Video Analysis and Abstraction in the Compressed Domain

A Thesis
Presented to
The Academic Faculty
by
Sangkeun Lee

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
October 2003

Copyright © 2003 by Sangkeun Lee
Video Analysis and Abstraction in the Compressed Domain

Approved by:

Dr. Mason H. Hayes, Advisor

Dr. Russel M. Mersereau

Dr. Yucel Altunbasak

Date Approved 10/25/2003
To my wife, HyunKyung; my parents, Jongdae and Jeongsik; my brothers, Sangjin and Sangsook; and my sister, Sangsook.
ACKNOWLEDGEMENTS

As I reflect on my graduate school years, there are many to whom I am deeply grateful. I am thankful to God for giving me the ability to pursue and complete this thesis. I could not have considered graduate work at all without the foundation and support of my parents, Jongdae in the heaven and Jeongsik Yeo. I am indebted to my advisor, Professor, Momou H. Hayes, who provided direction and encouragement throughout my research. I also wish to thank Professors Russell Mersereau and Yuvel Altunbasak for joining Dr. Hayes on my reading committee, and Professors John Barry and Jeffrey Gerstman for serving as additional committee members.

I am also grateful to the CSIP secretaries - Kay Gilstrap, Christy Ellis, and Charlotte Doughty - for all they have done to help me. Thanks also go to Keith May and Sam Smith, whose assistance with the computer network was invaluable.

During my time at Georgia Tech, I have developed many friendships that have made my graduate experience memorable, both academically and socially. I have benefited greatly from discussions with Paul Hong, Greg Slabaugh, John Glotzbach, Samuel Li, Matthew Lee, Yeeman Yoon, Hyungjoon Kim, Heejong Yoo, and Jungwon Lee. I must also thank some "non-CSIP" friends through the years - Sungwan Min and Wangmyung Woo - for helping me find things to do when I wasn't at school. Additionally, I am grateful for my friendships with Daniel Kim, Kunitoshi Arai, Kentaro Hara, and Fisher Londono. All of these friends were my Kando (Korean martial art; Kendo in Japanese, the way of sword in English) teammates in GKA (Georgia Kendo Alliance), with whom I shared numerous victories that I'll always remember.

Finally, I thank my wife, Hyangkyung, for her love and patience during my pursuit
of this degree. Her support and friendship helped me tremendously during the final months of my research.
TABLE OF CONTENTS

DEDICATION .................................................. iii

ACKNOWLEDGEMENTS ........................................ iv

LIST OF TABLES ........................................... x

LIST OF FIGURES .......................................... xi

SUMMARY .................................................. xiv

1 INTRODUCTION ........................................... 1
  1.1 Motivation .......................................... 1
  1.2 Scope of Thesis ..................................... 2
  1.3 Organization of Thesis .............................. 2

2 BACKGROUND ............................................. 3
  2.1 Terminology and Video Processing ................. 3
  2.2 Shot Change Detection Techniques for Full-motion Video .... 5
    2.2.1 Uncompressed Domain techniques ................ 5
    2.2.2 Compressed Domain Techniques ............... 7
  2.3 Shot Classification ................................ 9
  2.4 Video Abstraction ................................ 10
    2.4.1 Video Summarization ......................... 10
    2.4.2 Video Skimming ............................... 13

3 METHODOLOGY ........................................... 14
  3.1 The Abstraction System ............................ 14
  3.2 Compressing Images ................................ 16
    3.2.1 Terminology .................................. 16
    3.2.2 Intra-frame Compression ..................... 17
    3.2.3 Inter-frame Compression .................... 20
  3.3 Reconstructing DC-images ......................... 21
APPENDIX A — THE HIERARCHICAL MPEG STRUCTURE 119
APPENDIX B — MATHEMATICAL MORPHOLOGY 125
REFERENCES 133
VITA 139
LIST OF TABLES

1. Matrices $S_1$ and $S_2$. .................................................. 26
2. The distribution of MB types at shot change points. ................. 29
3. Parameters and values to identifying shot changes around sample candidates. ............................................. 43
4. Performance of the camera movement classification. ............... 45
5. Experimental comparison for shot change detection algorithms. ... 49
6. Results of the clustering speed comparison in seconds. ............. 64
7. Results for clustering accuracy. ........................................... 65
8. Computation complexities according to masks. ......................... 59
9. Characteristics of the faces in the test data set. ..................... 90
10. Face detection results. .................................................... 93
11. Overall speed comparison between the proposed system and Wang’s system. .................................................. 95
12. Characteristics of the refined key frames. .............................. 103
13. Overall speed comparison between the proposed system and Huang’s system. .............................................. 110
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Overall scheme for video abstraction.</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Frame types in MPEG.</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>A single macroblock with the blocks.</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Baseline intra-frame (a) encoder and (b) decoder.</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>The $Y$, $Cb$, and $Cr$ images for a macroblock in the MPEG 4:2:0 format.</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>Differential DC computation.</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Zigzag sequence of entropy coding DC and AC coefficients.</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>Partial decoding of the compressed data.</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>The basis functions for all of the two-dimensional DCT coefficients.</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>An example of reconstructing a DC-image by choosing DC coefficients in an intra-frame.</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>Reference block ($P_{ref}$), motion vector, and original block.</td>
<td>25</td>
</tr>
<tr>
<td>12</td>
<td>An original image and its DC-image.</td>
<td>26</td>
</tr>
<tr>
<td>13</td>
<td>Abrupt shot change points between two anchor frames.</td>
<td>28</td>
</tr>
<tr>
<td>14</td>
<td>Probability density functions of the first B-frame bi-MBs.</td>
<td>31</td>
</tr>
<tr>
<td>15</td>
<td>Proposed algorithm for detecting gradual shot changes.</td>
<td>33</td>
</tr>
<tr>
<td>16</td>
<td>The overall camera motion classification scheme.</td>
<td>36</td>
</tr>
<tr>
<td>17</td>
<td>An example of an MVF and sub-MVFAs.</td>
<td>37</td>
</tr>
<tr>
<td>18</td>
<td>Basic camera motion templates.</td>
<td>39</td>
</tr>
<tr>
<td>19</td>
<td>Temporal bi-MBs and candidates for abrupt shot change.</td>
<td>42</td>
</tr>
<tr>
<td>20</td>
<td>Gradual scene change detection: (a) temporal distribution of bi-MBs in sub-MB images and (b) its result.</td>
<td>44</td>
</tr>
<tr>
<td>21</td>
<td>The results of the subshot segmentation for a part of a movie sequence.</td>
<td>45</td>
</tr>
<tr>
<td>22</td>
<td>Key frames of a movie.</td>
<td>46</td>
</tr>
<tr>
<td>23</td>
<td>Key frames of a commercial film.</td>
<td>47</td>
</tr>
<tr>
<td>24</td>
<td>Procedures for making feature matrix $A$.</td>
<td>60</td>
</tr>
<tr>
<td>25</td>
<td>Refined key frames for a movie.</td>
<td>63</td>
</tr>
</tbody>
</table>
Graph of speed comparison results. ........................................ 65
Example of hierarchical video abstraction and browsing. ............... 67
(a) Sample skin color distribution in YCbCr color space and (b) its color histogram in the CbCr plane. ........................................ 73
(a) Gaussian mixture model fitted by skin samples in the CbCr plane and (b) its probability density function (pdf). ........................................ 74
Classification comparison according to different thresholds. .................. 76
An example of a binary map for skin regions; (a) original DC-image, (b) segmented binary map for skin, and (c) labeled skin regions. .................. 76
Orientation and axes of a region. ........................................ 79
(a) A potential face region and (b) the region after removal of the lower portion to achieve a height to width ratio of 1.6. .................. 80
the sub-sampled face template. ........................................ 81
Template matching procedure ........................................ 82
Zigzag ordering of DCT coefficients. ........................................ 84
The partitioning of the DCT coefficients. ........................................ 85
Faces according to their orientations. ........................................ 86
Separated regions in the DCT domain. ........................................ 87
Two pairs of partitioning AC coefficients. ........................................ 88
The AC coefficient relationships between symmetry edges. .................. 88
The ratio of moments with different mask sets. ........................................ 89
Some results of the proposed face detection algorithm. .................. 92
A frame and its EC-image. ........................................ 92
Some comparative results from the test set. The first column corresponds to the output of the CMU face detector; the second column corresponds to the output of Wang's system; and the third column corresponds to the output of the proposed algorithm. .................. 98
Procedures for multi-level video abstraction and browsing. .................. 100
Overall structure of the abstraction system. ........................................ 101
An example of refined key frames for a commercial film. .................. 102
Video summary example. ........................................ 104
A screen-capture illustrating the **Listview** interface after sorting of the priority column in descending order.

A screen-shot illustrating the developed video tools in this work.

**MPEG** hierarchical structure.

“A is dilated by the structuring element B.”

“A is eroded by the structuring element B to give the internal dashed shape.”

The opening (given by the dark dashed lines) of A (given by the solid lines). The structuring element B is a disc. The internal dashed structure is A eroded by B.

The closing of A by the structuring element B.

Results of morphological operations on a binary image by a disk element structure of size 2: (a) original image, (b) erode operation, (c) dilate operation, (d) opening operation, and (e) closing operation.

The skeletons of three simple regions.
SUMMARY

The objective of this work is to develop a fast system for video abstraction. The proposed system can summarize a video faster than real-time and can create images and short video clips according to a multi-resolution scheme for interactive video browsing. The proposed approach is camera event-based and is designed around a simple, fast, and reliable algorithm. Specifically, to reduce the high computational load, the measurements used to make decisions for key-frame extraction come directly from the compressed domain. First, shot segmentation and camera movement classification are performed. Next, a simple key-frame extraction algorithm selects the representative frames that are clustered according to their visual similarities using a new and fast clustering algorithm that combines the SVD and k-means algorithms. Finally, the clustered frames are assembled into a video summary using a hierarchical scheme. In addition, a new method for detecting human faces is proposed. Comparing the proposed approach to video abstraction with another algorithm, it is demonstrated that the proposed approach is fast and produces an effective video summary. An average overall processing time that is 9 times faster than real-time on a Pentium II, 400 MHz PC has been achieved for 352 × 240 video sequences.
CHAPTER 1

INTRODUCTION

1.1 Motivation

Rapid advances in communication and multimedia computing technologies have enabled home users to access vast amounts of multimedia data. The demand for various services such as video on demand (VOD) and digital library is rapidly increasing. As the amount and complexity of video information grows, the need for more intelligent video manipulating techniques becomes evident. Although video sequences convey real-world scenes most vividly, it is always a painful task to find either the appropriate video sequence or the desired portion of the video from a large video data collection. The situation becomes even worse on the Internet. Recently, video abstraction has been attracting considerable research interest [26, 36], and it is gradually playing a more important role in the multimedia database area. A concise and intelligently generated video abstraction will not only enable a more informative human/computer interaction during video browsing, but it will also help to build more meaningful and quicker video indexing and retrieval systems.

Because of the large amount of data, video sequences are often compressed into the motion picture expert group (MPEG) format for efficient transmission or for storage on-line. These compressed video sequences have to undergo computationally intensive processing steps to be decompressed prior to the application of any video-processing algorithm [11, 33]. Therefore, it is necessary for video sequences to be treated in the compressed domain for processing speed and real-time implementation.
1.2 Scope of Thesis

While progress has been made in identifying shot boundaries and the salience of different parts of the overall source video, relatively little work has focused on the composition or synthesis part of summarization, especially with respect to accuracy and speed. In order to accomplish this goal, several algorithms for quickly analyzing and summarizing a video are proposed. In addition, a simple and efficient method for detecting human faces is proposed. Each algorithm will be compared with well-known methods with various data sets.

1.3 Organization of Thesis

As stated in the title, the focus of this thesis is to implement a system for the analysis and abstraction of a video in the compressed domain. The developed algorithms for this purpose are presented in four chapters: one for video analysis and three for video abstraction. The abstraction chapters are split since their natures are very different. Chapter 4 is dedicated to algorithms for breaking a video sequence down into fundamental units and interpreting their contents. The representative frames of a sequence are clustered according to their visual similarities in Chapter 5. In Chapter 6, a simple and fast algorithm for detecting human faces is described. How to summarize a video sequence through interaction with a user is presented in Chapter 7.

In addition to the primary chapters, there are several support chapters. Chapter 2 includes background information on various topics relevant to this thesis. An overview of the proposed work and important aspects of working with compressed data are shown in Chapter 3. The last chapter draws conclusions from this work and gives suggestions for further lines of inquiry. A few additional pieces of information have also been placed in the appendices.
CHAPTER 2

BACKGROUND

In the following sections, the terminology is first addressed in three main areas of video processing: shot change detection, shot classification, and video abstraction. Next, the most successful methods are summarized. A detailed discussion of video abstraction is also presented.

2.1 Terminology and Video Processing

Before discussing the ways in which a video may be summarized, it is important to take a look at how video is structured. Video streams may be broken down into frames, shots, scenes, and segments [49]. A frame is a single image in the video stream. Although frames are often subdivided into subframes, for example macroblocks in an MPEG encoder, a frame may be thought of as the basic building block of a video sequence. A shot, on the other hand, consists of an unbroken sequence of frames taken by a single camera and is often referred to as the fundamental film component. A shot is sometimes broken down into subshots. A subshot consists of a sequence of frames within a shot in which the movements of objects within the shot are nearly identical. A scene is a high-level unit that consists of a sequence of shots that focus on the same point of interest [49]. Although some researchers use the terms “shot” and “scene” interchangeably, in this work, these terms are used according to the definitions given above. Finally, a segment is a sequence of scenes that forms a story unit.

There are three main areas in multimedia computing that can make the object of a video more accessible.

Shot Change Detection: Since frames within a shot tend to be visually similar,
techniques for video parsing typically rely on the detection of points in the video sequence where some quantitative measure of the difference between successive frames exceeds a certain threshold. These points, referred to as shot transitions, fall into two categories: abrupt transitions and gradual transitions. Abrupt transitions occur, for example, when there is a sudden shift from one camera to another, whereas gradual transitions occur in shot transitions that involve such effects as dissolves and fades.

Shot Classification: Once a shot has been identified, it is necessary to represent its content. This representation may involve the use of text, mathematical transformations, or images.

Video Abstraction: As the name implies, this is a short summary of the content of a longer video document with a small number of static images, key frames, or moving images. There are two general types of video abstractions: video summarization and video skimming [67]. Video summarization, sometimes referred to as “still-image abstraction,” involves the creation of a small collection of representative images that are extracted or generated from the underlying video sequence. Once the video summary has been created, it may be easily displayed or presented to the user since it consists simply of a storyboard or gallery of images. These images may be arranged in a number of ways. For example, they may be arranged temporally (so that the user is able to rapidly peruse or search the underlying video) or they may be arranged hierarchically in terms of importance or significance of the frames so that the user can get a quick snapshot of the essence of the video, or so that the user can search for shots or scenes using a tree-like search.

The second form of video abstraction is called video skimming, which is also referred to as “moving-image abstraction.” In video skimming, a collection of
image sequences are extracted from the video along with the corresponding audio tracks. Although generally more time-consuming than video summarization, video skimming has the advantage of using audio tracks, which may contain important information, as in education and training videos. Video skimming also has advantages during play back, since it is usually more natural and interesting for users to watch a trailer than to watch a slide show, and in many cases, the motion that is displayed is information-bearing.

2.2 Shot Change Detection Techniques for Full-motion Video

A number of methods have been suggested for video shot change detection in both the uncompressed (spatial) domain and the compressed domain since Seyler [54] developed a frame difference coding technique for television signals.

2.2.1 Uncompressed Domain techniques

There are pair-wise comparisons, which include pixel-, block-, and frame-level comparison, and histogram comparison techniques.

2.2.1.1 Pair-wise Comparisons

A simple way to detect a qualitative change between a pair of images is to compare the corresponding pixels in the two frames to determine how many pixels have changed [24]. The potential problem with this method is its sensitivity to camera and object movement. A shot change will be detected if these movements cause a large number of pixel changes.

The block-level comparison technique divides each frame into K-blocks and measures the disparity between corresponding blocks of the two images by computing the

---

1Compressed domain refers to DCT-based compression approaches such as JPEG and MPEG-1 and -2.
sum of point-by-point differences [46, 55]. This method is more robust in the presence of local or global motion between the two images and has the advantage of extracting motion information for each block.

Another approach involves comparing corresponding regions (blocks) in two successive frames based on second-order statistical characteristics of their intensity values. One such metric for comparing corresponding regions is called the likelihood ratio [34]. A potential problem of this method is that no change will be detected if two compared sample areas have the same mean and variance but completely different probability density functions. Fortunately, such a situation is unlikely.

### 2.2.1.2 Histogram Comparison

An alternative to comparing corresponding pixels or regions in successive frames is to compare some feature of the entire image. One such feature that can be used for partitioning purposes is a histogram of intensity levels. The principle behind this approach is that two frames having an unchanging background and unchanging objects will show little difference in their respective histograms. The histogram comparison algorithm should be less sensitive to object motion than the pair-wise pixel comparison algorithm since it ignores the spatial changes in a frame.

Let $H_i(j)$ denote the histogram value for the $i$th frame, where $j$ is one of the $G$ possible gray levels (the number of histogram bins can be chosen on the basis of the available gray-level resolution and the desired computation time). The difference between the $i$th frame and its successor will be given by the following formula:

$$SD_i = \sum_{j=1}^{G} |H_i(j) - H_{i+1}(j)|.$$  \hfill (2.1)

If the overall difference is larger than a given threshold $T_b$, a segment boundary is detected [46, 24]. Equation (2.1) can also be applied to histograms of individual color channels. A simple but effective approach is to use a color histogram comparison [24].
An alternate approach to linear histogram comparison is the following $\chi^2$-test equation:

$$SD_i = \sum_{j=1}^{G} \frac{(H_i(j) - H_{ref}(j))^2}{H_{ref}(j)}.$$  \hspace{1cm} (2.2)

This metric was proposed in [46] for its ability to enhance differences between the frames being compared. However, experimental results reported in [24] showed that, while this equation enhances the difference between two frames across a camera break, it also increases the difference resulting from camera or object movements. Therefore, the overall performance is not necessarily better than that achieved by using equation (2.1), while equation (2.2) requires more computation time.

### 2.2.2 Compressed Domain Techniques

Recently, the emphasis in multimedia processing has been on algorithms that are applied in the compressed domain using partially decoded data such as macroblock, motion vectors, or DC images [5, 36, 57, 14].

