is sub-sampled. It is often problematic to downsize an image by an arbitrary scale; therefore scaling down can be mimicked by convolving the image with a Gaussian and sub-sampling by taking every second pixel in each row and column.

Once the DoG images are computed, the local maxima and minima are to be detected. In the scale-space, each pixel has 26 neighbors, eight in the current scale, nine in the scale below and nine in the scale above (See Figure 5.2). Each pixel is compared with its 26 neighbors and is selected as a local extrema if it is larger than all of its neighbors or smaller then all of them. Since most pixels are eliminated after a few comparisons, the cost of detecting is much lower than building of the pyramid.

5.1.2 Keypoint localization

Once a set of potential keypoints have been found in the image by comparing a pixel to its neighbors in the scale space, for each keypoint, its sub-pixel and sub-space location (x, y, σ) are determined. This information allows points with low contrast to be rejected. Furthermore, keypoints that lie on edges are needed to be removed as well, because edges are poor keypoints since their location cannot be determined well along the edge. Poorly designed peaks in the DoG function will have a large principal curvature across the edge but a small one in the perpendicular direction. The $2x2$ Hessian matrix H can be computed at the location and the scale of the keypoint:

$$H = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{pmatrix} \tag{5.4}$$

The derivatives are estimated by taking differences of neighboring sample pixels. The eigenvalues of are proportional to the principal curvature of H. Based on the proportion of the eigenvalues, edge like features can be detected and eliminated.

5.1.3 Orientation Assignment

Finally by calculating the dominant orientation of each keypoint, the keypoint descriptor represented with this orientation becomes invariant to image rotation. First the Gaussian image in the octave with the closest scale is selected, so that all computations are carried out in a scale-invariant manner. For each keypoint the gradient magnitude, $m(x, y)$ and orientation, $\theta(x, y)$ are computed using pixel differences for the current scale $L(x, y)$.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{5.5}$$

$$q(x,y) = tan^-1((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \tag{5.6}$$

An orientation histogram is build using the gradient orientations within a region around the keypoint. The histogram contains 36 bins, separating the 360 degree orientations into regions of 10 degrees. Each sample is weighted by its gradient

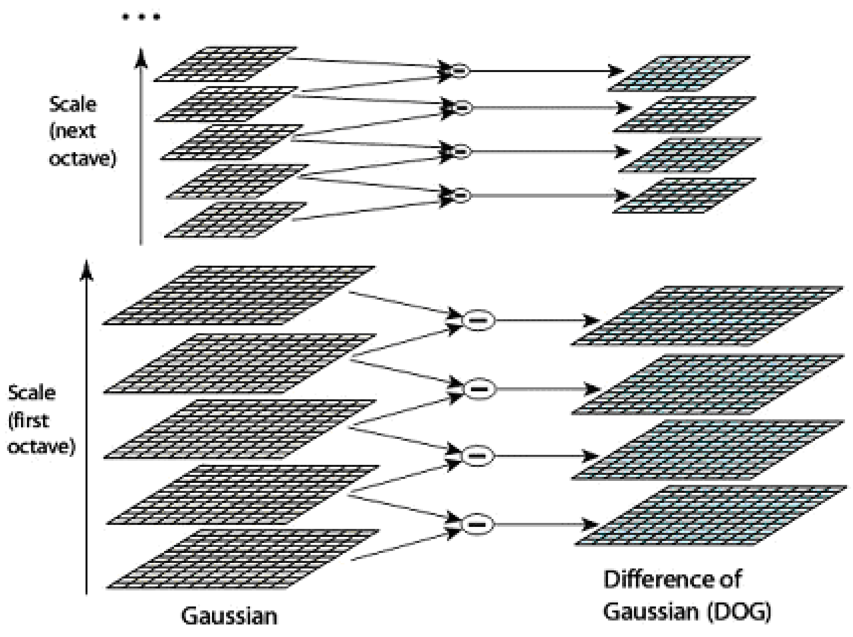


Figure 5.1: Difference of Gaussians are computed from a pyramid of Gaussians. Adjacent Gaussian images are subtracted to produce a difference of Gaussian (DoG) images [3].

magnitude before being added to the histogram. Significant peaks in the histogram correspond to dominant directions in the gradient. The highest peak in the histogram is selected to be the orientation of the keypoint, however if other local peaks within 80% of the highest peak are detected, a new keypoint with that orientation is created. For any keypoints with multiple peaks, multiple keypoints will be created. This improves the orientation invariance of the SIFT algorithm.

5.1.4 Local Image Descriptor

The previous operations extract image location, scale and orientation for keypoints from a given image. The next step is to describe a local image region in a manner which is invariant to scale and orientation as well as to changes in illumination and to 3D viewpoint. The previous operations extract image location, scale and orientation for keypoints from a given image. The next step is to describe a local image region in a manner which is invariant to scale and orientation as well as to changes in illumination and to 3D viewpoint.

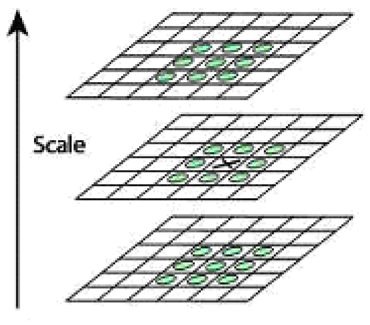


Figure 5.2: Maxima and minima of the DoG images are detected by comparing the pixel of interest by its 26 neighbors of the current and adjacent scales [3].

Figure 5.3 shows how keypoint descriptors are computed. In a region around the keypoint, image gradient magnitudes and orientations are computed. The gradient magnitudes are weighted by a Gaussian centered on the keypoint location. In a typical application the window is divided into $4 \times 4 = 16$ subregions and for each subregion, an orientation histogram is build and the histograms are placed at the center of the subregion. Boundary effects, in which the descriptor abruptly changes as a subregion shifts smoothly from one histogram to another or from one orientation to another, are avoided by interpolation where each gradient votes for an orientation in its neighboring histograms. The vote is weighted by $1 - d$ for each dimension, where d is the distance to the histogram. The coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation in order to maintain orientation invariance. The 4×4 descriptors computed from a 16×16 sample array provide a 128-dimensional descriptor array.

Finally, in order to reduce the effects of illumination changes, the feature vector is normalized to unit length. Since a change in image contrast where each pixel value is multiplied by a constant will also reflect the gradients by the same constant, normalizing the feature vector will remove the effect of contrast change. A brightness change where a constant is added to every image pixel will not change the gradient magnitudes since they are computed from pixel differences. Therefore, affine changes in illumination do not affect the normalized keypoint descriptor. In order to reduce the affects of non-linear illumination changes, which can occur due to camera change

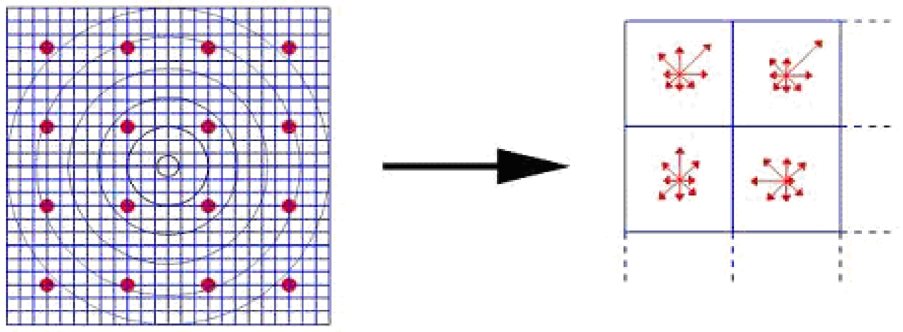


Figure 5.3: SIFT Descriptor. For each pixel around the keypoint gradient magnitudes and orientations are computed. These samples are weighted by a Gaussian and accumulated into 16 orientation histograms for the 16 subregions [3].

or due to a change in illumination orientation or amount that affect 3D surfaces, the influence of large gradient magnitudes in the unit feature vector are thresholded to be no longer than 0.2 and then the feature vector is renormalized to unit length. This is because nonlinear illumination changes are more likely to effect gradient magnitudes rather than gradient orientations, therefore by thresholding the magnitude, more emphasis is put onto orientations. The value 0.2 is determined experimentally by Lowe [3].

5.2 Matching

The best candidate match for each keypoint is found by identifying its nearest neighbor in the database of keypoints from training images. The nearest neighbor is defined as the keypoint with minimum Euclidean distance for the invariant descriptor vector. However, many features from an image will not have any correct match in the training database because they arise from background clutter or were not detected in the training images. Therefore, it would be useful to have a way to discard features that do not have any good match to the database. A global threshold on distance to the closest feature does not perform well, as some descriptors are much more discriminative than others. A more effective measure is obtained by comparing the distance of the closest neighbor to that of the second-closest neighbor. If there are multiple training images of the same object, then we define the second-closest neighbor as being the closest neighbor that is known to come from a different object than the first, such as by only using images known to contain different objects. This measure performs well because correct matches need to have the closest neighbor significantly closer than the closest incorrect match to achieve reliable matching. For false matches, there will likely be a number of other false matches within similar distances due to the high dimensionality of the feature space. We can think of the second-closest match as providing an estimate of the density of false matches within this portion of the feature space and at the same time identifying specific instances of feature ambiguity.