Two components are commonly used for shot change detection in the compressed domain: discrete cosine transform (DCT) coefficients and motion vectors.

#### 2.2.2.1 Techniques Based on the DCT Coefficients

Arman et al. [10] worked on image processing techniques directly applied to compressed data. Their technique consisted of correlating DCT coefficients of consecutive frames of JPEG compressed video. Zhang et al. [24] modified this method for detecting shot changes and applied the pair-wise comparison technique to the DCT coefficients of corresponding blocks of consecutive video frames in the I-frames of MPEG sequences.

An alternate approach is to use the DC coefficients of the DCT, which is directly proportional to the average value of the pixels within the DCT-applied block. Yeo and Liu [71, 72] used DC coefficient difference sequences. DC coefficient values for the I-, P-, and B-frames are extracted (DC coefficient values for the P- and B-frames are
motion compensated and reconstructed as in [6, 70]) and used to construct a DC frame sequence. Two metrics for the difference measure are then presented: the sum of the absolute DC frame pixel-to-pixel differences and the bin-to-bin differences between color histograms of the DC frame. Yeo and Lin also suggest the gradual shot change detection in their algorithm from investigating the gradually increasing multiframe difference by a plateau followed by a decreasing frame difference. However, luminance or color is sensitive to small changes, so these features produce false alarms.

Another interesting approach is to use the AC coefficients of the DCT. Lee et al. [51] proposed a fast shot change detection algorithm using direct feature extraction from MPEG compressed videos. First, binary edge maps are derived from the AC coefficients in blocks that are transformed using the DCT. Second, edge orientation, strength, and offset using the correlation are measured between the AC coefficients in the derived binary edge maps. Finally, they match two consecutive frames using these two features (edge orientation and strength). This method is fast and reliable, but it concentrates only on abrupt shot change detection.

2.2.2.2 Techniques Based on the Motion Vectors

Apart from intensity values and distributions, motion of both objects and the camera are major elements of video content. In general, within a single camera shot, the field of motion vectors should show relatively continuous changes, while this continuity will be disrupted between frames across different shots. Thus, a continuity metric for a sequence of motion vector fields should be able to serve as an alternate criterion for detecting shot boundaries. In an MPEG data stream, the residual error after motion compensation in both B- and P-frames is transformed into DCT coefficients and coeff. However, if this residual error exceeds a given threshold for certain blocks, motion compensation prediction is abandoned, and those blocks are represented by DCT coefficients, as in the I-frame.
Some researchers used statistics on the numbers and types of prediction vectors used to encode the P- and B-frames [33, 38]. To detect a shot boundary that occurs on an I-frame, Meng et al. [33] performed the color histogram comparison with DC coefficients of two consecutive I-frames, and an adaptive threshold method was used for peak detection. While Liu and Zick's method [38] detected only abrupt shot changes, the method of Meng et al. [33] was designed to detect dissolve effects.

2.3 Shot Classification

Once a shot is identified from the shot change detection procedure, it needs to be characterized for interpreting its contents. Identifying content is fundamentally a problem of perception; unfortunately, perception is ultimately an act of interpretation on the part of the perceiver, an act that is inescapably subjective [29]. A conventional method for analyzing a shot is to detect camera operations, which concentrate on a very low (and extremely objective) level of content analysis. Camera operation gives cues for inferring high-level semantic meanings such as the intention of video producers. Furthermore, a video sequence is composed of consecutive camera operations, and a shot is a sequence of frames with varying camera operations. Therefore, a shot can be segmented into smaller units of subshots that keep a homogeneous camera motion for more detailed manipulation.

Most of the existing methods are based on analyzing the optical flow computed between consecutive images [41, 47, 59, 66]. However, the estimation of the optical flow, which is usually based on gradient methods or block matching methods, is computationally expensive. Methods using optical flow information generally fall into one of two categories. The first category can classify camera motion through estimation of model parameters from the computed optical flow [41, 47]. The other directly analyzes the observed optical flow patterns without any motion model by using the angular distribution or the power of optical flow vectors [59, 66]. On the
other hand, since video data is usually available in MPEG-compressed form, it is desirable to directly process the compressed video without decoding [1, 31]. These approaches use MPEG motion vectors as an alternative to optical flow. This can save a high computational load in two steps: the full decoding of the bitstream and the optical flow computation. However, they still have expensive computations.

2.4 Video Abstraction

In this section, some of the approaches that have been proposed for video abstraction are reviewed. Since the proposed research is concerned with video summarization, this work will focus primarily on these techniques and only briefly discuss video skinning.

2.4.1 Video Summarization

To produce a video summary, the first step is to extract a set of key frames that will be used to provide a summary to the user. These key frames are often arranged as storyboards, in which the key frames represent shots and sequences to summarize the story. If key frames are properly selected, then many image content searches can be conducted on the key frames rather than the complete video. The key frames greatly reduce the computational requirements for such searches. It is worth noting, however, that the definition of a key frame is subjective, not unique, and application dependent.

Many different approaches for automatic key-frame extraction have been proposed. Most of these may be classified into one of three types of approaches: sample-based, shot-based, and segment-based [67]. The fundamental differences between these approaches are briefly summarized in the sections that follow.

2.4.1.1 Sample-based Key-frame Extraction

In much of the early work in video summarization, key frames were selected by randomly or uniformly sampling the video frames at certain time intervals [60]. Although
simple and fast, the problem with this approach is that important key frames may be missed within video segments that are short in duration, while multiple key frames that are similar in content may be generated for video segments that are of long duration.

2.4.1.2 Shot-based Key-frame Extraction

Instead of sampling the video at specific or random time intervals, a more sophisticated approach would be to extract key frames in a way that adapts to the dynamic content of the video. For example, recall that a shot is defined as an unbroken sequence of frames taken with a single camera. Since each shot may be considered to provide an important snapshot of a specific video clip, then it is reasonable to presume that a representative frame from each shot should be extracted and used in the video summary or storyboard.

Perhaps the simplest approach for extracting key frames would be to select the first frame of each shot [56, 48, 58]. However, while this may be an acceptable approach for stationary shots (those with little or no action or camera movement), one key frame per shot will not generally provide an acceptable representation of dynamic visual content. Therefore, in these cases, multiple key frames should be extracted. An alternative would be to extract key frames by adapting, in some way, to the underlying semantic content. The key frame extraction methods that use this approach typically interpret the content by employing some low-level visual features such as color and motion. Depending upon the features that are used, these methods may be categorized into the following three different classes: color-based approaches, motion-based approaches, and mosaic-based approaches.

In a color-based approach proposed by Zhang et al. [25, 68, 13], key frames for each shot are extracted in a sequential manner. Specifically, the first frame of a shot is always selected as a key frame. Using a color histogram, the difference between the
current frame and the previous key frame is then computed. If this difference exceeds a given threshold, then the current frame is selected as a new key frame. Since the color histogram is invariant to image orientations and is robust to background noise, color-based key-frame extraction algorithms, such as the one proposed by Zhang, have been used widely. As discussed in [21], when choosing a frame that is close to the beginning or the end of a shot, there is a possibility that the frame is part of a dissolve or some other non-abrupt shot transition at the shot boundary, which strongly reduces its representative quality. The same can be said for frames belonging to shot segments containing high camera or object motion (i.e., strong panning or a zoomed object moving close to the camera and covering most of the frame surface). Such frames may be blurred, and thus in some cases, not suitable for extraction.

In contrast to the color-based methods, motion-based approaches are based on temporal dynamics in a shot. These approaches are computationally intensive and generally use pixel-based image differencing or optical flow computation. In Wolf's work [65], for example, the optical flow for each frame is computed. A simple motion metric is then evaluated as a function of time, and the frames having a local minima of motion are selected as the key frames.

A major drawback of using one or more key frames for each shot is that it does not easily scale up for long videos since scrolling through hundreds of images is time consuming, tedious, and often ineffective. The mosaic-based approach provides an alternative by generating a synthesized panoramic image that can represent the entire content in an intuitive manner. Mosaics, also known as "salient stills" [42] or "video sprites" [40], are usually generated using the following two steps [45]: 1) fitting a global motion model to the motion between each pair of successive frames, and 2) composing the images into a single panoramic image by warping the images with the estimated camera parameters. Although the mosaic image is more informative and visually more pleasing than a regular key frame, it only works well when specific
camera motions, such as panning or tilting, are detected. Generally, they cannot be effectively applied to real-world videos with complex camera effects and frequent background/foreground changes.

2.4.1.2 Segment-based Key-frame Extraction

To enhance the method of shot-based key-frame extraction, researchers have begun to work on a higher-level video unit, which is called a “video segment.” A video segment could be a single scene (collection of shots), an event, or even an entire sequence. In this context, the segment-based key-frame set will surely become more concise than the shot-based key-frame set because it employs a clustering method to integrate the shots [8].

2.4.2 Video Skimming

Video skimming is the second main approach to video abstraction. There are basically two types of video skimming: summary and highlight [67]. A summary sequence is used to provide users with an impression about the entire video content, while a highlight only contains the most interesting part of the original video, like a movie trailer that only shows some of the most attractive scenes without revealing the story’s end. Defining which video segments are the highlights is actually a very subjective process, and it is also a challenging project to map human cognition into the automated abstraction process [26]. Thus, most existing video-skimming work focuses on the generation of a summary sequence. One of the more direct approaches in this case would be to compress the original video by speeding up playback. As studied by Omoigui et al. [44, 2], the entire video could be watched in less time by fast playback with almost no pitch distortion, using time compression technology.

13
CHAPTER 3

METHODOLOGY

The purpose of this chapter is to introduce the abstraction paradigms that have been developed in this work, as well as the type of data to which it is applied. A short description of each of the building blocks composing the abstraction system is given in Section 3.1. Section 3.2 provides an introduction to the important aspects of working with compressed data. Finally, the chapter concludes with an overview of the basic techniques required to understand the algorithms developed in Chapters 4, 5, 6, and 7.

3.1 The Abstraction System

The purpose of this section is to introduce the reader to the abstraction system developed in this thesis. The system itself is composed of a number of building blocks, as illustrated in Figure 1. The rectangular boxes indicate the algorithms developed for quickly summarizing a video with little redundancy. This system consists of two main parts: video analysis and video abstraction. In the video analysis part, the key-frame extraction module begins by detecting shot boundaries. Each shot is analyzed for camera movement and divided, if appropriate, into sub-shots. Key frames are then selected according to the complexity of the shots. In the video abstraction part, the key frames that have been extracted in the video analysis part are used as the input to a fast clustering algorithm. Here, the goal is to cluster similar key frames so that a more compact summary may be created for high-level summarization. Finally, a non-repetitive summary of the video content from the clustered key frames is created.
Figure 1: Overall scheme for video abstraction.
This summarization is designed to support different user requirements for video browsing and content overview by outputting both the set of key frames and a summarized version of the original video using a multi-resolution structure. In addition, a simple and efficient algorithm for human face detection is presented to help users with more content-based searching.

3.2 Compressing Images

Through this work, "compressed data" refers to temporally related video images that have been transform-coded with the DCT applied on a block basis. Existing standards that adhere to this type of compression scheme include MPEG-1 and MPEG-2. Numerous references on the MPEG-1 and MPEG-2 encoding standards exist [32, 52], and therefore, only a brief synopsis of these is given below. Aspects of the encoding/decoding standards pertinent to the work at hand are described in more detail. Before further discussion of compressed data, a few terms that will be used throughout this work are defined.

3.2.1 Terminology

In the MPEG standard, each frame in a video sequence is one of three types: an I-, P-, or B-frame as illustrated in Figure 2. Intra-frames, also called I-frames, are video images that are encoded independently of any other frames. Inter-frames, or the other
hand, are predicted from a reference frame and involve motion compensation. P- and B-frames are examples of inter-frames. B-frames are encoded using bi-directional predictive coding using two reference frames, a past and a future frame, which can be either I-frames or P-frames. P-frames are encoded only from a past reference frame, which can be either an I-frame or a P-frame. The number of B-frames between two anchor frames is application dependent. A frame is partitioned into sets of 16 × 16 pixels, called macroblocks. There are four types of macroblocks: intra-coded, forward-predicted, backward-predicted, and bi-directionally predicted macro blocks. A macroblock (MB) consists of four 8 × 8 pixel blocks as shown in Figure 3.

![Figure 3: A single macroblock with the blocks.](image)

An overview of the process of compressing an I-frame is briefly described below.

### 3.2.2 Intra-frame Compression

Baseline intra-frame compression shown in Figure 4 consists of five basic steps:

1. Transform image to luminance/chrominance color space \((Y C_b C_r)\).
2. Reduce the color components (optional).
3. Partition image into 8 × 8 pixel blocks and perform the DCT on each block.
4. Quantize resulting DCT coefficients.

5. Entropy code the reduced coefficients.

![Diagram of encoder and decoder](image)

**Figure 4:** Baseline intra-frame (a) encoder and (b) decoder.

In the first step, a single video frame is first transformed into \( YC_bC_r \) color space, resulting in a three-band image. The \( Y \) band represents brightness information and is referred to as the luminance band. The \( C_b \) and \( C_r \) bands contain color information and are called the chrominance blue and chrominance red bands, respectively. The \( C_b \) (\( C_r \)) component of an image is the scaled representation of the blue (red) component in the RGB color space minus the brightness component. The reason for the separation of luminance and chrominance is that more redundant information can be removed from the chrominance component than from the luminance component [32].

The second step is optional, but it is used in standard practice. While the luminance component is left at full resolution, the color components are subsampled by two horizontally and vertically. The most common subsampling can be done by throwing out every other pixel or averaging blocks of four pixels, and this scheme is referred to as the 4:2:0 format shown in Figure 5. This step is the first lossy step, and the amount of data is reduced to one-half that of the original.

The third step consists of separating each of the \( Y, C_b, \) and \( C_r \) image components into sets or “tiles” of \( 8 \times 8 \) pixels. The elements within the tiles are converted to
signed integers (for pixels in the range of 0 to 255, subtract 128). These tiles are then transformed into the spatial frequency domain via the forward DCT. The upper left element of the $8 \times 8$ block is referred to as the DC coefficient. The sixty-three other elements are referred to as $AC_{uv}$, where $u$ and $v$ are the position of the element in the array. DC is the average value of the $8 \times 8$ original pixel values.

The fourth step is required to quantize these blocks with quantization coefficients. This is the fundamental information losing step. Simply stated, the DCT coefficients are divided by their corresponding quantization coefficients to eliminate small coefficient values and rounded to the nearest integer. These coefficients are simply numbers stored in an array. The greater the amount of compression desired, the coarser the quantization matrix and the greater the number of high frequency coefficients being set to zero.

The fifth and last step is lossless. The resulting numbers are strung together and encoded using Huffman codes, which actually represent different runs of different values. DC values are encoded as the difference between the DC value of the current block and the DC value of the previous block (Figure 6). This differential coding is possible because there is a strong correlation between adjacent DC values. The remaining sixty-three AC coefficients are encoded in zigzag scan order illustrated in Figure 7 using lookup tables and run-length encoding. This irregular ordering keeps
low frequency coefficients together. Low frequency coefficients are more likely to be nonzero. Typically, the high-frequency coefficients create long strings of zeros which can be easily run-length encoded. Finally, the intra-frame compression process is completed by Huffman coding the reduced data related to all of the DCT coefficients of the frame.

3.2.3 Inter-frame Compression

Unlike intra-frames, inter-frames are not independently encoded. Instead, motion compensation is used on a MB basis to predict the content of the inter-frames. For
each MB of an inter-frame, a motion vector for the best matching patch of the reference image (within a search window) is determined. The difference between the matching patch and the pixels contained in the inter-frame MB is referred to as the error term. The DCT of the error is what is encoded in the MPEG bitstream. In P-frames, only motion vectors for previously decoded frames are determined in a process referred to as forward motion compensation. On the other hand, B-frames are predicted using bi-directional motion compensation, where motion vectors are determined from both the previous and subsequent reference frames. The application of motion compensation is responsible for the bulk of compression savings in the MPEG-1 and MPEG-2 schemes because of the strong correlation of frames that have been captured at a sufficiently high sampling rate. The remainder of the inter-frame compression process is the same as for intra-frames.

3.3 Reconstructing DC-images

The low-resolution images that are used in this work are the DC-images even though most part of the proposed system works in the compressed domain. When the term “compressed domain processing” is used in this work, it refers to any processing that requires partial decoding of compressed image data; this is illustrated by the dotted line in Figure 8. Therefore, a DC-image itself is still compressed data as described in the following sections. The following steps must be considered before the abstraction system begins:

1. Entropy decode the coefficients.

2. Motion vector reconstruction (inter-frames only).

3. Dequantize resulting DCT coefficients.

The purpose of using the DC-image is to avoid the application of inverse discrete cosine transform (IDCT), which consumes 40% of the decoding time to display a
frame, and to obtain a low-resolution image. A DC-image is sixty-four times smaller than the original.

The two-dimensional DCT in MPEG is

\[ \text{DCT}(u, v) = \frac{1}{4} C_u C_v \sum_{i=0}^{7} \sum_{j=0}^{7} \cos \left( \frac{(2i+1)\pi v}{16} \right) \cos \left( \frac{(2j+1)\pi u}{16} \right) f(i, j) \]  

(3.1)

where \( f(i, j) \) is the 2-D matrix to be transformed, and the normalizing coefficients are given as

\[ C_n = \begin{cases} \frac{\sqrt{2}}{2}, & \text{for } n = 0, \\ 1, & \text{otherwise}. \end{cases} \]  

(3.2)

The basis functions of the two-dimensional DCT are shown in Figure 9. The square in the top-left corner of the basis image corresponds to the DC term of the DCT and is solid in color. In fact, the DC coefficient of a block corresponds to 8 times the average of all the pixel values contained within the block. The remaining squares in Figure 9 represent the AC coefficients and indicate the frequency of gray-scale variation in a specific direction.

In practice, a DC-image can be obtained from two different types of frames: intra-frame and inter-frame. A detailed explanation of reconstructing DC-images can be found in [71].

3.3.1 DC-images from Intra-frames

Since I-frames are coded without reference frames in the sequence, their DCT coefficients can be obtained without using any other processes. A DC-image is reconstructed by choosing the DC coefficient of each block \( P_i \) in an I-frame as shown in
Figure 9: The basis functions for all of the two-dimensional DCT coefficients.

Figure 10. One pixel in a DC-image is used to represent the original sixty-four pixel block.

3.3.2 DC-images from inter-frames

The reconstruction of DC-images for P- and B-frames is slightly more complicated than for I-frames since motion information must be utilized to derive the DC-images. In Figure 11, suppose \( P_{\text{ref}} \) is the current block of interest and \( P_1, \ldots, P_4 \) are the four original neighboring blocks from which \( P_{\text{ref}} \) is derived. Let \( h \) and \( w \) be the height and the width of \( P_{\text{ref}} \cap P_i \), respectively. The shaded regions in \( P_1, \ldots, P_4 \) are moved by \((\Delta x, \Delta y) = (w_1, h_1)\). Due to the linearity of the DCT, the DC coefficient of \( P_{\text{ref}} \) is of the form

\[
DCT(P_{\text{ref}})_{00} = \sum_{x=1}^{4} \left( \sum_{n=0}^{7} \sum_{m=0}^{7} w_{nm}^{i} \text{DCT}(P_i)_{nm} \right),
\]

where the \((i,j)\) component of \( P_k \) is denoted by \((P_k)_{ij}\), and \( w_{nm}^{i} \) is a weighting factor for the contribution of \( \text{DCT}(P_i)_{nm} \). The following equation was presented in [6],

\[
w_{nm}^{i} = \text{DCT}(S_1)_{nm} \times \text{DCT}(S_2)_{00}
\]

23
\( S_{ij} \) is a matrix determined by the position of block \( P_i \) as shown in Table 1. Matrix \( S_{ij} \) serves to move the subblock of interest in \( P_i \) to the position in \( P_{ref} \). Each \( I_n \) is an identity matrix of size \( n \). If each block is represented as an \( 8 \times 8 \) matrix, then the reference block \( P_{ref} \) can be described in the spatial domain through matrix multiplication as

\[
P_{ref} = \sum_{i=1}^{4} S_{1i} P S_{2i}.
\]

(3.5)

3.3.3 Working with DC-images

In this work, DC-images are used for shot change detection in Chapter 4, key-frame clustering in Chapter 9, and face detection in Chapter 6. Although the DC-image is substantially smaller than the original image, it still preserves the most of significant details as seen in Figure 12.

In summary, working with DC-images brings the following advantages to the abstraction process (to be described in future chapters):

1. Global image details remain present in DC-images, while high frequency pixel variations are filtered out.

2. Working with DC-images eliminates the need for performing a full IDCT on each image block.

3. The number of pixels to be processed is reduced by a factor of sixty-four.
Figure 10: An example of reconstructing a DC-image by choosing DC coefficients in an intra-frame.

Motion vector = \((a_x, a_y) = (w_i, h_i)\)

Figure 11: Reference block \((P_{ref})\), motion vector, and original block.
Table 1: Matrices $S_3$ and $S_4$.

<table>
<thead>
<tr>
<th>Subblock</th>
<th>Position</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>Lower right</td>
<td>(egin{bmatrix} 0 &amp; 1 \ 0 &amp; 0 \end{bmatrix})</td>
<td>(egin{bmatrix} 0 &amp; 0 \ 1 &amp; 0 \end{bmatrix})</td>
</tr>
<tr>
<td>$r_2$</td>
<td>Lower left</td>
<td>(egin{bmatrix} 0 &amp; 1 \ 0 &amp; 0 \end{bmatrix})</td>
<td>(egin{bmatrix} 0 &amp; 1 \ 0 &amp; 0 \end{bmatrix})</td>
</tr>
<tr>
<td>$r_3$</td>
<td>Upper right</td>
<td>(egin{bmatrix} 0 &amp; 0 \ 1 &amp; 0 \end{bmatrix})</td>
<td>(egin{bmatrix} 0 &amp; 0 \ 0 &amp; 1 \end{bmatrix})</td>
</tr>
<tr>
<td>$r_4$</td>
<td>Upper left</td>
<td>(egin{bmatrix} 0 &amp; 0 \ 1 &amp; 0 \end{bmatrix})</td>
<td>(egin{bmatrix} 0 &amp; 1 \ 0 &amp; 0 \end{bmatrix})</td>
</tr>
</tbody>
</table>

(a) Original image (352 × 240)  
(b) DC-image (44 × 30)

Figure 12: An original image and its DC-image.
CHAPTER 4

VIDEO ANALYSIS

The algorithm described in this chapter produces a set of key frames. A fast and robust approach is presented for video analysis. The proposed scheme can detect shot changes using an adaptive threshold and can classify camera movements in MPEG compressed video. Specifically, the proposed scheme is based on the temporal and the spatio-temporal distribution of MB types for identifying abrupt and gradual shot boundaries, respectively. This method also partitions a shot into sub-shots through camera motion analysis using template-matching for interpreting the contents of a shot and summarizes a video with the number of key-frames based on the complexity of a shot. In order to evaluate the proposed algorithm, several successful methods are described, implemented, and compared with the proposed approach for shot change detection.

This chapter is organised as follows. An algorithm for fast shot change detection is described in Section 4.1, and in Section 4.2 we look at the classification of camera motion. Key-frame selection is outlined in Section 4.3. In Section 4.4, several algorithms are described and compared through a set of experimental results obtained by several video sequences. Finally, a summary and some remarks are presented in Section 4.5.

4.1 Shot Change Detection

In this work, it is assumed that any sudden or gradual changes occurring between frames have a uniform distribution and occur over the entire frame.

The shot change detection algorithm presented in this research uses the properties
of each frame, particularly the B-frame, in an MPEG video stream. B-frames are encoded using bi-directional predictive coding using two reference frames, a past and a future frame, which can be either I-frames or P-frames. The proposed scheme for shot change detection is based on the number of bi-directionally predicted MBs within the first B-frame after reference frames. Using this B-frame can speed up the detection processing because the proposed method skips several frames between the nearest reference frames when a shot change does not occur. The basic assumption is that the larger the number of bi-directionally predicted MBs within a B-frame, the higher the correlation between the past and future anchor frames, and the higher the correlation, the less likely it is that a shot change occurs between these frames. Thus, the number of bi-directionally predicted MBs within the first B-frame after an I- or P-frame can be used as a measure of the correlation between the past and future I- or P-frames and can be used as a detector of shot changes.

4.1.1 Abrupt Shot Change Detection

For simplicity, it is assumed that the group of picture (GOP) has only two B-frames within the nearest anchor frames as illustrated in Figure 13. An abrupt shot change may occur at three different points within the nearest two I or P anchor frames.

![Figure 13: Abrupt shot change points between two anchor frames.](image)

The algorithm first checks whether the number of bi-directionally predicted MBs, \( N_{Bi} \), in the first B-frame after an anchor frame is small or not since this number
represents the correlation between the nearest anchor frames. If this number is small, the number of backward predicted MBs, \( N_B \), and the number of forward predicted MBs, \( N_F \), are considered to find the exact shot change point. The ideal distribution of MB types in each case is shown in Table 2. In Figure 13(a) and Case A of Table 2,

<table>
<thead>
<tr>
<th>Frame</th>
<th>MB</th>
<th>( N_B )</th>
<th>( N_F )</th>
<th>( N_L )</th>
<th>( N_P )</th>
<th>( N_M )</th>
<th>( N_B )</th>
<th>( N_F )</th>
<th>( N_L )</th>
<th>( N_P )</th>
<th>( N_M )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case A</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Slight</td>
<td>Low</td>
<td>0</td>
<td>Low</td>
<td>0</td>
<td>High</td>
</tr>
<tr>
<td>Case B</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>0</td>
<td>Low</td>
<td>0</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Case C</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>0</td>
<td>Low</td>
<td>0</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

for example, a shot change occurs at the first B-frame since \( N_F \) is small compared to \( N_B \), which shows that the current frame is mainly coded from the future anchor frame and belongs to another shot. Similarly, in Figure 13(b) the second B-frame is a shot change point because \( N_F \) is large compared to \( N_B \) in the first B-frame and small in the second B-frame as shown in Case B of Table 2. When \( N_F/N_B \) is large in both B-frames, a shot change must have occurred at the future anchor frame since the two B-frames, theoretically, are predicted from the past anchor frame as shown in Figure 13(c).

In order to use the number of bidirectionally predicted MBs as a shot change detector, it is necessary to define a threshold. The difficulty with the use of a threshold is that it should be applicable to any type of sequence, and it should be able to detect most types of shot changes. It is assumed that intensity values between consecutive frames of video do not vary much in uncompressed-domain processing. Since an encoder compresses a sequence with the same concept regardless of its application, the characteristics of \( N_B \) are similar and can be estimated by an average value within a shot of a compressed sequence. Consequently, detecting a shot change is equivalent to finding the point whose value is far away from the average value of \( N_B \). To avoid
local minima and to satisfy a single pass implementation, an accumulated average is preferred. Therefore, consider the following adaptive threshold that is applied to the first B-frame after an I- or P-frame at time \( 0 \) in an MPEG video stream:

\[
T_B[n] = \frac{1}{h} \cdot \frac{N_r}{N} \sum_{k=0}^{N_r-1} \frac{N_{MB}(n-k)}{N} \tag{4.1}
\]

for \( n > 1 \), and \( T_B[0] = 0.25 \) for \( n = 0 \). Here, \( h \) is a scaling factor, \( N \) is the number of MBs in the B-frame, \( N_{MB}[n] \) is the number of bi-directionally predicted MBs in the \( n \)th frame, and \( N_r \) is the number of frames used to determine the adaptive threshold value. To select the proper scaling factor \( h \) and to verify the effectiveness of the above threshold, the **Bayesian decision rule for minimum cost** is used to classify \( N_{MB}[n] \) in a frame into a shot change class and a non-shot change class as follows [15]:

\[
R(i|X) = C_{11} \cdot f(w_1|X) + C_{00} \cdot f(w_0|X) \tag{4.2}
\]

\[
R(0|X) = C_{00} \cdot f(w_0|X) + C_{00} \cdot f(w_0|X) \tag{4.3}
\]

\[
R(1|X) \left\{ \begin{array}{l}
\frac{w_1}{w_0} \quad \text{if } R(1|X) \\geq R(0|X)
\end{array} \right. \tag{4.4}
\]

where \( w_1 \) and \( w_0 \) denote the two classes, respectively. A **posteriori probability** is denoted by \( f(w_1|X) \), i.e., the probability of being in class \( w_1 \) given sample \( X \). \( C_{00} \) and \( C_{11} \) are the cost coefficients of correct classifications; \( C_0 \) and \( C_{10} \) are the cost coefficients of false classifications. Therefore, \( R(i|X) \) is the "cost" of classifying an unknown sample \( X \) into class \( w_i \).

In the proposed scheme for shot change detection, the two classes are the shot change class \( w_1 \) and the non-shot change class \( w_0 \). Applying the **Bayesian Formula** (6.6) to the above equations, the decision rules are

\[
f(w_1|X) = \frac{f(X|w_1) \cdot f(w_1)}{f(X)} \tag{4.5}
\]

\[
f(X|w_1) \frac{w_1}{w_0} \geq \tau \tag{4.6}
\]

where

\[
\tau = \frac{(C_{10} - C_{11}) \cdot f(w_0)}{(C_{00} - C_{00}) \cdot f(w_1)} \tag{4.7}
\]
In the above equations, \( f(u) \) is the corresponding \textit{a priori} probability of class \( u \). A conditional probability density, \( f(X|u) \), is described by the functions of bidirectionally MBs (bi-MBs) of the first B-frames in the shot change or non-shot change frames as shown in Figure 14. These statistics, \( f(X|u_1) \) and \( f(X|u_0) \), were obtained by fitting shot change and non-shot change MBs into the Gaussian models, respectively, from ten sequences selected regardless of their nature. The amount of false alarms and false dismissals allowed with the threshold \( \tau \) can be controlled. However, these two curves are not given in practice. In Figure 14, the problem of finding shot change points is interpreted as that of detecting points below a threshold \( \beta \) which is away from the average of the non-shot change class. \( \beta \) may be simplified by

\[
\beta = \frac{1}{\hat{n}} \int x \cdot f(X|u_0) \cdot dx.
\]

In order to use this threshold regardless of the type of video sequence and to satisfy a single-pass implementation, the average of the number of non-shot change MBs,
\[ f(x) \cdot f(X|x_0) \cdot dx, \]
is replaced with their accumulated average as in Equation (4.1).
Therefore, \( \beta \) can be approximated to the adaptive threshold \( T_B[n] \). The range of \( h \) is greater than 1, but it has been shown empirically that \( 2 \leq h \leq 2.5 \) is enough to detect most shot changes. With this adaptive threshold, a fast and effective shot change detection algorithm that is based on the temporal distribution of the MB types in MPEG compressed video can be described as follows:

1) If \( N_B[n]/N \) is greater than the threshold \( T_B[n] \) at the \( n^{th} \) B-frame, then a strong correlation exists between the past and future I- or P-frames with respect to the current B-frame. Therefore, it is assumed that there is no abrupt shot change between the current B-frame and the next I- or P-frame. The algorithm repeats this step for the next B-frame that follows the next I- or P-frame. Otherwise, it is assumed that an abrupt shot change occurs either at one of the B-frames or the future I- or P-frame.

2) If \( N_B[n] \) is large compared to \( N_P[n] \), then most of the MBs for the B-frame are coded from the next I- or P-frame. Therefore, the shot change must have occurred at the current B-frame (Figure 13(a)). The algorithm then declares an abrupt shot change and goes back to step 1. However, if \( N_B[n] \) is smaller than \( N_P[n] \), then the algorithm checks the next B-frame against the same procedure used for the previous B-frame. If it is satisfied, then the shot change is declared at the next B-frame (Figure 13(b)), and the algorithm goes back to step 1. Otherwise, the algorithm proceeds to step 3.

3) Theoretically, the shot change should have occurred at the next I- or P-frame (Figure 13(c)). In a P-frame, the algorithm checks if \( N_P[n] \) is above \( N_{ref}[n] \) (backward and bi-directionally predicted MBs do not exist). The algorithm goes to step 1.
Note that if $N_B$ and $N_P$ are not checked when a shot change occurs at the second B-frame or a P-frame, respectively, the algorithm can generate a false positive detection in the case of the camera moving or in the case of an unstable effect made by the encoder. To increase accuracy, the frame and histogram difference methods using DC-images [71] at the nearest two reference frames of each shot change point is used if the shot change detector $N_B/N$ is below and close to the threshold $T_B[n]$.

### 4.1.2 Gradual Shot Change Detection

To detect a gradual shot change, the properties of what are referred to as sub-macroblock images are examined as described in Figure 15. The algorithm constructs an image from the spatial distribution of bi-directionally predicted MBs in a selected B-frame, which is referred to as an MB image. Suppose the image is uniformly and consecutively divided into several non-overlapping square areas. These areas are referred to as sub-macroblock (sub-MB) images. When a sequence has a gradual

![Diagram of video sequence showing frames, MB images, sub-MB images, and computations in each sub-MB images.](image)

**Figure 15**: Proposed algorithm for detecting gradual shot changes.
transition over several frames, the statistics of $N_M$ in each sub-MB image are similar to one another through selected frames because it is assumed that a gradual effect (dissolve or fade) will occur uniformly throughout each frame. Therefore, to detect a gradual shot change is to identify the frames that have similar sub-MB characteristics. The algorithm begins by defining the variance of $N_M$ within each sub-MB over the past frames as

$$
\sigma_{nm}^2[n,i] = \frac{1}{N_p} \sum_{k=0}^{N_p-1} (N_M[n-k,i] - \bar{m}_{sub}[n,i])^2.
$$

(4.9)

Here, $\sigma_{nm}^2[n,i]$ is the variance of the $i$th sub-MB image at the current frame $n$, $N_p$ is the number of past frames used to compute the variance, $N_M[n,i]$ is the number of bi-directional MBs in the $i$th sub-MB image, and $\bar{m}_{sub}[n,i]$ is the average of $N_M[n,i]$ over $N_p$ frames in the $i$th sub-MB. In order to detect gradual shot changes, a measure of the similarity (or dissimilarity) between frames is required. To this end, a dissimilarity measure is defined as

$$
\text{Dis}[n] = \frac{1}{N_s} \sum_{k=0}^{N_s-1} (\sigma_{nm}^2[n,k] - \bar{m}_{n}[n])^2.
$$

(4.10)

It is assumed that this measure will be small and last over 0.5 second (about 15 frames) during gradual transitions. $N_s$ is the number of sub-MB images, and $\bar{m}_{n}[n]$ is the average of the variances of the sub-MB images. The following measure is the refined dissimilarity used after thresholding:

$$
D[n] = \begin{cases} 
1 & \text{if } \text{Dis}[n] \geq \tau_{\text{Dis}} \\
0 & \text{otherwise.}
\end{cases}
$$

(4.11)

Even though the measure $\text{Dis}[n]$ stays below the threshold, the proposed algorithm may have false positive alarms when a shot has a static motion or a dynamic motion. In order to solve this problem, the template matching method for the camera motion analysis described in Section 4.2 is performed. When $\text{Dis}[n]$ is below the threshold, if the motion of the selected frame is due to neither stationary motion nor
camera motion, the frame is a candidate for a gradual shot change. A summary of the algorithm for gradual shot change detection follows:

1) Check \([s, e] = \{n | D[n] = 0, \forall n\}\) where \(s\) is the starting frame, and \(e\) is the ending frame in a gradual shot change candidate. Once the measure \(D[n]\) is 0, the window \([s, e]\) grows until \(D[n]\) is equal to 1. If the measure is 1 and the duration \((e - s)\) is longer than 0.5 sec., then the algorithm proceeds to step 2. Otherwise, \([s, e]\) is reset, and this step is repeated.

2) Check the refined portion of a gradual shot change using

\[
[s', e'] = \{n | \Psi[n] \not\in \Psi, \forall n \in [s, e]\},
\]

where \(\Psi[n]\) is a gradual shot change candidate for the selected \(n^{th}\) frame, and \(\Psi\) is the set of seven basic motion classes: static, pan-left, pan-right, tilt-up, tilt-down, zoom-in, and zoom-out. If \((e' - s')\) is longer than 0.5 sec, then a gradual shot change is declared over this interval; otherwise, the algorithm goes back to step 1.