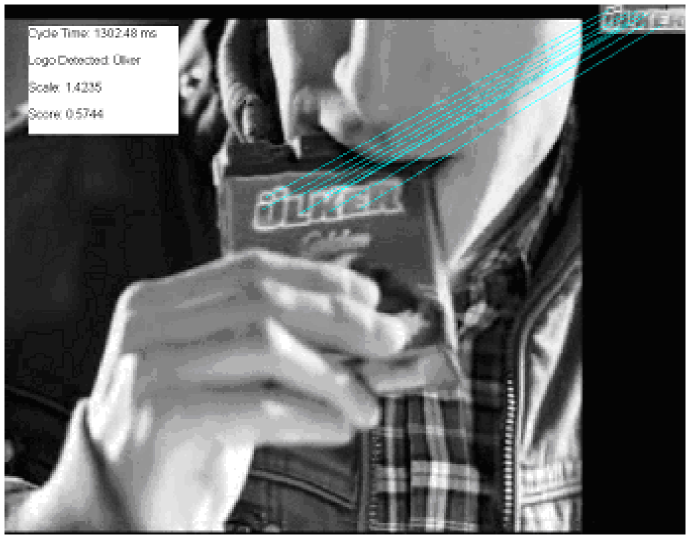

```
Cycle Time: 1302.48 ms
Logo Detected: Ülker
Scale: 1.4235
Score: 0.5744
```

Figure 5.4: An example of matching logo in the video frame.

Although the SIFT algorithm is advantageous for the recognition of static and rigid objects under small changes of illumination, it does not work under large illumination changes and non-rigid deformations [38]. It is also observed through this study as shown in Chapter 6 for “pantene” example.

5.3 SIFT on GPU

The parallel computing nature and programmability pipeline makes graphics processing unit (GPU) a powerful tool for data parallel computation problems, and it has been widely used for general purpose computation [39]. In this part of the thesis we utilize the computing power of a GPU to speed up the scale invariant transform

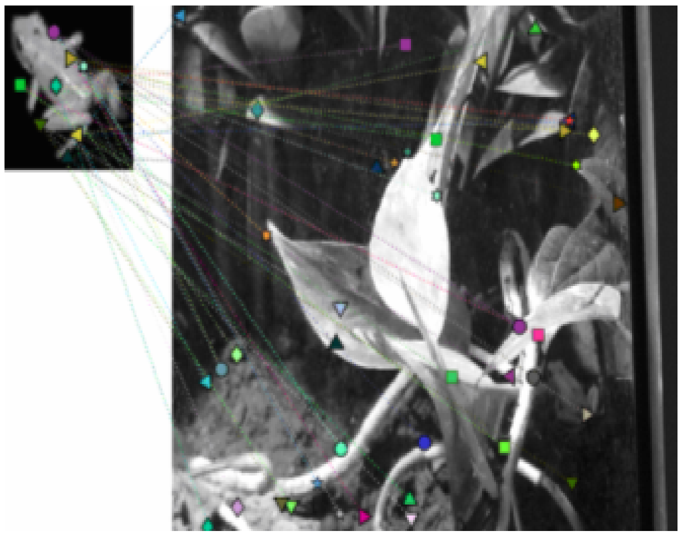


Figure 5.5: SIFT performance under large illumination variations.

computations.

SIFT detects the local maxima and minima of difference of Gaussian in the Gaussian scale space. Local dominant gradient orientations are then computed for each feature point, and sub-pixel localization is applied. Descriptors are then generated from the scale and orientation normalized image patches for each feature.

The first part, scale space computation, can be cast to a pixel parallel computation. It runs Gaussian filters on input images to get each pixel of new filtered images, and GPU can use a fragment shader to compute multiple pixels simultaneously. The second part, localization, orientation computation, and descriptor generation, can be seen as a feature parallel computation. Each feature can also be mapped to a pixel to run parallel on GPU.

Lowe's SIFT (Scale Invariant Feature Transform) [3] detect similarity invariant features in gaussian scale space of images, and it has been successfully applied in many computer vision problems [40]. By exploiting the data parallel computing feature of GPU, scale invariant feature transform can run much faster on GPU than on central processing unit (CPU).

Here we implemented the GPU SIFT using the feature extraction method described in [39]. Here the traditional GPU shaders are chosen as the implementation tool instead of compute unified device architecture (CUDA), considering the fact that images are easily mapped to textures on GPU. The level images in the scale space are intuitively stored as pixel-by-pixel mapped texture. Shown in Figure 5.6, the four color channels RGBA are used to store intensity, difference of Gaussian, gradient magnitude, and gradient orientation respectively. To save memory usage, feature list is used in this implementation. The feature list is also stored as textures as shown in Figure 5.6.

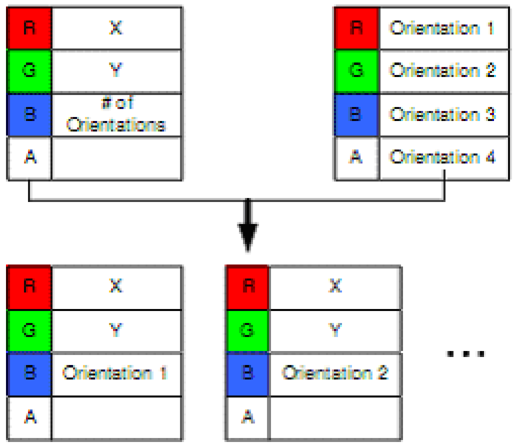


Figure 5.6: Storage of feature list as textures.

Features on different levels are stored separately and the scale information does not need to be stored. After the first stage of feature detection, a feature list texture that saves feature location and orientation count are used, and then the feature list texture is reshaped to make a list of features with separate orientations. A point that needs to mention is that all the feature list generation and feature list reshaping are implemented on GPU.

Similar with [41], separable gaussian filtering is used to run filtering horizontally and vertically separately. This is necessary to achieve good performance, because the Gaussian kernel needs to be very large for large σ. 6σ is used as the filter width, when the number of DOG levels is 3. Then largest Gaussian filter σ will be 3.0, and it will require a 19×19 Gaussian kernel. Using separable filter will save a lot of texture fetches and also reduces the shader code size.

Gaussian filter shaders are generated on the fly according to the parameter user inputs, each with different different sizes and kernels. Multiple texture coordinate feature of OpenGL is used, and when the number of coordinates is more than 8, they will be computed automatically in shaders.

Figure 5.7 demonstrates the two stage Gaussian filter. The second pass, by carefully writing back the temporary intensity to the original color channel, can read and write the same texture. The experiments show that reading and writing the same texture is faster than PingPong [42] which is a technique used with a method of using pixel data that has been rendered as a texture (RTT) to avoid reading and writing the same buffer simultaneously, instead bouncing back and forth between a pair of buffers. It can be explained by that PingPong requires more switching of texture caching. Difference of Gaussian is also computed in the second pass.

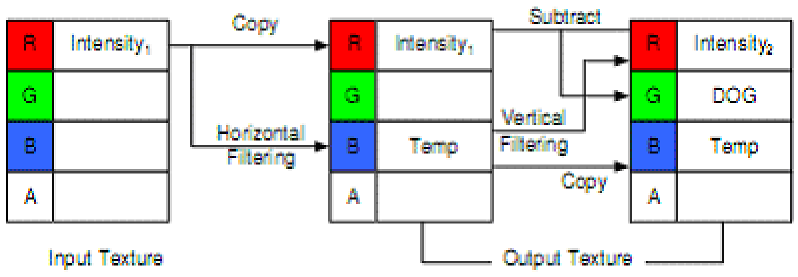


Figure 5.7: Two pass of gaussian filter that uses texture from destination.

After one octave is computed, sub-sampling is used to get the first several level images of the next octave. For example when the level range is from -1 to s + 2, the scale doubles every s steps. There are 3 pairs of doubling in one octave, and the highest 3 level of an octave can be used to generate the first 3 level of the next octave. One restriction is that the filter size cannot be truncated for higher level, other wise the Gaussian will be inaccurate for sub-sampling. When subsampling more than one level, both intensity and DOG can be generated from sub-sampling, and this can save some time on filtering. This trick hasn't been seen in other SIFT implementations.

5.3.1 Keypoint Detection

Keypoint detection needs to compare the DOG of a pixel with its 26 neighbours in the scale space. This step is split into intra-level suppression and inter-level to save texture fetching. As shown in Figure 5.8, the first pass will compare the DOG value of a pixel with its 8 neighbours, and save whether the point is a local minimum and local maximum to an auxiliary texture. The maximum and minimum of the 9 pixels are also stored in the auxiliary texture. Gradient magnitude and orientation is also computed in this pass. Edge elimination is also applied in this pass to delete the features that are on edges.

In the second pass, early-z is first applied to exclude the pixels that are already filtered out in the first pass, then each pixel is compared with the maximum or minimum value of its 2 neighbor in the scale space. A point is the maximum in the 3x3x3 cube only when it is identified as an intra-level local maximum and it is larger than the maximum values in its two neighbours. Similar thing applies to minimum.

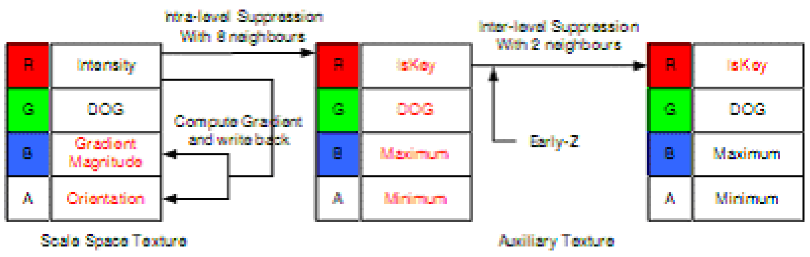


Figure 5.8: Keypoint Detection.

5.3.2 Feature List Generation

Method in [43] is used here to generate feature lists on GPU. Our implementation used the full 4 color channel to build the histogram pyramid, which can be seen as pointer textures and the feature list generated by tracking down the histogram pyramid. For every image, only one pixel at the top of the histogram pyramid needs to be read back, and the number of features is the sum of the four channels. This method can avoid the read-back of textures, and also avoid the upload of feature list.

5.3.3 Orientation Computation

This step computes the orientation candidates for each feature. It first obtain an weighted orientation histogram in the circular window of radius 3s, then apply smoothing on the histogram, and finally the angles whose voting is larger than

0.8 times the maximum are outputted. The 36 angle for orientation histogram is implemented as 9 float4/vec4. Since GPU arrays does not support dynamic indexing, a binary search of index is used here to locate the expected 4-angle bin. Then this bin is added with a voting vector as follows

$bin+ = weight * float4(fmod(idx, 4) == float4(0, 1, 2, 3))$

With this kind of 4 angle bins, smoothing can be easily applied with a larger window. The smoothing in [44] runs $(1, 1, 1)/3$ filtering for 6 times, and, because four values are stored in one bin, it can be implemented as running $(1, 3, 6, 7, 6, 3, 1)/27$ filtering for twice.

Finally, the orientations are written to the orientation texture, and the numbers of features are writing to the original feature texture. Then the point list generation method as in the feature list generation is used again to reshape the feature list, and this step is shown in Fig 5.7. Instead of assigning different point location in the last step, different feature orientations are assigned to different feature candidates.

A display list can also be generated on GPU without reading back the features. SIFT features here are displayed as scaled and rotated squares. A texture with 4 times space is allocated for saving the output vertices, and a shader will automatically compute the feature index of the point, and also the sub-index in the rectangle. The point can then be rotated and translated according to the feature orientation and scale. Figure 5.9 shows this vertex generation. The vertex result can then be copying to a Vertex BufferObject to demonstrate SIFT features. Figure 5.9 shows an example of the result.

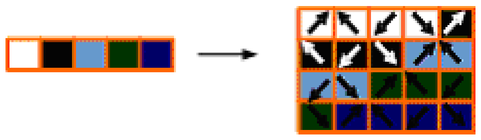


Figure 5.9: Display vertex generation.

Chapter 6

Experiments

The experiments are run on seven different datasets of commercial videos that is a total of 53668 frames lasting 36 minutes and 24 seconds. The broadcast videos are recorded from Turkish TV channels to the computer using the TV card that can record the current TV broadcast.

After data collection, we worked on the classification of video blocks. For this purpose we have designed the commercial block detector as explained in Chapter 3. Here we determined the transition of commercial blocks by checking the occurence of at least two consecutive black frames. The commercial block detector, using the mean and the variance of the gray values of the frames in process, was able to detect all of the blocks with a performance of 100%.

Through the data collection process, we generally attempted to collect videos with problematic trademark logo images. Trademark logo images retained are the ones including noisy images or the logos not studied extensively in the previous works. Also, most of the images are blurry due to their nature of motion in the commercial video since the attraction of the trademark logo is mostly carried out by motions or static images that stay on the screen for a long time. For example, the transparent logos have been the very problematic ones to recognize in previously used methods.

After analyzing our dataset we selected 50 different trademark logos for the recognition process. Transparent, low resolution and noisy logos are included in the selection since they have generally been one of the important problems of trademark logo recognition.

We then focused on the recognition algorithm and implemented the shape based object recognition algorithm, as explained in Chapter 3, for recognizing the trademark logos in video frames. The algorithm is explained in detail in Figure 6.1. For the test process we have firstly generated the models of the logos in the offline phase and saved them in memory to use them for online recognition. The edge points are saved in the model for the matching process. The algorithm is triggered by the detection of the black frame; the recognition process is carried on since the next black frame block is detected.

In the next phase we tried to implement another object recognition algorithm for recognizing the logos. We decided to implement the SIFT algorithm due to its high performance in object recognition. The block diagram of the system is as shown in Figure 6.2 below. In the offline phase the keypoints of logos are generated and saved in memory; in the online phase, the keypoint of the current frame is generated and matched with the keypoints of all 50 logos in the database. The SIFT features for the logos and the current frame are extracted, creating two sets of features. Feature matching is then performed between the features in each frame for all logos. For each feature in the logo, the nearest neighbor is found in the frame; in order to find the nearest neighbor, we use some distance measure based on feature descriptors or other properties, such as scale and orientation. In order to speed up the matching, the SIFT feature extraction and matching is done on a GPU card. The matching keypoints are normalized and a matching score is computed for a frame. If the score is higher than a specified threshold, the logo is assigned to the current commercial video; this process continues until the video ends. Here the "Logo Found?" box in the block diagram of the system is done by using the SIFT based matching. Also the cycle time is computed for a frame and scale and orientation is computed by using the locations of the matched keypoints.

For the SIFT algorithm, we have firstly used David Lowe's binary implementation which was coded in Matlab. The recognition results were promising, but the speed of the algorithm was very low. With these results in hand, we then coded the algorithm in C++ and tested in again. The recognition performances were the same and we were able to speed the computations up to 7-8 times as shown in Table 6.3. However, the timing results were still far away from those of the recognition speed of the shape based method. Then we used SIFT on a GPU in order to speed up the computations. With the implementation of the SIFT algorithm on the GPU, we could speed up the process around 50 times fasters regarding the implementation of Lowe in Matlab.

In addition, every 30^{th} frame is processed in order to speed up the recognition algorithm. The results below show the performance analysis for each of the 30th frames. This sampling ratio is determined by experiment. During the tests and we have also observed that a logo stays on screen at least 30 frames which is enough for us to recognize the logo.

The experiments were run on a $IntelPentium^{®}$ 3.2 GHz PC with NVidia GeForce 8400 GS GPU, $Microsoft\ Windows^{®}$ XP SP2 operating system and 3 GB RAM.

The following paragraphs we summarize the performances of the shape based and the SIFT based method under the most difficult situations, namely motion blur, transparency, low resolution, occlusion, illumination, scale and perspective transformations.

Tables 6.1, 6.2, 6.3 and 6.4 summarize the results of our experiments. The first column on each table shows the dataset number and the second column gives the number of frames which include the logo images from our library. The third column on the tables gives the the number of frames where the predefined logo was not recognized. The fourth column shows the number of frames where the false positives occured which means the logo recognition result was positive for a logo model from the library where in fact no logo was present inside the frame. In addition, fifth and sixth columns explain the processing cycles for all 50 logos and the number of frames processed (number of sampled frames for recognition) through the dataset, respectively.

From the tables we can conclude that the shape based algorithm in general performs better than the SIFT based method in recognition while the SIFT on GPU based method always performs faster than the shape based method regarding the computation time. Figure 6.3 shows that the shape based method performs very well under motion blur due to the powerful extraction of edge points, i.e. features. Dataset 5 includes a channel logo which is transparent and with the shape based method we can effectively identify the logo. Figure 6.4 shows that the shape based algorithm also performs better than the SIFT based method when the logo is transparent. The transparent logos are the most problematic ones and they have not been studied extensively in the literature. Figure 6.5 shows that the shape based algorithm gives better results in the cases of small and low resolution frames. This is also observed through all the datasets. For these small and low resolution logos we were able to extract the features by the help of the powerful edge detection algorithms which resulted in better performance through these cases. Figure 6.6 also shows that both of the algorithms performs similar in case of occlusion and here the SIFT on GPU also has the computation speed advantage. In addition we can conclude from Figure 6.7 that the shape based algorithm performs better in large illumination variations. Both algorithms in general perform similar for scale variations in general cases but if we reduce the size lower than a certain threshold the shape based algorithm still gives better results as shown in Figure 6.8. This robustness indicates that shape based algorithm is better than the SIFT based algorithm in scale changes. In addition Figure 6.9 shows that both of the algorithms perform similar in recognition in case of perspective transformation.