### 4.2 Camera Motion Classification

The proposed approach for fast video summary has focused on the speed of overall processing. To this end, a novel framework using motion vectors and their spatio-temporal distributions is presented for the analysis of camera motion to index MPEG compressed video faster than real-time. In order to consider camera motion analysis in an MPEG compressed video, it is necessary to focus on several factors in order to obtain a sufficient quality of results for interpreting camera movements. First, preprocessing is necessary to eliminate noisy motion vectors extracted from the bitstream. Second, the processing should be resilient to the presence of moving objects of large size. Finally, it should be fast to classify the camera motion types that are sufficient to describe various real video sequences. However, there has not been extensive work
addressing automatic camera motion characterization well in compressed video in terms of the above factors.

In the proposed analysis of camera movements, it is assumed that the small areas at the four corners of a frame are background regions where the camera events being considered occur. The overall procedure for the camera movement analysis is shown in Figure 16. It consists of three main steps. First, raw motion vectors for each P-frame are pre-filtered from the MPEG bitstream to form a motion vector field (MVF). Next, the MVF is divided into several non-overlapping square regions, these regions are referred to as sub-MVFs; these sub-MVFs are enforced to lie in either background regions or object regions. Finally, camera movements are classified by matching the predefined templates to the background sub-MVFs. A shot is segmented by camera motion characterization into sub-shot units based on the camera motion.

Figure 16: The overall camera motion classification scheme.
4.2.1 Preprocessing

The first step of the camera motion classification is to extract the motion vectors from the MPEG-1 or MPEG-2 bitstreams to construct the MVFs for each P-frame. The P-frame MVFs are then used to construct an MVF for each I-frame by interpolating the motion vectors between the two nearest P-frames.

It is well known that MPEG motion vectors do not always correspond to true optical flow since the estimation of a motion vector is carried out to minimize the prediction errors in the compression process [74]. Therefore, in a nearly uniform region, motion vectors may look like random noise. Therefore, it is necessary to preprocess the MVF to enhance its reliability. The suspected noisy components are filtered out from the MVF by applying filters to the magnitude of the horizontal and the vertical components of the motion vector separately. For this purpose, simple median filtering is effective enough to remove random noise vectors from the background region.

Figure 17: An example of an MVF and sub-MVFs.
4.2.2 Sub-MVF Processing

Suppose an MVF is uniformly divided into several non-overlapping square regions. These sub-MVFs are separated into two categories: background regions and object sub-MVFs. An example of segmenting an MVF into $3 \times 3$ sub-MVFs shown in Figure 17. Each sub-MVF has a motion vector from the average of the motion vectors in the corresponding region of an MVF.

During camera movement, motion vectors tend to be highly correlated within a particular frame, as well as between adjacent frames. Therefore, the average magnitude for the $s$th sub-MVF in frame $n$ is defined as follows:

$$M[s, n] = \frac{1}{N_s} \sqrt{ \left( \sum_{k=1}^{N_s} m_{h,k}[s, n] \right)^2 + \left( \sum_{k=1}^{N_s} m_{v,k}[s, n] \right)^2 }, \quad (4.13)$$

where $m_{h,k}[s, n]$ and $m_{v,k}[s, n]$ are the horizontal and vertical motion vector components for the $k$th MB of the $s$th sub-MVF, respectively, and $N_s$ is the number of MBs in the $s$th sub-MVF of the selected frame. Similarly, the average argument is defined as

$$\alpha[s, n] = \tan^{-1} \left( \frac{\sum_{k=1}^{N_s} m_{v,k}[s, n]}{\sum_{k=1}^{N_s} m_{h,k}[s, n]} \right). \quad (4.14)$$

In Equation (4.13), the average magnitude $M[s, n]$ is used to determine whether or not a given frame has a sufficient amount of motion associated with basic camera operations, and the average argument, $\alpha[s, n]$, is used to classify camera operations into one of six different classes.  

4.2.3 Template Motion Matching Processing

The first step in camera motion classification is to determine whether or not there is any camera motion. The measure used to determine if there is camera motion is

$$D_{00}[s] = \frac{1}{N_0} \sum_{s \in R} S[s, n], \quad (4.15)$$

38
where \( N_{B} \) is the number of background sub-MVF s, \( B \) is the set of background sub-MVF s, and

\[
\delta(s, n) = \begin{cases} 
1, & \text{if } M[s, n] \geq \tau_{\text{mag}}, \\
0, & \text{otherwise},
\end{cases}
\]

(4.16)

is an indicator function that is applied to the MVF magnitude function. If \( D_{B}[n] \) is greater than or equal to a threshold \( \tau_{D} \), it is assumed that camera motion is present, and the next step is to determine the type of motion using template motion matching. The templates used to classify the six basic camera motions are illustrated in Figure 18. In order to determine the most suitable template for camera movement within a given frame, the measure \( T_{b}[i] \) is defined by

\[
T_{b}[i] = \arg \min_{i} \left\{ \sum_{s \in B} \left| H[s, n] - H[x_{T}[s, i]] \right| \right\},
\]

(4.17)

where \( H[s] \) is the sixteen-bin histogram of the \( s^{th} \) sub-MVF in frame \( n \), and \( x_{T}[s, i] \) is the argument of the \( s^{th} \) sub-MVF for the \( i^{th} \) template. All sub-MVF s on background sub-MVF s are compared against the corresponding regions of the pre-defined motion templates. Therefore, \( T_{b}[i] \) finds the template whose histogram most closely matches the histogram of the given frame. If the match is close enough, i.e., if \( T_{b}[i] \) is greater
than or equal to a threshold $t_{temp}$, then the $n^{th}$ frame is tagged as having the given camera movement. A summary of the algorithm for analysis of the camera movements follows.

1. If $D_{p}[n]$ is greater than or equal to the threshold $t_{D}$ at the selected $n^{th}$ frame, then the algorithm proceeds to step 2; otherwise, the algorithm gets the next P-frame and repeats this step.

2. If $T_{c}[j]$ is less than or equal to the threshold $t_{temp}$, the selected $n^{th}$ frame is a candidate for the $n^{th}$ template camera movement; otherwise, the algorithm gets the next frame and goes back to step 1.

Note: In order to declare a camera movement (which is a sub-shot having homogeneous camera motion), the duration of the camera movement must be at least ten frames (almost 0.3 sec.) in duration.

4.3 **Key-frame Selection**

Once a video is segmented into shots and sub-shots by the shot change detection and camera motion analysis described in Sections 4.1 and 4.2, respectively, a simple and effective scheme for key-frame extraction is applied to the sub-shots in a sequential fashion. Since a shot is defined as a video segment within a continuous capture period, a natural and straightforward way to automate the key-frame extraction is to choose the first frame of each shot as its key-frame. However, while being sufficient for stationary shots, one key-frame per shot does not provide an acceptable representation of dynamic visual content. Therefore, multiple key-frames should be extracted by adapting to the underlying semantic content. To this end, key frames based on the complexity of each shot are extracted in order to keep the contents of a sequence for video summary. Simply, a key frame is selected at the end of a sub-shot that has a camera motion. However, in a no-motion sub-shot, a key frame is selected in the
middle of the subshot. This procedure mitigates the risk of extracting the blurred key frame at the boundary. Specifically, the key frame candidate is defined by

\[
K(m, k) = \begin{cases} 
\text{mid(subshot)}, & \text{if } n \notin M \\
\max(n), & \text{otherwise},
\end{cases}
\]

(4.18)

where \(K(m, k)\) indicates the \(k\)th key-frame candidate at the \(m\)th shot, \(M\) indicates a subshot having a camera movement in the \(m\)th shot in a stream, and \(\text{mid}(x)\) is the middle frame of a subshot \(x\). If there is no motion in a subshot, the algorithm selects a key-frame candidate in the middle of the subshot. Otherwise, this algorithm selects the last frame of a subshot as a key-frame candidate. A frame from the first subshot should be extracted as a key frame. The difference \(d(K(m, k_{\text{last}}), K(m, k))\) is then computed between the next key-frame candidate \(K(m, k)\) and the last extracted key frame \(K(m, k_{\text{last}})\) using a color histogram of the DC-image [70]. Once the difference exceeds pre-defined threshold \(N_k\), the current frame is selected as a new key frame, that is

1. \(k_{\text{last}} = 0\)

2. \(k_{\text{last}} = k\) if \(d(K(m, k_{\text{last}}), K(m, k)) > N_k\) for \(k \in [1, i_{\text{last}} - 1]\).

Here, \(N_k\) is the number of subshots within a shot. This algorithm produces at least \(m\) key frames, where \(m\) is the number of shots, according to the complexity of the video sequence.

### 4.4 Evaluation

In this section, the procedures of the proposed algorithm are briefly illustrated, and the experimental results are shown. In order to evaluate the proposed method for video analysis, the overall speed and accuracy of the results are focused on. The algorithm executes almost 10 times faster than real-time (real play time) for key-frame extraction. Also three successful shot change detection algorithms are described,
which are categorized into real-time and non real-time processing, and compared with the proposed algorithm.

4.4.1 Experimental Setup

Two test sets of MPEG-1 compressed data were selected to demonstrate the procedure of the algorithm and to compare the proposed scheme with three successful algorithms. The first test set consists of a movie sequence, two music videos, and a commercial film (CF). The second test set includes a movie sequence, a music video, and two commercial films (CF) sequences. All algorithms were implemented in C++ and run on a Windows 2000 system with a 400 MHz Pentium-II processor. For reference, it is assumed that the duration of each shot of an abrupt shot change is at least five frames, that of a gradual shot change is at least fifteen frames, that of camera movement classification is at least ten frames, and the required parameters are set as $h = 2, N_p = 5, N_s = 9, \alpha = 100, \tau_{DLP} = T_D |b|/5, \tau_{mag} = 4, \tau_D = 3/4, \tau_{temp} = 4$, and $N_c = 20\%$.

![Graph showing bi-MBs and candidates for abrupt shot change.](image)

**Figure 19:** Temporal bi-MBs and candidates for abrupt shot change.
4.4.2 Procedures and Experimental Results

For abrupt shot change detection, the part of the temporal distribution of bi-directionally predicted MBs in the first B-frames after I- or P-frames in the movie sequence is shown in Figure 19. The adaptive threshold can be obtained by averaging all of the selected B-frames after the I- or P-frames. All of the first B-frames below the adaptive threshold $T_d[r]$ are candidates, which may have abrupt shot change frames around them. For example, the frame numbers for labelled candidates from (a) to (d) in Figure 19 are 526, 544, 598, and 625, respectively. Table 3 summarizes all of the parameters, which are used to identify the exact shot change frames, and presents the results from the proposed algorithm around the sample candidates. As can be seen in Table 3, the first sampled candidate has a shot change at frame 526, which is determined by conditions 1) and 2) in Section 4.1. The second candidate declares a shot change at frame 544 with 1), 2), and 3), and the third one at frame 599 with 1) and 2). The last candidate, however, does not have any shot change because condition 2) is not satisfied when it checks the prerequisite condition 1).

For a gradual shot change example, the part of the temporal characteristics of nine sub-MB images in the music sequence, which has a dissolve effect from frame 34 to 81, is shown in Figure 20(a). The result from the gradual shot change detection...
algorithm that finally sets values to 0 from the starting frame (s') to the ending frame (e') in the case of a gradual shot change is illustrated in Figure 20(b).

![Figure 20a: Temporal distribution of bi-MBs in sub-MB images.](image)

![Figure 20b: Gradual shot change result.](image)

**Figure 20:** Gradual scene change detection: (a) temporal distribution of bi-MBs in sub-MB images and (b) its result.

To illustrate the subshot segmentation results through camera motion analysis, the algorithm in Section 4.2 is applied to the part of the movie sequence. As shown in Figure 21, the sequence is completely partitioned into subshots without any overlap. If the durations of the camera movements are very low, all minor movements can be identified, and if the duration is set to a very large value, only major camera movements can be detected. Therefore, setting this duration is completely dependent.
Figure 21: The results of the subshot segmentation for a part of a movie sequence.

on the user and the particular application.

In order to validate the efficiency and robustness of camera motion classification, the detected camera motions are compared with those determined from the ground truth built by manual observation. The performance of camera movement detection was tested on three videos. They included two five-minute music videos (Sequences A and B) and one-hour movie sequence (Sequence C) that contain very complicated motions, including large object motions as well as various types of camera movements. The results are shown in Table 4, where the number of segments classified as camera motions is compared against the ground truth.

<table>
<thead>
<tr>
<th>Sequence</th>
<th># of segments</th>
<th>Correct</th>
<th>Miss</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>49</td>
<td>47</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>61</td>
<td>56</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>232</td>
<td>232</td>
<td>21</td>
<td>9</td>
</tr>
</tbody>
</table>

Most of the missing points in Sequence B came from slow motions in a music video since it has small amplitude motion vectors in slow camera motions which can be ignored by the proposed method for camera movement classification. In movie Sequence C, this method has difficulty in detecting some events when a camera moves
diagonally (panning or tilting diagonally) and zooms to a region away from the image center. However, these kinds of problems can be overcome by using more templates and adjusting the threshold for the amplitude of the motion vectors.

In summarizing a video with key-frame extraction, for example, two compressed sequences having different characteristics are used. The sequences come from a movie and a CF in the first test set. The extracted key frames are illustrated in Figures 22 and 23 for the above sequences using the scheme in Section 4.3. The key frames within the first 10 shots of the movie sequence are shown in Figure 22. This part of the sequence has two camera movements, a pan-left at the fourth shot and a tilt-up at the seventh shot. These two camera movements separate the shot into three sub-shots. However, the second key-frame candidate of the fourth shot is eliminated since it does not satisfy the described condition in Section 4.3 (i.e., it is visually similar to the previously selected key frame). Consequently, the fourth and the seventh shots have two distinct key frames and three distinct key frames, respectively, from two

![Figure 22: Key frames of a movie.](image-url)
motionless subshots and one motion subshot. Similarly, a 15 sec. long CF that has seven shots and two camera movements is summarized by the key frames shown in Figure 23. This sequence has a zoom-in and a zoom-out sequentially at the third shot. This shot has three distinct key-frames even though two camera movements divide it into five subshots. These results show that the key-frame selection method based on complexity of shots can effectively summarize the contents of the sequences.

![Figure 23: Key frames of a commercial film.](image)

4.4.3 Comparison of Algorithms

As mentioned in Section 1, the definition of "key frame" is subjective, not unique, and application dependent. Thus, different algorithms with shot change detection schemes
are evaluated. For a performance comparison, three successful methods are implemented: Nagasaka's method [46], Yeo's method [71], and Fernando's method [14].

- Algorithm A: Color-based histogram [46]. This method extracts DCT DC-coefficients and uses the color information in the bitstream. Histograms using DC components of blocks for the luminance, Y, and chrominance, Cb and Cr, are computed, and the histogram differences are applied. Both static and locally adaptive thresholds are used for peak finding. Median-filtered differences are used to detect gradual transitions by looking for a series of medium-high difference values, the majority of which need to be above a soft threshold. This algorithm identifies abrupt and gradual transitions and uses I-, P-, and B-frames. This processing is not real-time. In implementing this algorithm, the number of bins for histogram-based approaches was 128 per color.

- Algorithm B: DC-image difference [71]. DCT DC-coefficients values for I-, Y-, and B-frames are extracted (DC-coefficient values for P- and B-frames are reconstructed as in [6]) and used to construct a DC-image sequence. Differences between these DC-images are then computed. Results are presented for two metrics: the sum of the absolute DC-image pixel-to-pixel differences, and the bin-to-bin difference between histograms of the DC-image pixel luminance. Automatic thresholding is achieved by a sliding window technique - a peak is declared if it is greater than the second largest difference within the window by some factor. This algorithm detects abrupt, gradual transitions and camera flashes, and uses I-, P-, and B-frames. This method is not real-time processing either.

- Algorithm C: Use of MB types [14]. This scheme extracts the types of MBs for a selected B-frame in the bitstream and first uses the bi-directionally predicted MB to find the correlation between anchor frames (I- or P-frame). Once this
value is below the pre-defined threshold, this algorithm goes to its next stage for identifying the exact boundary points. The advantage of this method is that it is faster than real-time since it can skip unnecessary frames to determine shot change points. This algorithm identifies abrupt shot boundaries and the beginning points of gradual change. However, this scheme depends heavily on the threshold according to the nature of the video sequence and suffers from miss or false positive boundary detection in the case of camera motions and editing effects.

In order to evaluate the performance of the proposed shot change detection algorithm, this work used the second test set, including four compressed sequences that had different characteristics. These sequences consisted of a movie sequence (Sequence A), a music video (Sequence B), and two commercial film sequences (Sequences C and D). In particular, Sequence D had a flexible GOP structure (for example, IBBPBPBB).

The performance is given in the precision and recall rates. The experimental results of the methods mentioned above are listed on Table 5.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Proposed method</th>
<th>Yen's method</th>
<th>Fernande's method</th>
<th>Histogram method</th>
<th>Weighted</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
<td>Recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>0.910</td>
<td>0.773</td>
<td>0.897</td>
<td>0.833</td>
<td>0.068</td>
<td>0.570</td>
</tr>
<tr>
<td>Motion</td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
<td>Recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>0.913</td>
<td>0.795</td>
<td>0.918</td>
<td>0.757</td>
<td>0.746</td>
<td>0.651</td>
</tr>
<tr>
<td>CF1</td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
<td>Recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>0.917</td>
<td>0.818</td>
<td>0.993</td>
<td>0.864</td>
<td>0.813</td>
<td>0.619</td>
</tr>
<tr>
<td>CF2</td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
<td>Recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>0.911</td>
<td>0.764</td>
<td>0.945</td>
<td>0.857</td>
<td>0.977</td>
<td>0.997</td>
</tr>
<tr>
<td>Total</td>
<td>Precision</td>
<td>Recall</td>
<td>Precision</td>
<td>Recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>0.909</td>
<td>0.818</td>
<td>0.858</td>
<td>0.823</td>
<td>0.812</td>
<td>0.712</td>
</tr>
</tbody>
</table>

1 Precision is defined to be the number of correct key frames divided by the number of correct key frames plus the number of false key frames, whereas recall is the number correct divided by the number correct plus the number missed.
These results show that the proposed scheme for shot change detection is more effective than the other methods in speed and performance. It is believed that this algorithm can be used as a pre-filter for indexing and retrieval since the precision and recall ratios are very high, and the speed is fast.

It is important to be noted that the Fernando's scheme is faster than the proposed algorithm for shot change detection, but the proposed algorithm produces better results in terms of precision and recall rates because the Fernando's algorithm has been designed to detect mainly abrupt shot changes.