The results shown in Figures 6.3 - 6.9 also indicate the general charateristics of the logo recognition system developed through the thesis. In other words, we can conclude about the general performances of the logo recognition algorithms from these figures.

6.1 Experimental Results of Shape Based Method

The results of the designed algorithm for shape based logo recognition are given in Table 6.1.

Table 6.1: Experiment results of shape based matching

Dataset	Logo Including Frames	Not Recognized Frames	Incorrectly Recognized Frames	Average Cycle Time (for 50 logos)	# Processed Frames
1	107	16	4	2.518 ms	325 out of 8130
2	93	13	12	2.640 ms	271 out of 6779
3	90	4	18	2.480 ms	211 out of 5281
4	85	4	15	2.594 ms	312 out of 7808
5	124	5	8	2.396 ms	124 out of 3123
6	164	21	10	2.497 ms	398 out of 9972
7	125	24	14	2.672 ms	503 out of 12575

6.2 Experimental results of SIFT based method

The results of the designed algorithm for SIFT based logo recognition are given in Tables 6.2, 6.3 and 6.4.

Table 6.2: Experiment Results of SIFT binaries in Matlab (code is provided by Lowe).

Dataset	Logo Including Frames	Not Recognized Frames	Incorrectly Recognized Frames	Average Cycle Time (for 50 logos)	# Processed Frames
1	107	35	56	100.053 ms	325 out of 8130
2	93	2	67	107.482 ms	271 out of 6779
3	90	11	31	106.406 ms	211 out of 5281
4	85	14	83	104.564 ms	312 out of 7808
5	124	0	82	108.416 ms	124 out of 3123
6	164	14	78	102.307 ms	398 out of 9972
7	125	5	54	108.346 ms	503 out of 12575

Table 6.3: Experiment results of C++ implementation of SIFT.

Dataset	Logo Including Frames	Not Recognized Frames	Incorrectly Recognized Frames	Average Cycle Time (for 50 logos)	# Processed Frames
1	107	35	56	14.293 ms	325 out of 8130
2	93	2	67	15.138 ms	271 out of 6779
3	90	11	31	15.200 ms	211 out of 5281
4	85	14	83	14.937 ms	312 out of 7808
5	124	0	82	15.057 ms	124 out of 3123
6	164	14	78	14.209 ms	398 out of 9972
7	125	5	54	15.260 ms	503 out of 12575

Table 6.4: Experiment results of SIFT on GPU.

Dataset	Logo Including Frames	Not Recognized Frames	Incorrectly Recognized Frames	Average Cycle Time (for 50 logos)	# Processed Frames
1	107	35	56	2.019 ms	325 out of 8130
2	93	2	67	2.140 ms	271 out of 6779
3	90	11	31	2.302 ms	211 out of 5281
4	85	14	83	2.294 ms	312 out of 7808
5	124	0	82	2.380 ms	124 out of 3123
6	164	14	78	2.307 ms	398 out of 9972
7	125	5	54	2.346 ms	503 out of 12575

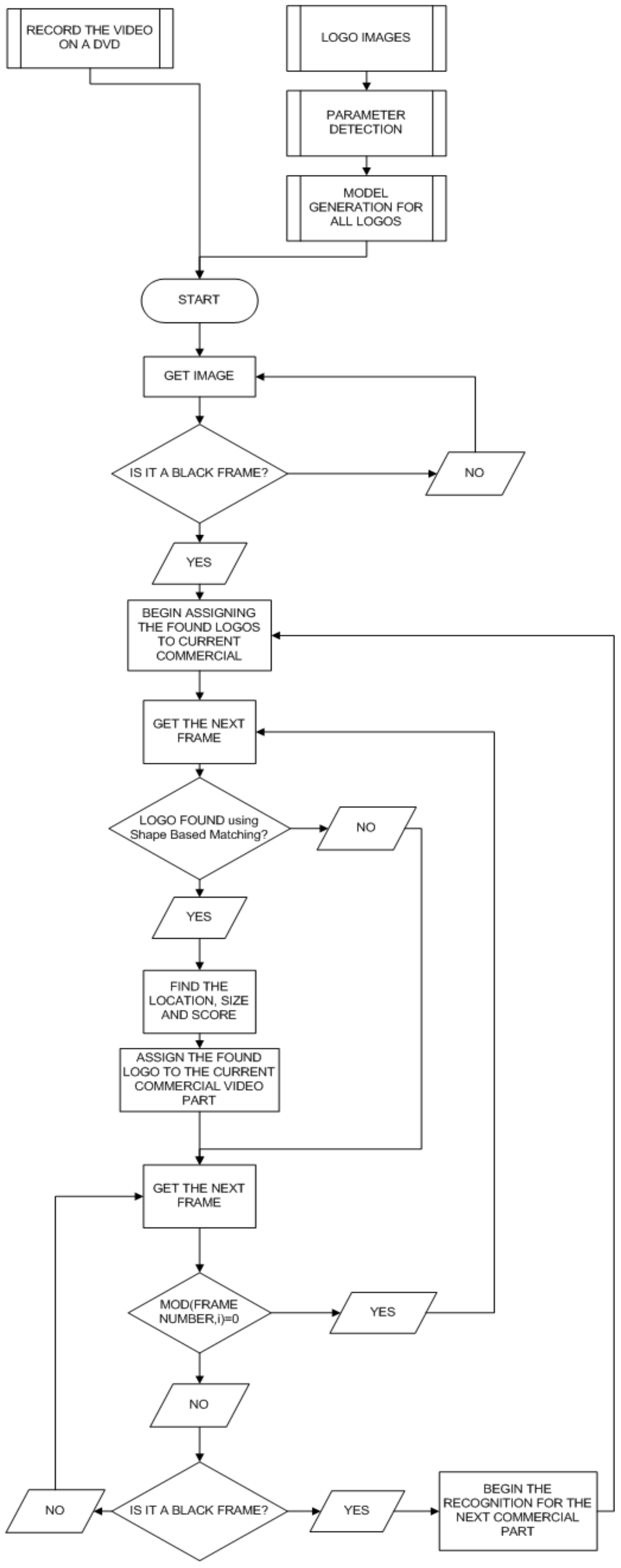

RECORD THE VIDEO ON A DVD
LOGO IMAGES
PARAMETER DETECTION
MODEL GENERATION FOR ALL LOGOS
START
GET IMAGE
IS IT A BLACK FRAME?
NO
YES
BEGIN ASSIGNING THE FOUND LOGOS TO CURRENT COMMERCIAL
GET THE NEXT FRAME
LOGO FOUND using Shape Based Matching?
NO
YES
FIND THE LOCATION, SIZE AND SCORE
ASSIGN THE FOUND LOGO TO THE CURRENT COMMERCIAL VIDEO PART
GET THE NEXT FRAME
MOD(FRAME NUMBER,i)=0
YES
NO
NO
IS IT A BLACK FRAME?
YES
BEGIN THE RECOGNITION FOR THE NEXT COMMERCIAL PART

Figure 6.1: Block diagram of the shape based logo recognition.

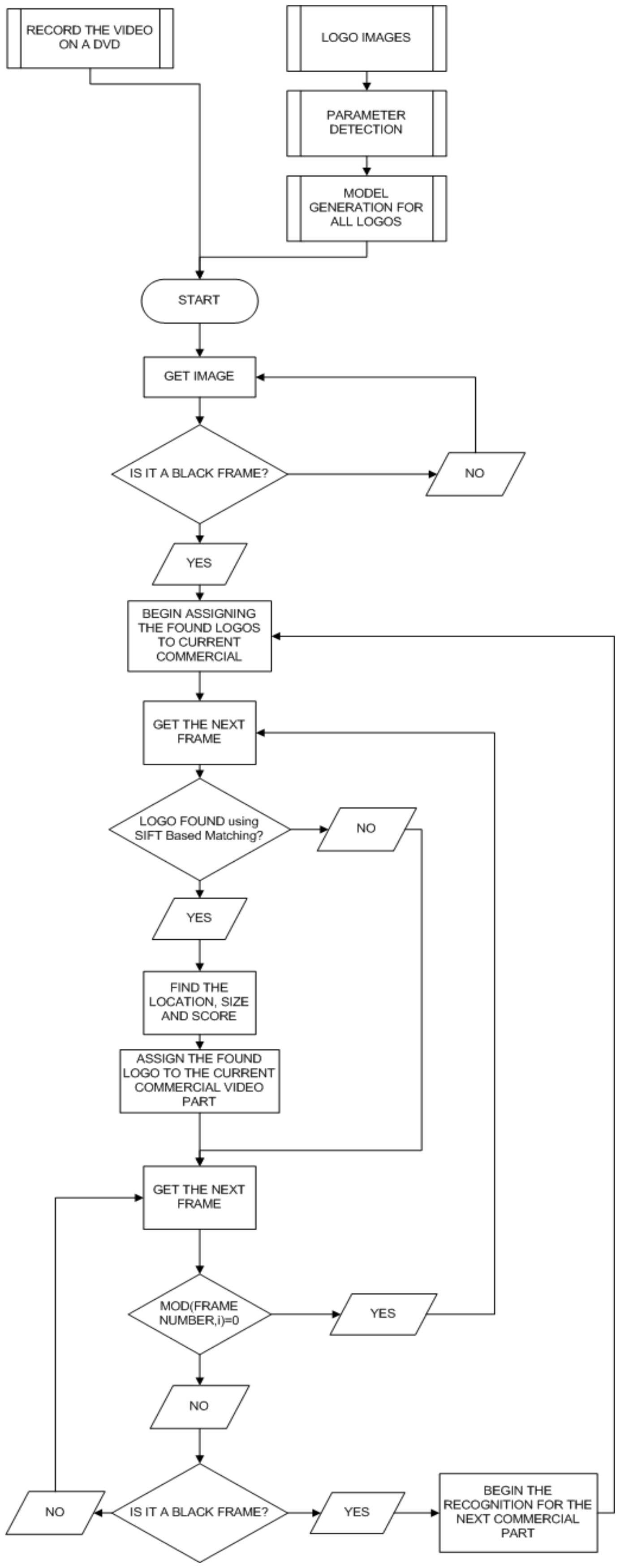
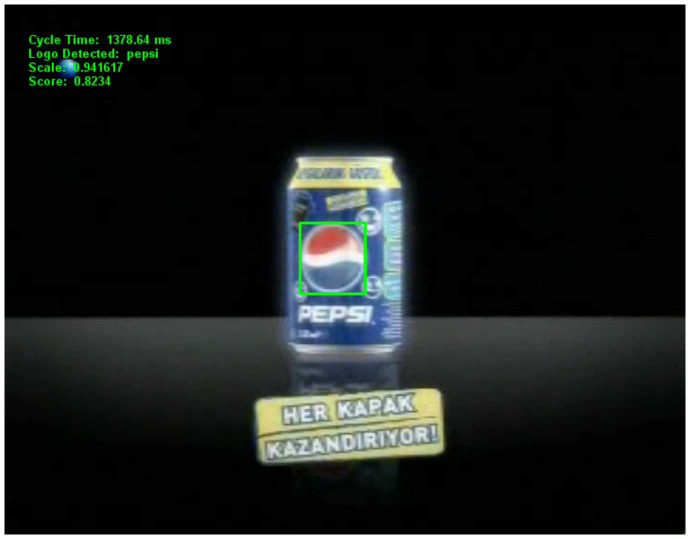
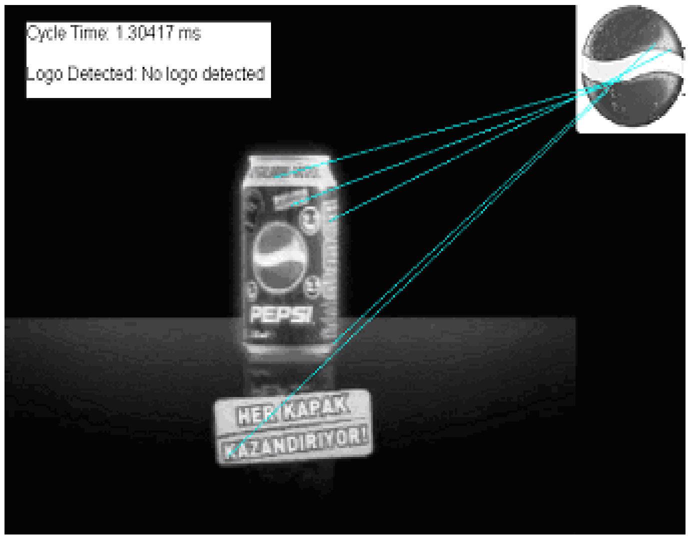

RECORD THE VIDEO ON A DVD
LOGO IMAGES
PARAMETER DETECTION
MODEL GENERATION FOR ALL LOGOS
START
GET IMAGE
IS IT A BLACK FRAME?
NO
YES
BEGIN ASSIGNING THE FOUND LOGOS TO CURRENT COMMERCIAL
GET THE NEXT FRAME
LOGO FOUND using SIFT Based Matching?
NO
YES
FIND THE LOCATION, SIZE AND SCORE
ASSIGN THE FOUND LOGO TO THE CURRENT COMMERCIAL VIDEO PART
GET THE NEXT FRAME
MOD(FRAME NUMBER,i)=0
YES
NO
IS IT A BLACK FRAME?
NO
YES
BEGIN THE RECOGNITION FOR THE NEXT COMMERCIAL PART

Figure 6.2: Block diagram of the SIFT based logo recognition.


```
Cycle Time:  1378.64 ms
Logo Detected:  pepsi
Scale:  0.941617
Score:  0.8234
```

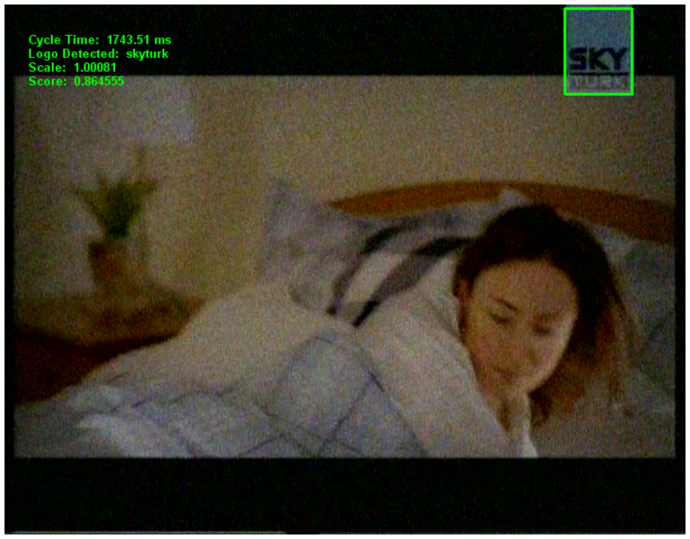
(a)


```
Cycle Time: 1.30417 ms

Logo Detected: No logo detected
```

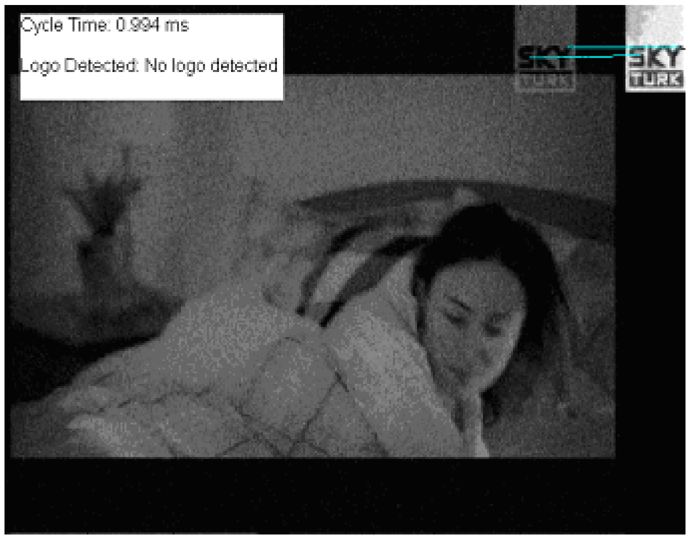
(b)

Figure 6.3: Motion blur a) shape based matching; b) SIFT based matching.