4.5 Summary and Remarks

This chapter presents a simple and fast processing algorithm for shot segmentation and camera motion classification. A scheme for summarizing a video with key frames is also introduced. In order to have this kind of video summary, shot segmentation and camera movement classification are performed initially, regardless of the type of video sequence. Motion analysis allows us to break down complex shots into a sequence of meaningful events. Then, a simple key-frame extraction algorithm selects the representative frames based on the complexity of a shot. Also, in order to evaluate the proposed method for shot change detection, three successful algorithms are implemented and compared using four video sequences. Experimental results show that the presented method is fast and can detect most shot changes independent of the nature of the sequences in the shots. It is believed that each module of the proposed scheme can be used as a pre-filter for video indexing, browsing, and retrieval.
CHAPTER 5

VIDEO ABSTRACTION I: CLUSTERING

The previous chapter focused on the speed of extracting key frames of a video sequence, but the compactness of the key frames, in terms of redundancy and number, is not considered even though it must be addressed for effective video abstraction. In order to create a concise summary of a video sequence, it is very important to ensure that the summarized representation of the original video contains little redundancy and gives equal attention to each video segment. Therefore, given a collection of key frames, we would like to cluster them into representative groups so that, with a hierarchical summary, we would be able to select a set of the most important key frames that summarize the video sequence. An important component in video summarization is the determination of how many key frames are required to provide a good summary and overall representation of the video with minimal redundancy. The singular value decomposition (SVD) is known for its capabilities of deriving a low-dimensional refined feature space from a high-dimensional raw feature space, and of capturing the essential structure of a data set in feature space [19]. However, the usefulness of the SVD for real applications has been severely limited because of the computational cost associated with finding the singular values and the singular vectors [9]. To reduce the computational complexity of the SVD algorithm, we propose to use color histograms of the DC-images in the compressed video stream to represent the features in the key-frame.

This chapter introduces useful properties of the SVD and uses them to quickly summarize a video sequence based on the visual similarities of its frames. To verify the effectiveness of these properties, the SVD is performed on an original feature space,
then the \( k \)-means algorithm for clustering is applied directly to this refined feature space. In order to evaluate the proposed scheme, the speed of the single \( k \)-means algorithm applied to the original feature space for clustering is compared with that of the proposed method.

This chapter is organized as follows. Some useful properties of the SVD are introduced in Section 5.1, and a fast key-frame clustering is presented in Section 5.2. In Section 5.3, some experimental results are provided, and finally, Section 5.4 summarizes this chapter.

### 5.1 Properties of the SVD

The SVD of an \( m \times n \) matrix \( A = [a_1, a_2, \ldots, a_n] \) is the decomposition of \( A \) into the product of three matrices as follows [19]:

\[
A = U\Sigma V^T = \sum_{k=1}^{p} c_k u_k v_k^T
\]

where \( p = \min(m, n) \), \( U = [u_1, u_2, \ldots, u_m] \) is an \( m \times m \) orthogonal matrix, \( V = [v_1, v_2, \ldots, v_n] \) is an \( n \times n \) orthogonal matrix, and \( \Sigma \) is an \( m \times n \) matrix with elements \( \sigma_k \) along the diagonal and zeros everywhere else. Matrix \( U \) is called the left singular matrix, \( V \) is called the right singular matrix, and \( \Sigma \) is the singular value matrix.

If the singular values are ordered so that \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \), and if the matrix \( A \) has a rank \( r < p \), then the last \( p-r \) singular values are equal to zero, and the SVD becomes

\[
A = \sum_{k=1}^{r} \sigma_k u_k v_k^T.
\]

The first property to establish is that the Euclidean distance between any two columns of \( A \) is equal to the distance between the two corresponding rows of \( V \) weighted by the singular values.

**Property 1:** If \( A \) is an \( m \times n \) matrix of rank \( r \) with a singular value decomposition \( A = U\Sigma V^T \), then the Euclidean distance between any two column vectors of \( A \)
is equal to the weighted Euclidean distance between the corresponding columns of $V^T$, where the weighting is by the singular values, $\sigma_k$, i.e.,

$$|a_i - a_j|^2 = \sum_{k=1}^r \sigma_k^2 (v_{ik} - v_{jk})^2$$

To establish this property, let $\phi_i$ be the $i$th column vector of $V^T$, i.e., $V^T = [\phi_1, \phi_2, \ldots, \phi_r]$. From the SVD in Equation (5.1), it follows that the $i$th column of $A$ is given by

$$a_i = U \Sigma \phi_i$$

Taking the difference between two column vectors, $a_i$ and $a_j$, we have

$$a_i - a_j = U \Sigma [\phi_i - \phi_j]$$

Therefore, it follows that the square of the norm of the difference is

$$|a_i - a_j|^2 = [\phi_i - \phi_j]^T \Sigma^T (U^T U) \Sigma [\phi_i - \phi_j]$$

Since $U$ is orthonormal and $\Sigma$ is a diagonal matrix, then

$$|a_i - a_j|^2 = [\phi_i - \phi_j]^T \Sigma^2 [\phi_i - \phi_j]$$

Thus, since $\phi_i = v_i^T$, and $v_{ik} = \phi_{ik}$, we have

$$|a_i - a_j|^2 = \sum_{k=1}^r \sigma_k^2 (v_{ik} - v_{jk})^2 = \sum_{k=1}^r \sigma_k^2 (v_{ik} - v_{jk})^2,$$

and the property is established.

**Property 2:** Let $A$ be an $m \times n$ matrix of rank $r$. Let $V = [v_1, v_2, \ldots, v_n]$ be the $n \times n$ right singular matrix of $A$, and $V^T = [\phi_1, \phi_2, \ldots, \phi_r]$. If two column vectors of $A$ are the same, $a_i = a_j$, then the first $r$ elements of the $i$th and $j$th columns of $V^T$ are equal, i.e., $\phi_i = \phi_j$.
To establish this property, if \( A = [a_1, a_2, \ldots, a_n] \), then from Equation (5.3) we have
\[
a_i - a_j = U \Sigma (\phi_i - \phi_j) = \sum_{k=1}^{r} \sigma_k u_k (\phi_i - \phi_j)
\]  
\( (5.5) \)
where \( u_k \) are the left singular vectors of \( A \). Note that if \( a_i = a_j \), then
\[
\sum_{k=1}^{r} \sigma_k u_k (\phi_i - \phi_j) = 0
\]
Since the singular vectors \( u_k \) are linearly independent, then the sum on the left can be equal to zero only if the scalars \( \sigma_k (\phi_i - \phi_j) \) are equal to zero for \( k = 1, 2, \ldots, r \). Therefore, since the rank of \( A \) is \( r \), then the singular values \( \sigma_k \) are non-zero for \( k = 1, 2, \ldots, r \), and it follows that \( \phi_{ik} = \phi_{kj} \) for \( k = 1, 2, \ldots, r \), and the property is established.

Note that by repeated application of this property, it follows that if two or more columns of \( A \) are identical, then the first \( r \) elements of the corresponding columns of the right singular matrix will be equal. It is also worth mentioning that it is easily to generalize this property to state that if two columns of \( A \) are proportional, i.e., \( a_i = \alpha a_j \), then the corresponding columns of \( \Sigma^T \) are proportional by the same constant \( \alpha \) to the rank of \( A \), i.e.,
\[
\phi_{ik} = \alpha \phi_{kj} \quad k = 1, 2, \ldots, r.
\]
Before continuing, for an example, consider the following \( 8 \times 6 \) matrix
\[
A = \begin{bmatrix}
0.9501 & 0.8314 & 0.9355 & 0.1389 & 0.1389 & 0.1389 \\
0.2311 & 0.4447 & 0.4169 & 0.2028 & 0.2028 & 0.2028 \\
0.6068 & 0.6154 & 0.1103 & 0.1987 & 0.1987 & 0.1987 \\
0.4860 & 0.7919 & 0.8206 & 0.6038 & 0.6038 & 0.6038 \\
0.8913 & 0.9296 & 0.0679 & 0.2722 & 0.2722 & 0.2722 \\
0.7623 & 0.7382 & 0.3209 & 0.1388 & 0.1988 & 0.1988 \\
0.4565 & 0.1763 & 0.8132 & 0.0153 & 0.0153 & 0.0153 \\
0.0185 & 0.0257 & 0.0099 & 0.7468 & 0.7468 & 0.7468 \\
\end{bmatrix}
\]
(5.6)
where columns four, five, and six are the same. Since the rank of \( A \) is four, then it follows from Property 2 that the first four elements of columns four, five, and six in \( V^T \) will be the same. The transpose of the right singular matrix of \( A \) is

\[
V^T = \begin{bmatrix}
-0.4994 & -0.5587 & -0.4893 & -0.2576 & -0.2576 \\
0.3446 & -0.0319 & 0.4491 & -0.4783 & -0.4783 \\
-0.3119 & -0.3629 & 0.7526 & 0.1161 & 0.1161 \\
-0.5081 & 0.7455 & 0.0177 & -0.1572 & -0.1572 \\
0.0000 & 0.0000 & 0.0000 & 0.8175 & -0.4082 & -0.4082 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.7071 & 0.7071
\end{bmatrix}
\]  

(5.7)

and, as seen, the first four elements of columns four, five, and six are the same. Note that \( V^T \) has the following structure

\[
V^T = \begin{bmatrix}
\Phi_0 & \Phi_1 \\
\Phi_2 & \Phi_3
\end{bmatrix}
\]  

(5.8)

where, as known from Property 2, \( \Phi_1 \) has the form \( \Phi_1 = [w_0, w_0, w_0] \). In addition, note that \( \Phi_2 \) is a zero matrix

\[
\Phi_2 = 0
\]

and \( \Phi_3 \) is matrix whose rows sum to zero, i.e.,

\[
\Phi_3 = \begin{bmatrix}
w_1^T \\
w_2^T
\end{bmatrix}
\]

where

\[
[1 \ 1]w_1 = [1 \ 1]w_2 = 0
\]

The next property states that this structure holds, in general, when \( A \) satisfies the appropriate set of conditions.

Property 3: Let \( A_1 = [a_1, a_2, \ldots, a_n] \) be an \( m \times n \) matrix with \( m \geq n \) and rank \( n \), i.e., the columns of \( A_1 \) are linearly independent. Let \( a_0 \) be a vector of length
$m$ that is linearly independent of the vectors $a_i$ in $A_1$, and let $A_2$ be an $m \times d$ matrix of rank one that is formed by repeating the vector $a_0$ $d$ times, i.e.,

$$A_2 = [a_0, a_0, \ldots, a_0]$$

Let $A$ be an $m \times (n+d)$ matrix that is constructed from $A_1$ and $A_2$ as follows,

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$$

with a singular value decomposition $A = U\Sigma V^T$. Suppose that we partition $V^T$ as follows

$$V^T = \begin{bmatrix} \Phi_0 & \Phi_1 & \Phi_2 \end{bmatrix}$$

(5.9)

where $\Phi_0$ and $\Phi_2$ have $n$ columns, the same number as $A_1$, and $\Phi_1$ and $\Phi_2$ have $d$ columns, which is the number of duplicates of $a_0$ in $A_2$. The partitioning by rows is done so that $\Phi_0$ and $\Phi_1$ have $n+1$ rows, which is the rank of $A$. With this partitioning of $V^T$, it follows that

$$\Phi_2 = 0$$

and

$$[1,1,\ldots,1]\Phi_2^T = 0$$

i.e., the sum of the elements in each row of $\Phi_2$ is equal to zero.

To establish this property, it begins by using the partitioning of $V^T$ given in Equation (5.9) and the fact that $V$ is orthonormal, to write

$$V^TV = \begin{bmatrix} \Phi_0\Phi_0^T + \Phi_1\Phi_1^T & \Phi_0\Phi_1^T + \Phi_1\Phi_2^T \\ \Phi_0\Phi_1^T + \Phi_1\Phi_2^T & \Phi_2\Phi_2^T + \Phi_2\Phi_2^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(5.10)

Therefore, from the upper right partition,

$$\Phi_0\Phi_1^T + \Phi_1\Phi_2^T = 0.$$  

(5.11)
Since, as shown in Property 2, the $d$ columns of $\Phi_1$ are the same, and we may express this matrix as

$$\Phi_1 = [w_0, w_0, \ldots, w_0] = w_0[1, 1, \ldots, 1] \quad (5.12)$$

Therefore, from Eqs. (5.11) and (5.12) it follows that

$$\Phi_0\Phi_0^T = -\Phi_1\Phi_1^T = -w_0[1, 1, \ldots, 1][\Phi_1]^T = -w_0[\lambda_1, \lambda_2, \ldots, \lambda_d] \quad (5.13)$$

where $\lambda_i$ is the sum of the terms in the $i^{th}$ column of $\Phi_0^T$, i.e., the sum of the terms in the $i^{th}$ row of $\Phi_2$.

Since $V$ is orthonormal, the rows and columns of $V$ are linearly independent. Therefore, the $(n+1)$ rows of the upper partition of $V^T$ are linearly independent, i.e.,

$$\Phi = \begin{bmatrix} \Phi_0 & \Phi_1 \end{bmatrix} = \begin{bmatrix} \Phi_0, & w_0, w_0, \ldots, w_0 \end{bmatrix}$$

has $n+1$ linearly independent rows and, consequently, $n+1$ linearly independent columns. However, since $\Phi_0$ has $n$ columns, and since each of the $d$ columns of $\Phi_1$ are the same, then the columns of $\Phi_0$ must be linearly independent, and $w_0$ must be linearly independent of the columns of $\Phi_0$.

From Equation (5.13), the $i^{th}$ column of the matrix product $\Phi_0\Phi_0^T$ is a linear combination of the columns of $\Phi_0$, which is linearly independent of the $i^{th}$ column vector on the right. Therefore, it follows that $\Phi_2 = 0$ and each $\lambda_i = 0$, and the property is established.

Although Property 3 has stated in terms of a matrix $A$ having the last $d$ columns the same, the property may be generalized to allow any set of $d$ columns being the same. Specifically, let $P$ be a permutation matrix$^1$. Right multiplication of $A$ by $P$ permutes the columns of $A$, and from the singular value decomposition, the following

---

$^1$ A matrix $P$ is a permutation matrix if exactly one entry in each row and column is equal to one, with all other entries zero. Left multiplication of a matrix $A$ by a permutation matrix permutes the rows of $A$ while right multiplication permutes the columns.
equation is given by

$$AP = UΣ(V^T P)$$

which shows that the columns of $V^T$ are permuted in the same way.

The last property is concerned with the norm of the column vectors in $Φ_1$. Specifically, it states that the norm square of these vectors is equal to $1/d$, where $d$ is the number of times that $w_0$ is duplicated in $A$.

Property 4: Let $A$ be an $m × n$ matrix that satisfies the constraints given in Property 3, and let $V$ be the right singular matrix, with $V^T$ partitioned as in Equations (5.8), with

$$Φ_1 = [w_0, w_0, ..., w_0]$$

It then follows that the Euclidean norm square of $w_0$ is

$$|w_0|^2 = 1/d$$

where $d$ is the number of duplicates of $w_0$ in $Φ_0$.

To establish this property, consider the following two matrices,

$$D_0 = \begin{bmatrix} Φ_1 \\ \vdots \\ Φ_2 \end{bmatrix}$$

and

$$D_1 = \begin{bmatrix} Φ_3 \\ Φ_4 \end{bmatrix}$$

that are formed from the partitioning of $Φ_0$ as given in Equations (5.8). In the following, it is going to be using the fact that since $V$ is an orthonormal matrix, then the sum of the squares of each row and of each column (the square of the Euclidean norm) is equal to one. Since $D_1$ has $d - 1$ rows, and since each row of $D_1$ corresponds to a
column of the orthonormal matrix \( V \), then the sum of the squares of the elements in \( D \) (the square of the Frobenius norm) is

\[
\|D\|_F^2 = \|\Phi_0\|_F^2 + \|\Phi_3\|_F^2 = d - 1
\]  
(5.14)

However, since \( \Phi_3 = 0 \), it follows that

\[
\|\Phi_3\|_F^2 = d - 1
\]

Now consider the matrix \( D_0 \), which has \( d \) columns with each column corresponding to a row of \( V \). Thus, with \( V \) being orthonormal, it follows that the sum of the squares of each column of \( D_0 \) is equal to one, and the following equation is expressed by

\[
\|D_0\|_F^2 = \|\Phi_1\|_F^2 + \|\Phi_3\|_F^2 = d.
\]

Therefore, using Equation (5.14), then

\[
\|\Phi_1\|_F^2 = 1.
\]

Finally, since

\[
\Phi_3 = [w_0, w_0, \ldots, w_0]
\]

then

\[
\|\Phi_3\|_F^2 = d\|w_0\|^2 = 1
\]

and the property follows.

### 5.2 Key-frame Clustering

To create the feature matrix \( A \) described in Section 5.1, key frames are taken from subsets of a video as described in Chapter 4. For this purpose, each DC-image of a key frame is divided into 9 blocks to get spatial information of the color distribution and create a 3-D histogram in RGB color space with 20 bins for R, G, and B for each block. These nine histograms are then concatenated together to form a 540-dimensional feature vector for the key frame. Next, a feature frame matrix for the
video sequence, \( A = [a_1, a_2, \ldots, a_n] \), is created, where the \( i \)th column vector \( a_i \) is the feature vector corresponding to key frame \( i \). Procedures for making feature matrix \( A \) is illustrated in Figure 24.

![Figure 24: Procedures for making feature matrix A.](image)

Since a small image block is not expected to contain a myriad of colors, the feature frame matrix \( A \) will typically be sparse. Therefore, SVD algorithms designed for sparse matrices can be used, and they are much faster and memory efficient than SVD algorithms for non-sparse matrices \([18]\).

A summary of the procedure for clustering the feature vectors is described below. Before clustering begins, however, the columns of the matrix \( A \) are normalized so that columns that were simply scaled versions of each other are now the same. This procedure is necessary in the following application.
Clustering Algorithm:

1. Perform the SVD of $A$.

2. Generate a new matrix $\hat{A}$ with columns $a_{il}$ for $l = 1, 2, \cdots, k$ and sort the columns of $\hat{A}$ column by column, in ascending order according to the norm of the column vector $e_j^T$ to the rank $k$ of $A$. This sorting is performed in order to speed up the clustering algorithm. Also, the norm of $a_{ij}$ indicates the content activity of a video sequence. The smaller the norm of a vector in $V^T$, the more likely there is repetition of key frames in $A$.

3. Apply the $k$-means algorithm to the sorted matrix $\hat{A}$.

In the above operation, the energy of the normalized matrix $A = [a_{ij}]$ is

$$\|A\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 = n = \sum_{i=1}^{\text{rank}(A)} \sigma_i^2,$$

and the value of $k$ is selected such that

$$k = \arg \min \left\{ \alpha : n \leq \sum_{i=1}^{k} \sigma_i^2 \right\}.$$

Using the experimental results, key frames were successfully clustered by setting $\alpha$ to 0.9991. In the clustering method, the key frame representing a cluster that has more than two key frames is selected according to the temporal order of the key frames in that cluster, i.e., the first key frame in time.

5.3 Evaluation

To show that the proposed algorithm for clustering performs competitively with existing algorithms, the commonly used $k$-means algorithm was selected for comparison in processing speed for key-frame clustering. The amount of time taken for clustering depends, of course, on the number of key frames that need to be clustered.
5.3.1 Experimental Setup

Four sets of key frames extracted from four different MPEG-I formatted video sequences with different characteristics are considered. These sequences include a movie sequence, a music video, a documentary, and a CF sequence. All algorithms were implemented in C++ and run on a Windows system with a 400 MHz Pentium-II processor.

5.3.2 Experimental Results

Once the key frames have been extracted automatically which are referred to as original key frames, similar key frames are combined manually based on their visual similarities, which are referred to as expected key frames, counted, and this number is used as the ground truth. Figure 25 shows the key frames for the movie sequences through the proposed clustering method, and the key frames are referred to as refined key frames. In Figure 25, the top box illustrates the extracted key frames from the key-frame extraction method within the first 12 shots of the movie sequence. This part of the sequence has two camera movements, which are a pan-left at the fifth shot and a tilt-up at the seventh shot. These two camera movements separate the shot into three subshots. However, the second key-frame candidate of the fifth shot is eliminated since it does not satisfy the condition pre-described by the key-frame extraction section; i.e., it is visually similar to the previously selected key frame. Consequently, the fifth and the seventh shots have two distinct key frames and three distinct key frames, respectively, from two motionless shots and one motion subshot.

The bottom box in Figure 25 shows the refined key frames after performing the fast clustering algorithm combining both the SVD and the k-means algorithms. Here, the fourth shot is merged with the second shot because these two key frames are similar in terms of visual content, but the second key frame comes first temporally. Similarly, the 10th and the 12th shots are merged.
Figure 25: Refined key frames for a movie.
The first 30 minutes of the movie was used as a test sequence for the speed comparison with a single k-means algorithm. In this comparison, it was assumed that the cluster numbers of the single k-means algorithm was equal to those of the proposed method in Equation (5.10). The k-means algorithm in the VOICEBOX toolbox (http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html) was used. The proposed algorithm for clustering was generated using MATLAB for consistency even though the proposed video summary system has been implemented using C++. In this test, the clustering speed comparison is only considered, and further analysis for clustering accuracy is provided with the previously mentioned four test sets. From Table 6 and Figure 26, it is shown that the proposed clustering method is much faster than the single k-means algorithm. As the number of key-frames to be clustered grows, the processing time difference between the compared algorithms increases. In Figure 26, the solid and the dotted lines show the results of the proposed clustering algorithm and the k-means algorithm, respectively. The speed of the proposed method is almost linear while that of the single k-means algorithm increases almost quadratically as the data grows. It means that the k-means algorithm using the reduced data dimension through the SVD algorithm is much faster than the single k-means algorithm directly applied to the original data.

The detailed evaluation results for the test set are shown in Table 7. In this table, correct key frames are the key frames clustered correctly, false key frames are the key frames that should be separated but belong to the wrong cluster, and missed key
frames are the opposite. This movie video (as is usually the case for action movies) contains many shots and subshots from camera movements and dialogue-like shots. As shown in Table 7, most of the visually similar shots have been properly merged quickly by the proposed method for clustering. The documentary video has more incorrectly clustered key frames than other sequences because it has some key frames which have quite similar color distributions but different visual contents. In the CF sequence, most shots are successfully clustered even though this sequence has a lot

![Graph of speed comparison results.](image)

**Table 7: Results for clustering accuracy.**

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Original</th>
<th>Expected</th>
<th>Refined</th>
<th>Correct</th>
<th>False</th>
<th>Missed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
<td>383</td>
<td>298</td>
<td>296</td>
<td>287</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>Music video</td>
<td>27</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Documentary</td>
<td>15</td>
<td>63</td>
<td>59</td>
<td>54</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>CF</td>
<td>14</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
of camera and editing effects. The average of precision² and recall rates are 0.96 and 0.94, respectively, for the test sequences. These results show that the proposed method can effectively and quickly summarize the contents of the sequences with little redundancy.

In order to show how this clustering algorithm can be used for hierarchical video abstraction and browsing, another movie sequence (from *Die Hard III: Die Hard with a Vengeance*, Disc 1), which is almost one hour long, and has 407 clusters, 821 shots, and 1537 key frames, is used. From these clusters, for example, 3 significant clusters can be selected, as illustrated in Figure 27, according to priorities that will be described in Chapter 7. For effective browsing of a video, each cluster is linked to its member shots, and each member shot is, in turn, connected to key frames.

5.4 Summary and Remarks

In this chapter, useful properties of the SVD were presented and illustrated how these properties could be used to reduce the amount of time it takes to cluster vectors in a video abstraction system. Experimental results show that the proposed method is a robust and fast solution for clustering. It is believed that the properties of the SVD are applicable to signal processing, and the proposed scheme can be used as a useful module for video abstraction and retrieval.

²Precision is defined to be the number of correct key frames divided by the number of correct plus false key frames, whereas recall is the number of correct divided by the number of correct plus the number missed.
Figure 27: Example of hierarchical video abstraction and browsing.
CHAPTER 6

VIDEO ABSTRACTION II: FACE DETECTION

The objective of this chapter is to provide a simple and efficient method to detect human faces in the compressed domain. The proposed detection algorithm consists of three stages: the skin detection stage, the template matching stage, and the confirmation stage. Specifically, first a sub-sampled image is extracted from a compressed video sequence using DC coefficients of the DCT, and skin regions are separated from non-skin regions using the Bayesian decision rule with a Gaussian mixture model (GMM). Then, the location of a face is detected by using a deformable template that considers shape and orientation of the face. In the final stage, face candidates are verified by using the moments computed directly from the DCT coefficients in the face region. In particular, the coefficients are manipulated differently according to the orientation of the face. Good results have been obtained for a large variety of video sequences. This algorithm can be applied to JPEG images without any modification as well.

This chapter is organized as follows. A brief overview of what other researchers have been doing to tackle the problem can be found in Section 6.1, detecting skin regions is described in Section 6.2, and their feature extraction is presented in Section 6.3. Identifying face regions are presented by doing template matching and computing the moments in Section 6.4 and 6.5, respectively. Experimental results are described in Section 6.6 through the test sets obtained by several MPEG compressed videos, and a summary and some discussions are presented in Section 6.7.
6.1 Background

The human face is an important subject in image and video databases because it is a unique feature of human beings and is ubiquitous in photos, news, video, and documentaries. Faces can be used to index and search images and video, classify video scenes, and segment human objects from the background [64]. Face detection is performed to determine if there are any faces in an image and locate the position of each face. There are many problems closely related to face detection as summarized in [69].

- **Face localization** involves determining the spatial position of a single face; this is a simplified problem of face detection with the assumption that there is only one face in an input image [43]. Many researchers, however, use the phrases face detection, face location, and face localization interchangeably as does this work.

- **Facial feature detection** involves detecting the presence and location of facial features, such as eyes, a nose, and eyebrows with the assumption that only one face exists in an image [30].

- **Face recognition or face identification** compares an input image against a database and finds a match [61, 3].

- **Face tracking** involves continually estimating the location and orientation of a face in an image sequence [17].

- **Facial expression recognition** identifies the affective states of humans [16].

The first step of any of the face processing systems described above is to determine the locations in images and video scenes where faces appear. Therefore, research on face detection is critical in image and video database applications. However, face detection from a single image is a challenging task because of variability in scale,
location, orientation (for example, upright or rotated), and pose (for example, frontal or profile).

Different approaches have been developed in recent years for the face detection problem. Some of the most representative works include shape-feature approaches [73, 23]. Neural network approaches are used in [22], and template matching methods are proposed in [43, 7, 37]. These approaches still tend to focus on grayscale images. While they report good performance, they are often computationally expensive. This is especially true of neural network approaches, since they require processing for each possible position and scaling of the image. Another method for detecting faces is to use color information. Skin-color based face detection approaches have several advantages over other methods since under constant lighting conditions, color is almost invariant against changes in size, orientation, and partial occlusion of the face. Moreover, the processing of color information has proven to be much faster than the processing of other facial features which is an important point when dealing with video sequences [64].

At present, almost all visual information is stored in compressed formats, among which the formats defined by the Joint Picture Expert Group (JPEG) and the Motion Picture Expert Group (MPEG) are two of the most typical formats that are used widely on the Internet or in image/video databases. The inverse discrete cosine transform (IDCT) is the last step of the decoder in JPEG and MPEG. Thus, the direct feature extraction from the DCT in JPEG/MPEG streams can discard the necessity of decomposing the image/frame and then exploring its features in the pixel domain. The importance of feature extraction explicitly in the compressed domain is strongly emphasized in [51, 39]. To this end, one of the most efficient face detection algorithms was proposed in [64]. The algorithm starts at the MPEG macroblock (16 x 16) level, a lower resolution version of the video frames, so that the amount of data to be processed can be greatly reduced. Moreover, the algorithm directly uses
chrominance, shape, and DCT frequency information in the compressed domain, and reports 85-90% accuracy for three test sets with the assumption that the orientation of the face is almost up-right.

In this chapter, a color-based system that we describe is developed and implemented for fast detection of human faces in images and video sequences where the faces appear in the compressed domain. First, skin regions are separated from non-skin regions. After that, the human faces are located within the skin regions. The remainder of the proposed method consists of three stages. In the first stage, the algorithm generates chroma charts that show likelihoods of skin and non-skin colors, respectively, using a GMM. These chroma charts are used to generate gray-scale images from the reduced DC-image which is a block-based (8 x 8) sub-sampled image [70]. These images have the property that the gray value at a pixel represents the likelihood of that pixel being skin or not. The gray scale image is segmented into separate skin regions and non-skin regions using the Bayesian decision rule. Then, the DC luminaace component itself is used together with deformable template matching to determine if a given skin region represents a human face or not. In the worst case, face candidates are verified by using the second moments (variances) directly computed from the DCT coefficients in the face region in the final stage. In particular, the moments are manipulated differently according to the orientation of the face. Only I-frames from MPEG streams are analyzed in order to avoid costly decompression of the other frame types.

6.2 Detection of Skin Regions

The first stage of the proposed scheme checks each block of the video frame to see if there exist potential face areas using skin-tone color statistics. Because compressed video is typically stored in YCrCb format, it is desirable to generate skin-tone color statistics in the chrominance (C_b/C_r) plane. The proposed scheme needs a reliable
skin color model that is adaptable to people of different skin colors and to different lighting conditions. In this work, a Gaussian mixture distribution is used to model skin pixels using 315 sample face patches from video frames, and a Bayesian decision rule is employed.

6.2.1 Skin Color Statistics

Chromatic colors have been effectively used to segment color images in many applications [50]. It is also well suited in this case to segment skin regions from non-skin regions in the compressed domain. The color distribution of skin colors of different people was found to be clustered in a small area of the chromatic color space [15, 18]. Although skin colors of different people appear to vary over a wide range, they differ much less in color than in brightness. In other words, skin colors of different people are very close, and differ mainly in their intensities [18]. This finding can be used to develop a skin-color model in the chromatic color space. A total of 315 skin sample patches are used to determine the color distribution of human skin in chromatic color space. As the face skin samples were extracted from color video frames and still images, they were filtered using a low-pass filter to reduce the effect of noise in the samples. Skin-color samples are plotted in \( Y_C G_C \) space in Figure 28(a), and their color histogram is shown in the \( G_C \) plane in Figure 28(b), respectively.

It is noticed that the intensity value \( Y \) has little influence on the distribution in \( G_C \) space and that the color histogram of skin-colors forms simple and small clusters in the \( C_a G_C \) plane. Therefore, a skin-color distribution can be represented efficiently by a GMM, defined as follows. Let the conditional density for a pixel \( X = [C_a, G_C]^T \) belonging to class \( w_i \) be a mixture with \( M \) component densities

\[
p(X|w_i) = \sum_{j=1}^{M} p(X|j) \cdot p(j)
\]

(6.1)

where a mixing parameter \( p(j) \) corresponds to the prior probability that a pixel \( X \) is generated by component \( j \) and \( \sum_{j=1}^{M} p(j) = 1 \). Each mixture component is a Gaussian

72
with mean \( \mu \) and covariance matrix \( \Sigma \) described by

\[
p(X|y) = \frac{1}{2\pi \sqrt{|\Sigma|}} \cdot e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)}.
\]

Expectation maximization (EM) is a well-established maximum likelihood algorithm for fitting a mixture model to a training data set [4]. Often a suitable number may be selected by a user, roughly corresponding to the number of distinct colors appearing in the object to be modeled. In this work, the selected components are \( M = 2 \) and \( M = 3 \) for skin and non-skin tone samples, respectively. A GMM is generated for skin samples as shown in Figure 29. Similarly, non-skin patches are fitted by another GMM. In particular, another 300 patches for the non-skin model were obtained to overcome a drawback of the color-based skin detection scheme from the regions that had a similar skin color but were not actually skin regions.
6.2.2 Classification based on a Bayesian Decision Rule

A Bayesian decision rule for minimum cost is employed, as described in Chapter 4, to assign a color into a skin-like class and a non-skin-like class as follows [64]:

\[ R(1|X) = C_{11} \cdot p(w_1|X) + C_{10} \cdot p(w_0|X), \]  
\[ R(0|X) = C_{01} \cdot p(w_1|X) + C_{00} \cdot p(w_0|X), \]  
\[ R(1|X) \leq_{w_1} R(0|X), \]

where \( w_1 \) and \( w_0 \) denote the two classes, respectively; \( p(w_i|X) \) denotes the a posteriori probability, i.e., the probability of being in class \( i \) given sample \( X \). \( C_0 \) and \( C_1 \) are the cost coefficients of correct classifications; \( C_{01} \) and \( C_{02} \) are the cost coefficients of false classifications. Therefore, \( R(i|X) \) is the "cost" of classifying an unknown sample \( X \) into class \( w_i \).

In this work, the two classes are the skin-like class \( w_1 \) and the non-skin-like class \( w_0 \). Applying the Bayesian Formula

\[ p(w_i|X) = \frac{p(X|w_i) \cdot p(w_i)}{p(X)}, \]
to the above equations leads to the following decision rules,

\[
p(X|w_i) \geq \beta, \quad p(X|w_0) < \beta,
\]

where

\[
\beta = \frac{(C_{10} - C_{11}) \cdot p(w_0)}{(C_{01} - C_{00}) \cdot p(w_1)}.
\]

In the above equations, \( p(w_i) \) is the corresponding a priori probability of class \( w_i \). \( p(X|w_0) \) describes the conditional probability density functions of skin and non-skin colors in the \( C_bC_r \) plane.

The amount of false alarms and false dismissals allowed with the threshold \( \beta \) can be controlled. The relations between false alarm rates and false dismissal rates over different thresholds are illustrated in Figure 30. As the threshold increases, the false alarm rate decreases while the false dismissal rate increases. In order to show the relations between false alarms and false dismissals, first, the number of false alarm regions is counted with threshold \( \beta = 1 \) in a segmented binary image and used as a reference number for relative false alarm rates as the threshold increases. For relative false dismissal rates, the missed face regions are calculated over the total face regions in an original reference image as the threshold increases.

6.2.3 Skin Regions

The skin tone detection approach applies the above minimum cost decision rule to MPEG video streams and classifies each MPEG block as a candidate skin or non-skin one. This work uses only the DCT DC coefficients of the corresponding \( Y, C_b \), and \( C_r \) blocks, which are equivalent to the average values of the blocks in the spatial (pixel) domain. A block is classified as a skin block if its chrominance values fall within the region plotted in Figure 29(a), and its luminance value is within the interval \( 40 \leq Y \leq 240 \) in Figure 28(a). After classifying the block, a binary map image is generated for each I-frame of each video, where a "one" indicates a skin region, and a "zero" indicates a non-skin one. The binary map image is post-processed.
by morphological operations (erosion and dilation consecutively) to eliminate noise, smooth the boundaries from the inside, and fill in holes in the image (see Appendix chapter B). The most important advantage of this simple algorithm is that the skin detection can be implemented with a look-up table (LUT), which makes the algorithm extremely fast. However, it is important to note that the detected regions may not necessarily correspond to skin as in Figure 31(b). It is only reasonable to conclude

Figure 31: An example of a binary map for skin regions; (a) original DC image, (b) segmented binary map for skin, and (c) labelled skin regions.

that the detected regions have the same color as that of skin. The important point is that this process can reliably point out regions that do not have the color of skin and
such regions would not need to be considered further in the face finding process. It is clear from the results above that not all detected skin regions contain faces. Some correspond to hands and arms and other exposed parts of the body, while others correspond to objects with colors similar to those of skin. Hence, the second stage of the face finder will employ facial features to locate the face in all these skin-colored segments.

6.2.4 Region Labelling

Using the above results, the algorithm proceeds to determine which regions could possibly correspond to a human face. To do so, it is necessary to determine the number of skin regions in the image. A skin region is defined as a closed region in the binary map image. The process of determining how many regions are in a binary image involves labelling such regions, where a label is an integer value. This work used a 4-connected neighborhood (i.e., four neighbors of a pixel) in order to determine the labelling of a pixel. If any of the neighbors had a label, the algorithm labels the current pixel with that label (i.e., the current pixel is now included in a segmented region). If not, then the algorithm uses a new label. At the end, the method counts the number of labels and this will be the number of regions in the segmented image as in Figure 51(c). To separate each of the regions, the algorithm scans through each labelled region in turn and generates a new image for each region containing ones in the corresponding labelled positions and zeros otherwise. These new images are rectangular bounded by the sizes of the regions. After this, the proposed scheme iterates through each of the regions found in order to determine if the region might suggest a human face or not.