Cycle Time: 1743.51 ms
Logo Detected: skyturk
Scale: 1.00081
Score: 0.864555

(a)

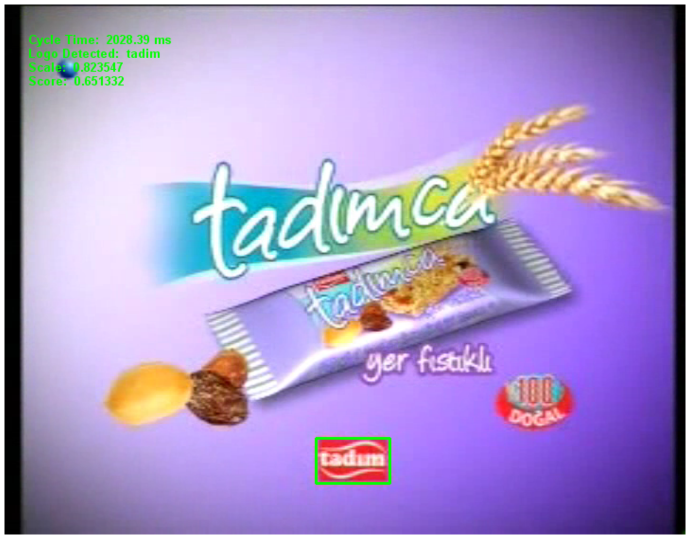


Cycle Time: 0.994 ms

Logo Detected: No logo detected

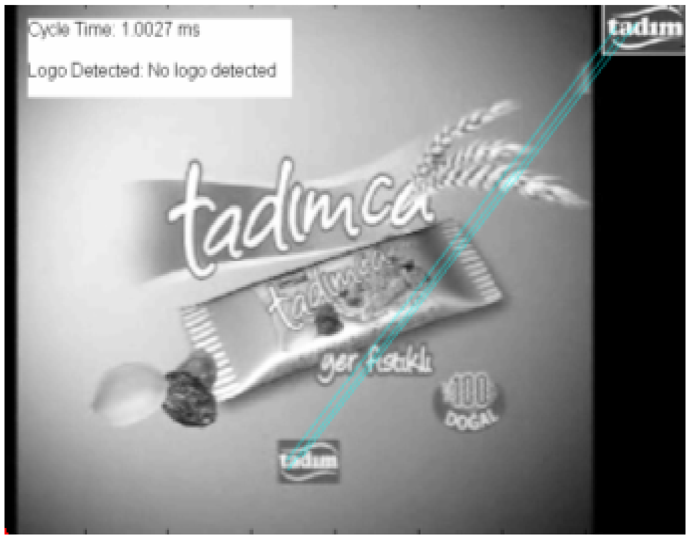
(b)

Figure 6.4: Transparent logo recognition a) shape based matching; b) SIFT based matching.



```
Cycle Time:  2028.39 ms
Logo Detected:  tadim
Scale:  0.823547
Score:  0.651332
```

(a)

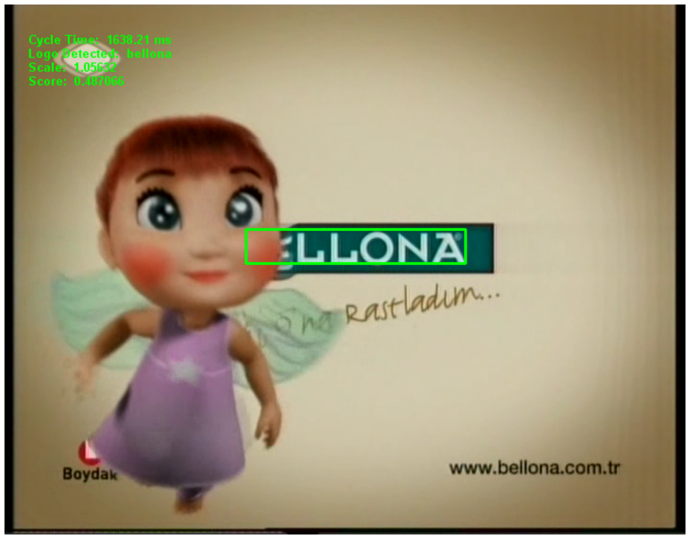


```
Cycle Time: 1.0027 ms

Logo Detected: No logo detected
```

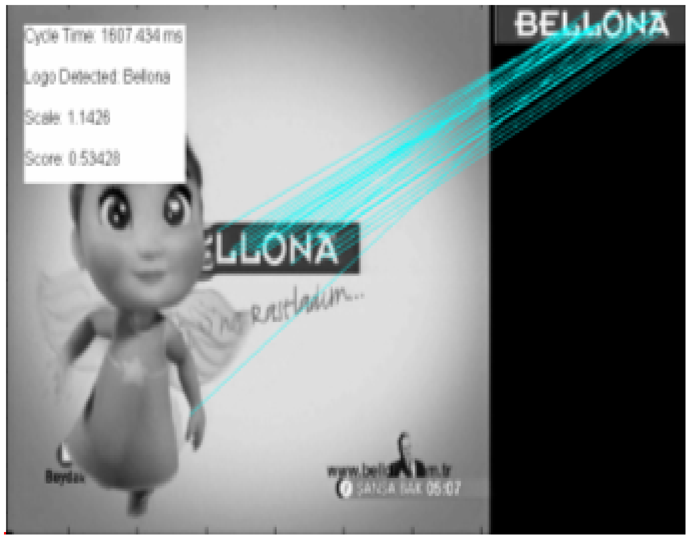
(b)

Figure 6.5: Small and low resolution logo image a) shape based matching; b) SIFT based matching.



```
Cycle Time:  1638.21 ms
Logo Detected:  bellona
Scale:  1.05632
Score:  0.487066
```

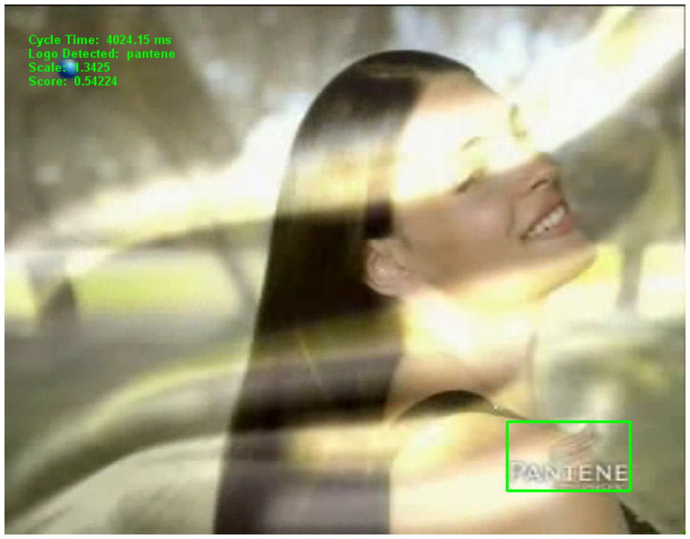
(a)



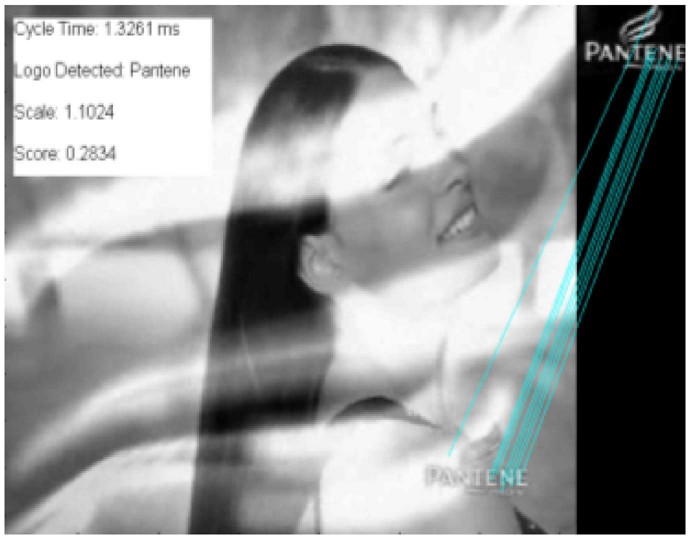
```
Cycle Time: 1607.434 ms
Logo Detected: Bellona
Scale: 1.1426
Score: 0.53428
```

(b)

Figure 6.6: Occlusion a) shape based matching; b) SIFT based matching.

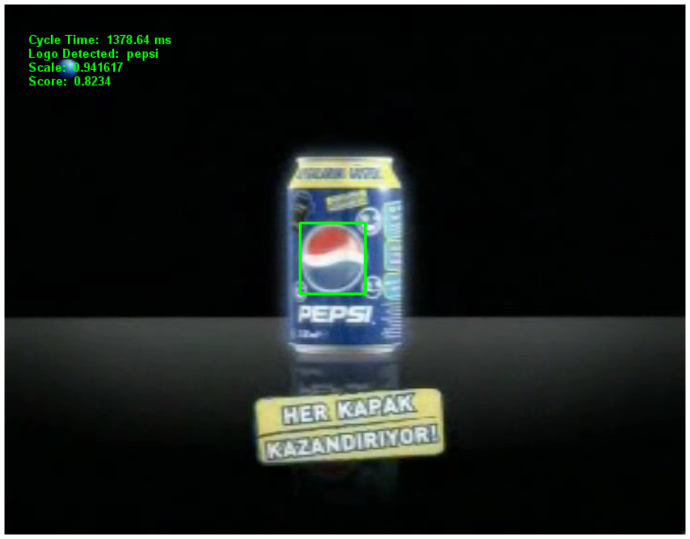


(a)



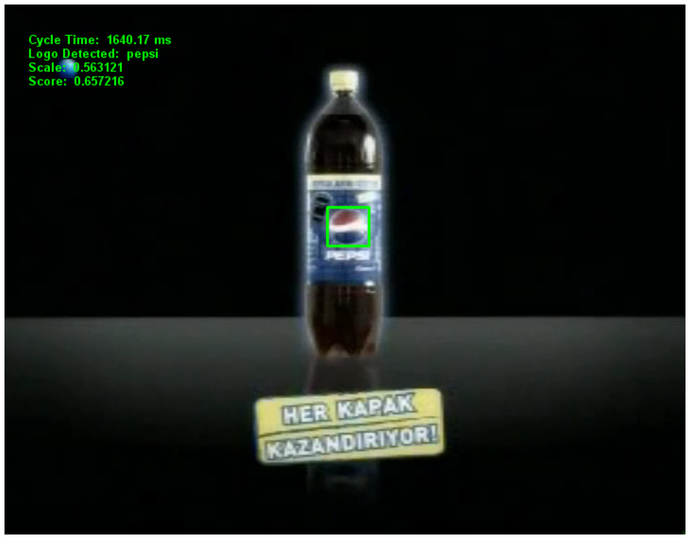
(b)

Figure 6.7: Illumination a) shape based matching; b) SIFT based matching.