6.3 Regional Features on a Binary Map Image

Once the system has determined that skin regions exist, it proceeds to analyze some characteristics in the particular regions. In this section, shape constraints are applied
to each region to remove undesired segments that may correspond to skin-tone objects or background. To study each region in the binary map image, the proposed approach first needs to determine its area, center, and orientation.

### 6.3.1 Center of Each Region

The center of each region can be calculated from the following equations [50]:

\[
x_c = \frac{1}{A} \sum_{i=1}^{m} \sum_{j=1}^{n} i \cdot B(i,j),
\]

\[
y_c = \frac{1}{A} \sum_{i=1}^{m} \sum_{j=1}^{n} j \cdot B(i,j),
\]

where \(B\) is the \(m \times n\) matrix representation of the region, and \(A\) is its area obtained by summing all the values in the binary image as

\[
A = \sum_{i=1}^{m} \sum_{j=1}^{n} B(i,j).
\]

### 6.3.2 Orientation and Axes

Most of the faces that researchers have considered [64, 32, 18] are oriented vertically. However, some of the faces have a considerable orientation. In order to have a higher performance, this feature should be considered for face detection. In this work, the shape of a potential face region is approximated by an ellipse such as in Figure 32. The ellipse is defined by [27]

\[
W^T F W = 1,
\]

where

\[
W = [x - x_c, y - y_c]^T
\]

and

\[
F = R^T (d D^{-2}) R = \begin{bmatrix} d & e \\ e & f \end{bmatrix},
\]

\[
R = \begin{bmatrix} a & b \\ c & d \end{bmatrix}
\]

is a rotation matrix.
The rotation and scaled matrices, \( R \) and \( D \), respectively, are defined as follows:

\[
R = \begin{bmatrix}
\cos \theta & \sin \theta \\
-\sin \theta & \cos \theta
\end{bmatrix},
\]

(6.14)

\[
D = \begin{bmatrix}
d_1 & 0 \\
0 & d_2
\end{bmatrix},
\]

(6.15)

where \( \theta \) is the angle of rotation of a region, \( d_1 \) is the major axis length, and \( d_2 \) is the minor axis length of the ellipse. At the end of processing, the orientation and axes of the candidate face region are given by

\[
\theta = \frac{1}{2} \tan^{-1} \left( \frac{2d}{d^2 + 2f} \right),
\]

(6.16)

\[
d_1 = \frac{2\sqrt{2}}{\sqrt{(d + f) - \sqrt{(d - f)^2 + 4e^2}}},
\]

(6.17)

\[
d_2 = \frac{2\sqrt{2}}{\sqrt{(d + f) + \sqrt{(d - f)^2 + 4e^2}}},
\]

(6.18)

The above lengths can be used as the height and width of each region, respectively, to improve the decision process and resize the template face so that it has the same height and width as the region.
6.3.3 Shape Constraints

The constraint of the region size is applied to save computing time. The video frame size is the upper-bound of the face region size, and each region is discarded if its area is below $3 \times 3$ pixels in the DC-image ($24 \times 24$ pixels in the full image) in the proposed system. The size limit is enough to detect the smallest identifiable human faces because it is generally believed that $32 \times 32$ pixels is the lower limit for face detection in the face recognition field [18].

Besides a certain region size, the height to width ratio of the human face can be bounded because the shape of a human face is unique and consistent. The aspect ratio has been found to be in a narrow range between 1.4 and 1.6 [62]. In the proposed method, the region ratio is always greater than or equal to 1 after orientation is taken into account. This value is naturally selected as the minimum value of the region ratio. By observing the results of the experiments with the prototype sets, it was determined that a good upper bound is approximately 1.6 for a human face. However, there are some situations where there is a human face but the ratio is higher than 1.6. This occurs when the person has no shirt or is dressed in such a way that part of the neck and below is uncovered. In order to account for these cases, this work sets the ratio to be 1.6 and removes the region below the corresponding height to this ratio as in Figure 33. However, the ratio of regions over 3.2 (i.e., twice the face

![Figure 33: (a) A potential face region and (b) the region after removal of the lower portion to achieve a height to width ratio of 1.6.](image)

80
upper bound) are not considered since they represent rare cases (even though these cases fall into the previously described category). While the above strategy improves classification, it can cause false alarms in case of long arms which have region ratio between 1.6 and 3.2. This problem is solved in the following section.

6.4 Spatial Domain Face Detection

This section shows how to do the matching between the part of the image corresponding to the skin region and the template face. A human frontal face template is used to make a decision in determining if a skin region represents a face or not. This template was chosen by averaging 20 frontal-view faces of males and females wearing no glasses and having no facial hair. This template was sub-sampled every 8 pixels in both directions to be consistent with the region in a luminance DC-image, and was low-pass filtered. The sub-sampled template is shown in Figure 34. The template is

![Figure 34: the sub-sampled face template.](image)

also vertically and horizontally centered around the tip of the nose of the model. At this point, the proposed face detection algorithm has all the required parameters to do the matching between the part of the image corresponding to the skin region and the human face template. Human faces (frontal faces and side faces) with orientation are detected using the template matching in the spatial domain and moments in the compressed domain. For the binary map image corresponding to a skin region, all holes, if any, in the region are filled, and the region is multiplied by the luminance DC-image corresponding to the face region. An example is shown in Figure 35. In the figure, the template face is resized, rotated, and positioned to the same coordinates as the skin region.
Figure 35: Template matching procedure
Specifically, the front face model is resized according to the height and width of the region computed, and rotated according to the target orientation calculated in Section 6.3. Now, the template face is aligned to the same direction as the skin region (d). Then, the algorithm selects the same size boundaries (e) from both the target image (c) and the model image (d) to reduce the computation. Finally, the cross-correlation value is computed between the part of the image corresponding to the skin region and the template face, and the face region is indicated by the rectangular box (f). It was empirically determined from experiments with prototype sets that a good threshold value of the cross-correlation for classifying a region as a frontal face is 0.8, and a good value for ignoring a potential region is 0.2.

It is important to note that the proposed algorithm needs to determine two threshold values (lower and upper values) because the purpose of the proposed system is to find all the faces whether they are front or side views, but only the frontal template is used for the template matching process, and this matching process gives corresponding outputs only to frontal faces. Therefore, the lower value is used to discard a potential face region and the upper value is used to classify the region as a face in the template matching process. However, the regions that score between the lower and the upper values go to the next stage to verify if they are face regions in the compressed domain.

6.5 Compressed Domain Face Detection

The main purpose of this stage is to verify the human faces based on the previous template matching results. One of the most important characteristics of the proposed approach is that the method uses moments directly computed from the DCT coefficients of the Y-component to determine the face regions according to their orientation, and these moments are manipulated by the orientation of a potential face region.
6.5.1 Calculating Moments

This work will use the relationship between the pixels' DCT coefficients for verifying human faces. The two-dimensional DCT in MPEG is

$$\text{DCT}(u, v) = \frac{1}{4} C_u C_v \sum_{i=0}^{7} \sum_{j=0}^{7} \cos \left( \frac{(2i + 1)\pi}{16} \right) \cos \left( \frac{(2j + 1)\pi}{16} \right) f(i, j),$$  \hspace{1cm} (6.19)

where

$$C_\eta = \begin{cases} \frac{\sqrt{2}}{2}, & \text{for } \eta = 0, \\ 1, & \text{otherwise}. \end{cases}$$ \hspace{1cm} (6.20)

The DC coefficient represents the average of a single block in the original image, and the AC coefficients represent variations in gray values in certain directions at certain rates. Figure 36 shows the DCT coefficients and the zigzag ordering for encoding.

![Figure 36: Zigzag ordering of DCT coefficients.](image)

In order to consider the properties of AC coefficients, for example, DCT(0, 1) is

$$\text{DCT}(0, 1) = \frac{1}{4} C_0 C_1 \sum_{i=0}^{7} \sum_{j=0}^{7} \cos \left( \frac{(2j + 1)\pi}{16} \right) f(i, j),$$  \hspace{1cm} (6.21)

which can be expressed as

$$\text{DCT}(0, 1) = \frac{16}{4} \left\{ \cos \frac{\pi}{8} \left( \sum_{i=0}^{7} f(i, 0) - \sum_{i=0}^{7} f(i, 7) \right) \\ + \cos \frac{\pi}{8} \left( \sum_{i=0}^{7} f(i, 1) - \sum_{i=0}^{7} f(i, 6) \right) \\ + \cos \frac{\pi}{8} \left( \sum_{i=0}^{7} f(i, 2) - \sum_{i=0}^{7} f(i, 5) \right) \\ + \cos \frac{\pi}{8} \left( \sum_{i=0}^{7} f(i, 3) - \sum_{i=0}^{7} f(i, 4) \right) \right\}.$$ \hspace{1cm} (6.22)
The above equation indicates that DCT(0,0) essentially depends on intensity differences in the horizontal direction, therefore it is sensitive to the vertical edges. Similarly, DCT(u,v) depends on the vertical intensity differences.

Direct computation of moments in the DCT domain was proposed in [12]. The first moment (mean) and the second moment (variance) are defined, respectively, in a luminance block with size \(2 \times 8\) as

\[
\mu_1 = \frac{1}{8} \cdot \text{DCT}(0,0),
\]

\[
\mu_2 = \frac{1}{64} \sum_{u=0}^{7} \sum_{v=0}^{7} \text{DCT}(u,v)^2, \text{ for } u \neq 0 \text{ and } v \neq 0,
\]

where DCT(u,v) is the DCT coefficient at (u,v) in the block. The concise expressions (6.23) and (6.24) show that the mean is directly derived from the DCT coefficient, while the variance is just the average of squared DCT coefficients. In order to use these moments as a measure for detecting a face region, it is needed to partition the DCT coefficients corresponding to the directional features [64, 28]. There are more discontinuities of intensity levels in the vertical direction compared to the horizontal direction of the image in face regions because of the existence of the eyes, nose, mouth junction, and the lips in the face regions. These discontinuities are indicated by the DCT coefficients in the frequency band. The grouping of the DCT coefficients, where groups H, V, and D correspond to vertical (i.e., horizontal discontinuities), horizontal, and diagonal edges, respectively, is shown in Figure 37. In these masks, a "one" indicates the value of the DCT coefficient at that position, and a "zero" means

![Figure 37: The partitioning of the DCT coefficients.](image)
the DCT coefficient is not used. Given a potential face region, its rectangular bounding box $B$ of size $m \times n$, and its size $A$ in the binary image, the second moments based on the direction features are defined at the corresponding DCT blocks in the luminance image as

$$\mu_{11} = \frac{1}{64} \cdot \frac{1}{A} \cdot \sum_{i,j=0}^{m-1} \sum_{u,v=0}^{n-1} B(i,j) \cdot \text{DCT}_{u,v}(u,v)^2 \cdot H(u,v),$$  

(6.25)

$$\mu_{12} = \frac{1}{64} \cdot \frac{1}{A} \cdot \sum_{i,j=0}^{m-1} \sum_{u,v=0}^{n-1} B(i,j) \cdot \text{DCT}_{u,v}(u,v)^2 \cdot V(u,v),$$  

(6.26)

where $\text{DCT}_{u,v}(u,v)$ is the set of DCT coefficients of the $8 \times 8$ block from the original image corresponding to the pixel $(i,j)$ of the potential face region. The ratio of the vertical moment to the horizontal moment can be used to detect the region that has more discontinuities in the vertical direction. However, these masks can be used with the assumption that the potential face regions are vertically positioned, or only tilted a little. As a simple example, an almost upright face and a $-56^\circ$ tilted face are shown in Figure 38. Theoretically, the ratio $\mu_{12}/\mu_{11}$ of the left image is greater than 1, but that of the right image is less than 1. Therefore, it is necessary to manipulate the above masks according to the orientation of potential face regions. Intuitively, a simple way to solve the above problem is to rotate the tilted region by a negative angle ($-\theta$) in the original image in the spatial domain, to perform the DCT transformation on it, and to get the ratio of moments to determine if it is a human face. This simple approach requires a lot of processing time. To make the proposed algorithm faster, a more efficient approach without going back to the spatial domain is necessary.
6.5.2 Selecting Coefficients

In order to select the AC coefficients directly in the DCT domain according to the orientation of a potential face, the DCT coefficients are separated into four regions as shown in Figure 39. In the figure, only horizontal edges (vertical discontinuities)

are considered to explain the procedure of selecting AC coefficients. As a horizontal edge ($\theta = 0$) rotates to become a vertical edge, the corresponding features move consecutively from regions (a) to (d). Similarly, this observation can be applied to vertical edges. It is expected that a tilted human face can be compensated and dealt with as an upright face by selecting appropriate AC coefficients according to the orientation of the face.

For verifying a human face in the DCT domain, the required number of DCT coefficients is equal to the original image size, which is 64 times bigger than the DC-image size that the proposed approach deals with. However, to reduce the number of computations and save the decoding time for each coefficient, this work uses only eight AC coefficients in a block, which are illustrated as solid and striped boxes in Figure 39. In [51], edge information in the DCT domain is successfully extracted by using five AC coefficients of each block in zigzag order and is used to identify shot change points. In particular, the proposed method uses two sets of masks to compute
the second moments based on the angle of a potential face as illustrated in Figure 40.

\[
\begin{bmatrix}
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
H_x \\
V_y \\
H_x \\
V_y
\end{bmatrix}
\]

Figure 40: Two pairs of partitioning AC coefficients.

Each pair of masks is selected according to its orientation of a potential face by

\[
[H_x, V_y]^T = \begin{cases} \xi = 1, & \text{if } 0 \leq |\theta| < \theta_1, \\
\xi = 2, & \text{if } \theta_1 \leq |\theta| < \pi/4. \end{cases} \quad (6.27)
\]

When the magnitude of an angle $|\theta|$ is larger than $\pi/4$, the angle is updated by

\[\theta = \pi/2 - |\theta|,\]

and Equation (6.27) is applied again. Then the horizontal mask $H_x$ and the vertical mask $V_y$ are exchanged based on the symmetry property shown in Figure 41. The solid lines are diagonally symmetric edges in the spatial domain.

\[\begin{bmatrix}
\text{DCT}(u, v) \\
\text{DCT}(v, u)
\end{bmatrix}\]

Figure 41: The AC coefficient relationships between symmetry edges.

manipulated second moments are defined by

\[
\begin{align*}
\mu_{2H_x} &= \frac{1}{2\xi} \cdot \frac{1}{A} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} B(i,j) \cdot (\text{DCT}_2(u,v))^2 \cdot H_x(u,v), \\
\mu_{2V_y} &= \frac{1}{2\xi} \cdot \frac{1}{A} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} B(i,j) \cdot (\text{DCT}_2(u,v))^2 \cdot V_y(u,v).
\end{align*}
\]  

(6.28)  

(6.29)
The only difference between Equation (6.25) and Equation (6.26) is in the masks used. However, these operations can reduce the computational complexity by almost 3 or 6 times as illustrated in Table 8. Moreover, the selected masks can save partial decoding time since several numbers of the corresponding AC coefficients are necessary instead of all the AC coefficients. In order to get an AC coefficient in the DCT domain, partial decoding procedures including Huffman decoding and inverse quantization are required. Therefore, it is preferable to use a small number of AC coefficients to reduce computations for face verification. Some test results of moments using different mask sets are shown in Figure 42. In the figure, the solid line indicates the ratio of the second moments using the proposed masks in Figure 40 according to the orientations of potential faces. From the result, each set of masks can be used to

Table 8: Computation complexities according to masks.

<table>
<thead>
<tr>
<th>Moments</th>
<th>Add</th>
<th>Multiply</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>h_{2,2}</td>
<td>MxN(2x2)</td>
<td>N(N+1)x3</td>
<td>N(N+1)x4</td>
</tr>
<tr>
<td>h_{2,3}</td>
<td>MxN(2x3)</td>
<td>N(N+1)x3</td>
<td>N(N+1)x4</td>
</tr>
</tbody>
</table>

Figure 42: The ratio of moments with different mask sets.
verify a face. However, the proposed method is more stable than any other method used. This can be seen in Figure 42, where the range of ratio of moments is smallest for the proposed method. Additionally, the moment ratio is used as the boundaries to determine if the potential region is a face or not.

6.6 Evaluation

To evaluate the performance of the proposed face detection system, this research focuses on the overall speed of the system and the accuracy of the results. The algorithm executes in real-time for locating the human faces. In particular, to show that the proposed algorithm performs competitively against existing algorithms, the face detection algorithms developed by Rowley et al. [23] at Carnegie Mellon University and Wang et al. [64] were selected for comparison in processing accuracy and speed for detecting faces.

6.6.1 Experimental Setup

The proposed algorithm has been evaluated using the first test set in Table 9. This test data set contains 50 images that have been extracted as key frames from the 1-frames of various MPEG videos. The size of each frame is 352 x 240 pixels. The sequences consist of a CF, news, and movies. This set of 50 images contains 99 faces, which are again classified into two categories according to their orientations, and 5 images which do not contain faces. They cover most of the cases that the algorithm

<table>
<thead>
<tr>
<th>Type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontal</td>
<td>28</td>
</tr>
<tr>
<td>Semifrontal</td>
<td>79</td>
</tr>
<tr>
<td>Side</td>
<td>32</td>
</tr>
<tr>
<td>Little tilted (±15°)</td>
<td>87</td>
</tr>
<tr>
<td>Much tilted (±90°)</td>
<td>12</td>
</tr>
<tr>
<td>Total faces</td>
<td>99</td>
</tr>
</tbody>
</table>

90
has to deal with. In Table 9, a detailed description of the contents of this set is
given according to facial characteristics. In order to evaluate the proposed human
face detection algorithm using a template and moments in the compressed domain,
the algorithm was implemented in C++, and the experiments were performed using
a Pentium II 400 MHz PC, under the Windows operating system.

6.6.2 Experimental Results

In Figure 43, some results of the proposed face detection scheme are presented for
12 images of the test data set. These examples include color images with multiple
faces of different sizes, different colors, different positions, and a frame which does
not contain any faces. Detected face regions are marked by black rectangular outlines
overlaid on the original video frames even though the proposed system performs on
the DC-image frames (an example is shown in Figure 44). The proposed system
has been compared with two well-known face detectors with respect to processing
accuracy and speed. These two systems are summarised as follows:

- The CMU face detector [23] has two stages. First, a set of neural network-
  based filters are applied to an image and examine each location in the image at
scales, looking for locations that might contain a face. Then a network
  arbitrator merges the detections from each filter and eliminates overlapping
  detections. It was designed to detect faces with a minimum size of 20 \times 20
  pixels. This system requires a lot of time to detect faces even though a fast
  implementation was introduced in [23].
Figure 43: Some results of the proposed face detection algorithm.