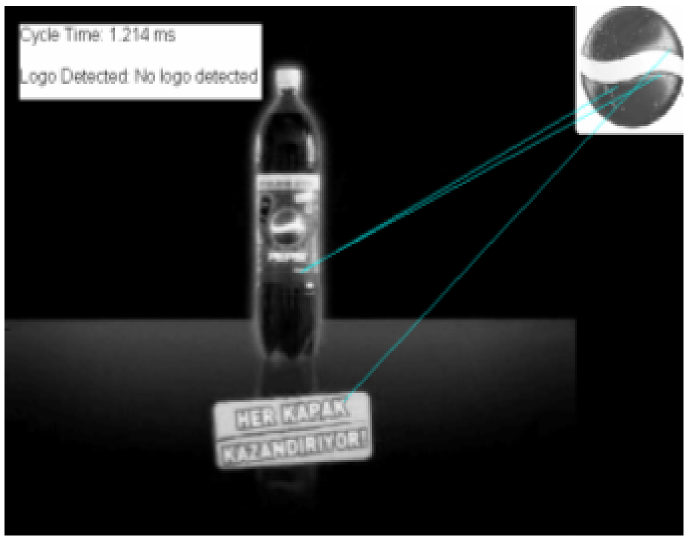
```
Cycle Time:  1378.64 ms
Logo Detected:  pepsi
Scale:  0.941617
Score:  0.8234
```

(a)



```
Cycle Time:  1640.17 ms
Logo Detected:  pepsi
Scale:  0.563121
Score:  0.657216
```

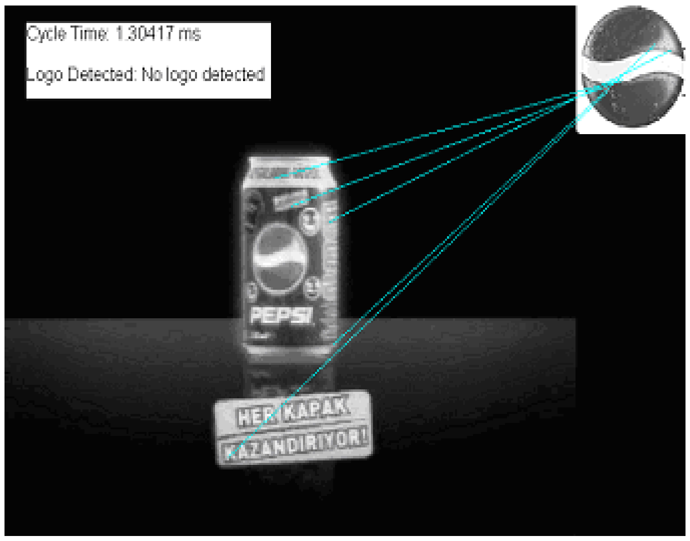
(b)



Cycle Time: 1.214 ms

Logo Detected: No logo detected

(c)

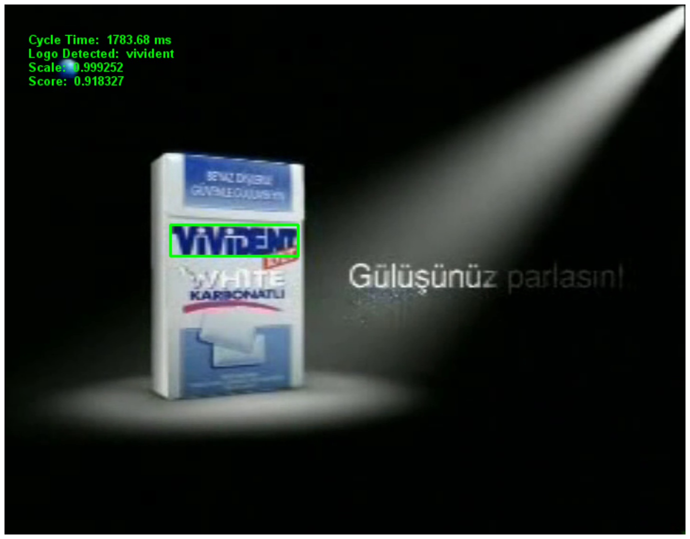


Cycle Time: 1.30417 ms

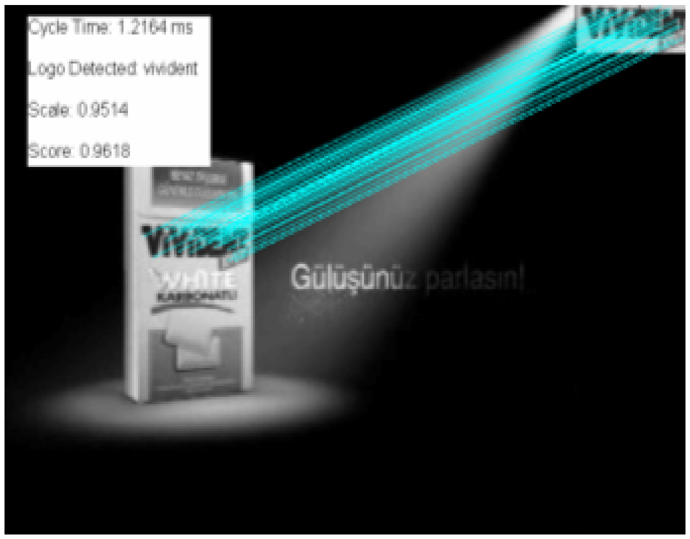
Logo Detected: No logo detected

(d)

Figure 6.8: Scale; a and b) shape based matching; c and d) SIFT based matching.


Cycle Time: 1783.68 ms
Logo Detected: vivident
Scale: 0.999252
Score: 0.918327

(a)


Cycle Time: 1.2164 ms
Logo Detected: vivident
Scale: 0.9514
Score: 0.9618

(b)

Figure 6.9: Perspective transformation a) shape based matching; b) SIFT based matching.

Chapter 7

Summary and Conclusion

In this study we have explored the applications of object recognition algorithms for trademark logo application in video frames. For this purpose, an automated logo recognition system is proposed, designed and tested. The system is implemented using the shape-based matching and SIFT based matching, as explained in detail in Chapters 3 and 4, respectively. As stated above, shape based matching uses global shape features (edges and edge points) for keypoint generation and SIFT uses local features (DOGs) for keypoint generation. This work also compares the performance of the global and local feature based algorithms for the purpose of logo recognition. The recognition and timing performance is analyzed in the previous section.

In the first part of the thesis we have implemented a commercial block detection algorithm in videos by using the statistical information of the gray values in frames. In this part of the thesis we have researched about the characteristics of the commercial blocks in TV broadcast and have tried to implement the distinguishing parameters of the commercial broadcast blocks. Then we have implemented our algorithm and tested our commercial detection algorithm. As a result we observed that the implemented algorithm works very well on the TV broadcast in Turkey. As a result we concluded that using the statistical information within the frames were sufficient to detect the black frames.

With the success in commercial video block detection, we have focused on the trademark logo detection in video frames. In the second part of the thesis we have implemented a shape based trademark logo recognition algorithm in order to catch the logos in video frames and assign them to the commercial block on broadcast.

During our tests, we have observed that the shape - based trademark logo detection algorithm was robust to occlusion, scale, and illumination as shown in the results in Table 6.1. Also the logos in TV broadcast video frames are affected by motion blur by their characteristics since the trademark logo is generally in motion in video to get people's attraction. Because of the communication issues, the frames generally have low resolution. As shown above the shape based trademark logo recognition algorithm was successful in classifying the trademark logos in low resolution frames.

In the next chapter, our dissatisfaction with the timing results of the shape based object recognition led us to implementation of the well known SIFT algorithm on a GPU card. The timing results of the SIFT were better than were those of the shape based recognition algorithm but the recognition performance was worse than recognition results for the shaped based one. Experiments demonstrate that the shape based trademark logo recognition algorithm was better than the SIFT in many applications ranging from the blurred, low resolution and transparent images to the ones that have very large illumination variances. However the SIFT algorithm on the GPU card performs faster than the shape based method.

From this thesis work we can also conclude that the shape based methods perform better than the SIFT based methods in rapidly changing environments since we have observed that SIFT was not very successful in environments that has large variance of illumination changes. Also since the local features all depend on the photometric features they are not robust to changes in illumination. However they perform better than the shape based method in occlusion and perspective transformation in recognition. Also for the logo recognition issues the transparent logos are one of the most problematic cases to recognize. It is also observed in this thesis that the shape based recognition algorithm showed very good results on the transparent trademark logos compared to the SIFT algorithm.

To sum up we can conclude that for trademark logo recognition shape based methods perform better in recognition and reasonable performance in computation. In addition we have observed that SIFT based methods perform better in computation performance.

7.1 Future Works

Our technique can be improved on several fronts. First of all, in our technique, we record the TV broadcast on a DVD and the work from the offline video. For future studies a system that stores the TV broadcast on a buffer and does the detection and recognition without recording the video on a DVD can be implemented.

In addition the parallel processing can be used which means the simultaneous use of more than one CPU to execute a program. Ideally, parallel processing makes a program run faster because there are more engines (CPUs) running it. Also with single-CPU computers as we have used it is possible to perform parallel processing by connecting the computers in a network in order to speed up the computations. However, this type of parallel processing requires very sophisticated software called distributed processing software.

In addition the system can be combined with a speech recognition system that recognizes the speech signal throughout the commercial video blocks.

During our experiments we have observed that the shape based recognition algorithm is better suited for low quality images and changes in illumination and SIFT is robust to coarse translations and rotations with low variance illumination changes. A hybrid approach, combining the outputs of shape based algorithm and SIFT based algorithm might result in a better recognition system.

Bibliography

[1] R. Lienhart, C. Kuhmnch, and W. Effelsberg, “On the detection and recognition of television commercials,” in *in Proc. IEEE Conf. on Multimedia Computing and Systems*, 1997, pp. 509–516.

[2] [Online]. Available: http://www.mvtec.com/halcon/

[3] D. Lowe, “Distinctive image features from scale-invariant keypoints,” in *International Journal of Computer Vision*, vol. 20, 2003, pp. 91–110.

[4] B. Satterwhite and O. Marques, “Automatic detection of tv commercials,” *Potentials, IEEE*, vol. 23, no. 2, pp. 9–12, April-May 2004.

[5] C. Steger, “Occlusion, clutter, and illumination invariant object recognition,” in *International Archives of Photogrammetry and Remote Sensing*, vol. XXXIV, part 3A, 2002, pp. 345–350.

[6] Y. Lamdan, J. Schwartz, and H. Wolfson, “Affine invariant model-based object recognition,” vol. 6, pp. 578–589, 1990.

[7] S. Belongie, J. Malik, and J. Puzicha, “Shape matching and object recognition using shape contexts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[8] C. Schmid and R. Mohr, “Local grayvalue invariants for image retrieval,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–535, 1997.

[9] T. Gevers and A. Smeulders, “Color-based object recognition,” vol. 32, no. 3, pp. 453–464, March 1999.

[10] D. Lowe, “Distinctive image features from scale-invariant keypoints,” vol. 60, no. 2, pp. 91–110, November 2004.

[11] Y.-S. Kim and W.-Y. Kim, “Content-based trademark retrieval system using visually salient features,” in *CVPR*. IEEE Computer Society, 1997, pp. 307–312.

[12] P. Y. Yin and C. C. Yeh, “Content-based retrieval from trademark databases,” *Pattern Recognition Letters*, vol. 23, no. 1-3, pp. 113–126, Jan. 2002.

[13] J. P. Eakins, K. J. Riley, and J. D. Edwards, “Shape feature matching for trademark image retrieval,” in *CIVR*, 2003, pp. 28–38.

[14] R. H. van Leuken, M. F. Demirci, V. J. Hodge, J. Austin, and R. C. Veltkamp, “Layout indexing of trademark images,” in *CIVR ’07: Proceedings of the 6th ACM international conference on Image and video retrieval.* New York, NY, USA: ACM, 2007, pp. 525–532.

[15] F. Aldershoff and T. Gevers, “Visual tracking and localisation of billboards in streamed soccer matches,” 2004.

[16] B. Kovar and A. Hanjalic, “Logo appearance statistics in a sport video: Video indexing for sponsorship revenue control,” 2002.

[17] R. J. M. den Hollander and A. Hanjalic, “Logo recognition in video stills by string matching,” in *ICIP (3)*, 2003, pp. 517–520.

[18] R. den Hollander and A. Hanjalic, “Logo recognition in video stills by string matching,” *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, pp. III–517–20 vol.2, Sept. 2003.

[19] F. Pelisson, D. Hall, O. Riff, and J. Crowley, “Brand identification using gaussian derivative histograms,” 2003, p. 492 ff.

[20] G. Kienast, H. Stiegler, W. Bailer, H. Rehatschek, S. Busemann, and T. Declerck, “Sponsorship tracking using distributed multi-modal analysis (direct-info),” 2005, pp. 341–348.

[21] R. Hurst, "Tv commercial editor," p. 31, 1994.

[22] D. A. Sadlier, S. Marlow, N. E. O'Connor, and N. Murphy, "Automatic TV advertisement detection from MPEG bitstream," in *PRIS*, A. L. N. Fred and A. K. Jain, Eds. ICEIS Press, 2001, pp. 14–25.

[23] X. Wen, T. D. Huffmire, H. H. Hu, and A. Finkelstein, "Wavelet-based video indexing and querying," *Multimedia Syst.*, vol. 7, no. 5, pp. 350–358, 1999.

[24] A. Hampapur and R. Bolle, "Feature based indexing for media tracking," *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 3, pp. 1709–1712 vol.3, 2000.

[25] C. M. A. N. H. C. Iggulden, Jerry (Santa Clarita, "Method and apparatus for controlling a videotape player to automatically scan past recorded commercial messages," July 1994. [Online]. Available: http://www.freepatentsonline.com/5333091.html

[26] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.*, vol. 24, no. 4, pp. 325–376, 1992.

[27] S. Tanimoto, "Template matching in pyramids," vol. 16, no. 4, pp. 356–369, August 1981.

[28] S. H. Lai and M. Fang, "Robust and efficient image alignment with spatially-varying illumination models," in *IEEE Computer Vision and Pattern Recognition or CVPR*, 1999, pp. II: 167–172.

[29] G. Brogefors, "Hierarchical chamfer matching: A parametric edge matching algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 849–865, 1988.

[30] W. J. Rucklidge, "Efficiently locating objects using the hausdorff distance," *Int. J. Comput. Vision*, vol. 24, no. 3, pp. 251–270, 1997.

[31] C. F. Olson and D. P. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Transactions on Image Processing*, vol. 6, pp. 103–113, 1997.

[32] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," in *RCV87*, 1987, pp. 714–725.

[33] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[34] C. Steger, "An unbiased detector of curvilinear structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 2, pp. 113–125, 1998.

[35] W. Foerstner, "A framework for low level feature extraction," *Lecture Notes in Computer Science*, vol. 800, pp. 383–396, 1994.

[36] C. Steger, "Subpixel-precise extraction of lines and edges," in *International Archives of Photogrammetry and Remote Sensing*, vol. XXXIII, part B3, 2000, pp. 141–156.

[37] A. P. Witkin, "Scale-space filtering," in *International Joint Conference on Artificial Intelligence*, 1983, pp. 1019–1022.

[38] O. Pele, "Sift presentation slights," 2006.

[39] C. Wu, "Sift on gpu," Department of Computer Science, University of North Carolina - Chapel Hill, Tech. Rep., 2007.

[40] M. Brown and D. G. Lowe, "Recognising panoramas," in *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision.* Washington, DC, USA: IEEE Computer Society, 2003, p. 1218.

[41] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Gpu point list generation through histogram pyramids," Max-Planck-Insitut fr Informatik, Tech. Rep. MPI-I-2006-4-002, 6 2006.

[42] [Online]. Available: http://www.gpgpu.org/w/index.php/Glossary#Ping-Pong

[43] G. Ziegler, A. Tevs, C. Theobalt, and H.-P. Seidel, “GPU-based video feature tracking and matching,” Department of Computer Science, University of North Carolina - Chapel Hill, Tech. Rep. TR06-012, May 5 2006, mon, 14 Aug 2006 14:36:13 UTC.

[44] A. Vedaldi, “An open implementation of the SIFT detector and descriptor,” UCLA CSD, Tech. Rep. 070012, 2007.