(a) Original video frame  (b) DC-image (enlarged by 8 × 8)

Figure 44: A frame and its DC-image.
Wang's face detector [64] has three stages. In the first stage, skin color classification is performed at the MPEG macro-block level directly in the chrominance plane \(C_{bC_{c}}\) without taking the intensity value \(Y\) into account. A binary rectangular template is used to match with the segmented outputs of the first stage. Then it determines if the selected potential regions have faces or not using the ratio of vertical to horizontal energy distributions. The minimum face size this system can detect is \(3 \times 3\) macro-blocks (48 \(	imes\) 48 pixels). The average run times of this algorithm on a SPARC 5 workstation and an SGI Indigo 2 workstation were reported as 32.6 ms and 15.6 ms, respectively, for 100 I-frames of size 352 \(	imes\) 240.

The first test set of 50 images was tested using the interactive demonstration of the CMU face detector at http://www.vasc.ri.cmu.edu/cgi-bin/demos/fndface.cgi that allows users to submit an image for processing in batch mode and to retrieve the resulting image with bounding boxes overlaid on the detected faces. The results of the proposed algorithm compared to the CMU face detector and Wang's face detector on the test set are illustrated in Table 10. Some comparative examples of the results are presented in Figure 45. Each column maps to the output of the CMU detector, Wang's face detector, and the proposed scheme from left to right, respectively.

<table>
<thead>
<tr>
<th>Results</th>
<th>The proposed method</th>
<th>CMU face detector</th>
<th>Wang's face detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>94 (94.95%)</td>
<td>92 (92.93%)</td>
<td>87 (87.88%)</td>
</tr>
<tr>
<td>False alarms</td>
<td>7 (7.05%)</td>
<td>12 (12.12%)</td>
<td>5 (5.05%)</td>
</tr>
<tr>
<td>False dismissals</td>
<td>5 (5.05%)</td>
<td>7 (7.07%)</td>
<td>12 (12.12%)</td>
</tr>
</tbody>
</table>

It is important to note that the minimum sizes for the compared face detectors are 20 \(\times\) 20, 48 \(\times\) 48, and 24 \(\times\) 24, respectively, for the CMU, Wang's, and the proposed systems. In the first and second rows of Figure 45, the CMU system and the proposed scheme found several small faces, whereas Wang's face detector could not find them.
A main difference between the compared systems is whether color information is used or not. The CMU face detector does not use color information. Only the geometrical intensity features of a face are used in the pixel domain. Therefore, a set of candidate face areas cannot be built, and the neural network filters have to be applied at every pixel location in each image of the multiscale pyramid. The CMU scheme is more time consuming than Wang's and the proposed scheme. Moreover, the CMU face detector may find false alarms in complex backgrounds that have non-skin areas as illustrated in Figures 45(a) and (d). The proposed system does not have this drawback. On the other hand, as seen in Figure 45(g), Wang's scheme and the proposed scheme fail in the case of extreme lighting conditions causing bad skin color segmentation.

As shown in Table 10, the proposed algorithm detects 94 of the 99 faces, which means a successful detection rate of 94.95%, whereas the CMU system and Wang's system detect 92 faces and 87 faces, respectively, leading to a successful detection rate of 92.63% and 87.88%. It was observed that the CMU face detector and Wang's face detector are more sensitive to the orientation of a face than the proposed method, especially for faces with a large amount of tilt. Examples can be seen in the fourth row of Figure 45. The main reason is that Wang's system does not take the orientation of the face into account, assuming that faces are upright or have little tilt, and the CMU face detector does not consider this situation during the stage of neural network filters. Fewer false alarms are obtained using Wang's face detector; however, Wang's scheme provides many false dismissals (12.12%) mainly in small faces and faces with a large amount of tilt. Seven false alarms and five false dismissals (5.06%) are obtained with the proposed method, whereas 12 false alarms and seven false dismissals (7.07%) are obtained using the CMU face detector. After the prominence segmentation step, a number of potential false alarms appear, but the proposed shape constraints reduce them in a very efficient way. In particular, the selective moment calculations using intensity facial features directly in the compressed domain is effectively used when
the result of template matching is in predefined range.

For an overall system speed comparison, a fast system by Wang et al. [64] is implemented as a baseline system. The proposed face detection system used 100 frames of a news sequence, whose frame size was 352 x 240, as the second test set. The main differences between Wang’s system and the proposed system are the processing image size and the mask size of DCT coefficients. Wang’s system is based on the macro-block level image size whereas the proposed system is based on the block level image size. To verify a face in the compressed domain, Wang’s detector needs to decode all the coefficients, but the proposed scheme selects and decodes only several coefficients if necessary. The proposed method decodes some DCT coefficients selectively according to the results of the template matching. Even if Wang’s system is much faster in the skin-tone segmentation stage, it requires a large number of computations to verify a face in the DCT domain. The elapsed time of the two systems depends on the output of the skin-tone detection stage. After segmentation, if only a few potential regions are shown in the binary image, then little template matching is involved, so less time is needed. On the other hand, if the video sequence contains many skin color regions and is complex, then the template matching step will take longer.

The overall processing time comparison is shown in Table 11 for the test sequence.

The average run time of Wang’s system (31.3 msec.) is approximately 1.53 times faster than the proposed system (47.9 msec.). It is worth noting that Wang’s face detector has many false dismissals, but the proposed method detects them because of the minimum face size limitation. In this case, Wang’s detector does not have to go
to the next verifying stage. When GOPs in MPEG format is taken into account, the
GOP has one I-frame and consists of almost 15 frames. It is enough to provide real-
time processing only if a system performs an algorithm during decoding two GOPs
per second. Consequently, the proposed system is quite fast and performs in real-time
under the described experimental environment.

6.6.3 Limitations and Discussions
The algorithm can be applied to compressed color images and videos because the
color information is used to separate skin regions from the background. Considering
the quality of detection, the proposed algorithm does not give the exact outlines of
the faces as shown in Figure 45, since the algorithm works on DC-images to avoid
having to apply the IDCT. Sometimes, the positions of the faces detected are not
perfectly aligned because the face rectangles this system detects lie on the borders
of the 8 × 8 blocks. This situation can be improved if the algorithm uses more
AC coefficients (such as DC+2AC coefficients) which still avoids the full decoding
procedure. The proposed face detection system has been designed to be independent
of lighting conditions, but very poor lighting conditions still affect the number of
false alarms and false dismissals. From the experimental results, it is observed that
the CMU face detector could be applied under any set of conditions at the cost of
increasing the number of false alarms and increasing the processing time, and Wang's
detector is preferred only if large size faces (such as an anchor person scene of a news
sequence) are considered.

Despite some restrictions, the proposed algorithm reports good results with re-
spect to accuracy and speed and can be used as a useful tool for video indexing, re-
trieval, and abstraction. Additionally, although the algorithm uses a simple Bayesian
approach to detect skin color, any reliable technique (such as the Watershed trans-
form [33]) can be integrated into this system.

96
6.7 Summary and Remarks

This work proposes a simple and fast processing algorithm for detecting human faces in the compressed domain. The algorithm has main three steps. First, a DC-image is extracted from an I-frame of a compressed video sequence using the DCT DC coefficients, and potential face regions are separated from non-skin regions using color information. Then, the location of a face is detected using a deformable template obtained from averaging real frontal human faces. If the result of template matching is enough to determine whether the potential region is a face or not, the algorithm stops. Otherwise, it goes to the final verification step using the manipulated DCT coefficients according to the orientation of faces.

Experimental results show a high detection rate (94.95%) regardless of the number, size, and orientation of the faces. The algorithm executes in real-time. It is believed that the proposed system can be used as a significant tool for video analysis.
Figure 45: Some comparative results from the test set. The first column corresponds to the output of the CMU face detector; the second column corresponds to the output of Wang’s system; and the third column corresponds to the output of the proposed algorithm.
CHAPTER 7

VIDEO ABSTRACTION III: ARCHITECTURE

This work has proposed a novel system that can automatically create an optimal and nonrepetitive summarization and support different user requirements for video browsing and content overview by outputting both the optimal set of key frames and a summarized version of the original video with the user-specified time length.

This chapter describes the last building block of the proposed video abstraction system. In Section 7.1, the proposed structure for video abstraction is presented. Section 7.2 illustrates how to make a video abstraction with user interaction. To evaluate the overall abstraction system, this work is compared with another algorithm based on the speed of summarizing a video, and some restrictions are discussed in Section 7.4. The procedures of summarizing and browsing contents of a video is illustrated in Figure 46. The solid arrow indicates the direction of abstraction, while the dotted arrow indicates the direction of browsing contents of a video.

7.1 Multi-Resolution Summaries

This section proposes a content-based multi-resolution structure for a large-scale video database that provides users with the ability to browse and find desired video content, either on a local workstation or through the Internet with very limited bandwidth. The video abstraction architecture is shown in Figure 47 and consists of three categories: media, user, and database. The media component includes the video sequences, which are manually classified according to their genre. The user component indicates the activity of users searching and selecting relevant video on a local workstation or through the Internet. In the database component, it is assumed that
the video key frames are clustered and have priorities assigned to them according to different characteristics such as length of a shot, the number of key frames of the shot, human presence, and camera motions. Let \( \Omega \) be a set of the refined key frames in a video arranged with temporal order, and \( \Omega_k = \{f_1, \cdots, f_p\} \) be a set of \( p \) key frames. \( L \) is the number of summary levels, and \( f_q \), which is the temporal position of the original video, is the \( q^{th} \) key frame. At any level, the summarization can be represented as

\[
\Omega_k = \{f_1, \cdots, f_p\}, \text{ where } 1 \leq k \leq L \text{ and } 1 \leq q \leq p. \tag{7.1}
\]

The following equation defines the number of key frames to be selected at level \( k \):

\[
N_k = \lfloor p \times \frac{1}{L} \rfloor, \tag{7.2}
\]

where \( 1 < k \leq L \), and \( \lfloor \cdot \rfloor \) is the rounding operator. For the first level, this system selects the most static key frame to show users the representative image at the beginning of their video browsing. For level \( k \), the system picks \( N_k \) key frames according
to their content activities but produces a summarization according to their temporal orders. Finally, level $L$ must have all the key frames, i.e. $\Omega_L = \Omega$. The system also provides a summarized video clip, which can be available at each level, for the user's convenience. The short video clip can be understood as a temporal extension of the key frames at each representation level. Each shot or subshot, to which at least one extracted key frame belongs, is taken as a key video segment. These key segments are then concatenated to form the short video clip. It is preferable to use entire shots or subshots for making a clip due to their complete contexts. Such completeness makes the short video clip understandable. An alternative to this is to use only extracted key frames, and assign an equal or different time length depending on the duration of their shot or subshot in an original video. However, the probability of having a complete context is considerably lower in this case even though it is simple way. The following section describes the procedure to select the representative images at each level from user interactions.

Figure 47: Overall structure of the abstraction system.
7.2 User Interaction

In order to show how users (clients) interact with the database server to find their intended information, the system uses the result clustered key frames in Figure 43 for a CF sequence. Let $K(m, n)$ be the temporal position of the refined $n^{th}$ key frame at the $m^{th}$ shot. The characteristic of this sequence has two subshots of a zoom-in and a zoom-out camera operation, sequentially, at the third shot $K(2, 0)$. It is assumed that the characteristics of the key frames are given as shown in Table 12. In practice, the characteristics are assigned numerical values from the proposed system.

If a client wants to have the information according to the degree of no-motion and human content in a video (in this example, the human content is weighted more heavily than the camera motion content), and chooses three levels ($L = 3$), the first level ($k = 1$) is $\Omega_1 = [f_6]$, where $f_6$ and $f_5$ were candidates but $f_6$ was chosen because it
cane first temporarily. Similarly, the rest of the levels are $\Omega_2 = [f_1, f_2, f_3, f_4, f_5, f_6]$ and $\Omega_3 = [f_1, \ldots, f_6]$ from Equation (7.2). The summary example is shown in Table 12: Characteristics of the refined key frames.

<table>
<thead>
<tr>
<th>Frame</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
<th>$f_6$</th>
<th>$f_7$</th>
<th>$f_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>Low</td>
<td>No</td>
<td>No</td>
<td>Low</td>
<td>High</td>
<td>No</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Motion</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Figure 49. The elements at each level may be changed based on priorities emphasized more or less by the client.

This architecture enables users to browse and find relevant data efficiently within the limited bandwidth, even though users go through several rounds of interactions with the system in their searching process for desired information in the worst case using the key-frame sets. In practice, the database (a server) only needs the key-frame set $\Omega$ because any level of summarisation can be performed at the user's request and be displayed on the user's displaying device.

### 7.3 Interactive System Interface

The proposed system also provides an interactive interface to correct the results of the algorithms described in previous chapters as an option, even though the default operation of the system is fully automatic.

Errors produced by any one of the automatic algorithms will propagate, resulting in event boundaries that are incorrect. Therefore, it is important to provide interfaces so that a human operator (user) can verify and correct the results produced automatically at the video analysis and clustering steps.

The proposed system does not use predefined video structures and domain knowledge, such as the broadcasting program and its story structure models to facilitate story segmentation. Hence, the proposed approach is not domain or program specific.
Figure 49: Video summary example.
The context produced from the analysis and clustering algorithm allows scalable summary generation. Thus, the video summary can be tailored according to the client's profile and needs.

Now the tools provided to the system user is the server side are described to modify the results of automatic story segmentation at both the shot and event levels. The steps in the interactive generation of the final organized video can be summarized as follows:

1. Automatic construction of a shotlist, which contains the information of (sub)shot lists, from the video analysis, clustering, and human face detection algorithms on the compressed video.

2. Viewing and editing the (sub)shot change structure and clustering results to add new (sub)shots or merge (sub)shots.

3. Automatic generation of multimedia video summary, where the generated video clip is used as a shot clip at the highest level of the multi-resolution summary, while the generated key frames are used as an input for clients' browsing to find their relevant information.

The second step in the list above requires user (human operator: server side) interaction, and therefore, an interface needs to be provided with the required functionality. Since these interfaces work with higher level representations of the video, a separate component is also provided to view the raw video. The proposed system provides three main interfaces to the user (server side), which communicate with each other so that changes made using one component produce the appropriate updates in the other interfaces.

Analyzer: This interface is used to analyze video contents, to show the intermediate graphs, and to produce shotlists. The video stream is represented as key frames containing all information such as duration of (sub)shot and human content.
Video player: This interface is used for playing the video from any point in the video. It has the functionality of a VCR including fast forward, rewind, pause, and step.

Listview: This interface is used to view and alter the automatic grouping of (sub)shots based on visual similarity. Once the similarity-based clustering results have been finalized, they are used as an interface to correct event (or story) boundaries. It also provides extra information, which includes an icon of each shot in temporal order; the length of the shot; all of the key frames of the shot; and motion, human, and cluster information. A screen-capture illustrating the Listview interface is shown in Figure 50. The user is allowed full freedom in changing the

![Diagram of Listview interface]

Figure 50: A screen-capture illustrating the Listview interface after sorting of the priority column in descending order.

event boundaries and other information, since semantic information can often
be missed or misinterpreted by automatic processing. The following operations are provided to facilitate changes on an item of the list:

- **Sort items:** Each row lists all of the information extracted from the algorithm described in the previous chapters. Rows can be sorted by clicking a column to sort.

- **Add/delete rows:** New rows can be added below the destination row. Alternatively, unnecessary rows can be removed by the user.

- **Updates:** When changes are made in the order of the (sub)shots using *ListView* and the user wants to see these changes reflected in the Analyzer, (s)he can opt to send a signal to the Analyzer to reload the rearranged *shotlist* after saving it from *ListView*. The Video player can be called from this interface exactly as in the Analyzer interface.

- **Change priorities:** Users can change the value of each priority as is necessary for their own application. After changing the values of priorities and sorting a priority column in the *ListView*, they can overview or make a short clip with desired contents. For example, a short clip can be made only with two significant shots based on length of a shot and human appearance. Priorities provided in the proposed system are temporal order of a shot, length of a shot, number of key frames, camera motion, human presence, and size of cluster.

The user (server side) can invoke these operations to regroup the stories into more meaningful stories and sub-stories. The order of stories can also be changed from their usual temporal order to a more logical sequence. Though the primary function of the *ListView* interface is to interact with the high-level structure, the same interface can also be used to view and modify the groups generated by the automatic clustering process.
7.4 Evaluation

To evaluate the performance of the proposed automatic video abstraction system, the compactness and the overall speed of the resulting summaries are focused on. For evaluating the overall abstraction system, an algorithm proposed by Huang [68] is used as a baseline system for example.

7.4.1 Experimental Setup

To show that the proposed system can be applied to any type of video sequence, four different compressed MPEG-1 sequences are considered. These sequences include a movie sequence, a music video, a documentary, and a CF sequence. All algorithms were implemented in C++ and run on a Windows system with a 400 MHz Pentium-II processor.

7.4.2 Experimental Results

Huang et al. [68] proposed extracting the key frames using an unsupervised clustering scheme. Basically, all video frames within a shot \( s = \{ f_{r1}, f_{r2}, \ldots, f_{rN} \} \), which are segmented by a shot change detection algorithm [5], are first clustered into \( M \) number of clusters based on the \( 16 \times 8 \) 2-D HSV color histogram similarity comparison in HSV color space, where a predefined threshold \( \rho \) controls the density of each cluster. A similarity measure is defined as

\[
\sum_{h} \sum_{s} \min(H(h, s), H(h, s)).
\]  

(7.3)

Next, all the clusters that are large enough are considered to be the key clusters, and a representative frame closest to the cluster centroid is extracted from each of them. To implement their system, a color histogram with DC-images is used to speed up the processing time, and the threshold \( \rho \) is set to 0.9. Huang’s unsupervised clustering algorithm for video summary can be summarized as follows:

108
1. Initialization: \( f_{r_1} \rightarrow \sigma_1 \), \( f_{r_2} \rightarrow \) the centroid of \( \sigma_1 \) (denoted as \( \epsilon_{r_1} \)), \( i \rightarrow \) numberCluster;

2. Get the next frame \( f_{r_i} \). If the frame pool is empty, go to 6;

3. Calculate the similarities between \( f_{r_i} \) and existing clusters \( \sigma_k \) (\( k = 1, 2, \ldots, \) numCluster): \( \text{sim}(f_{r_i}, \sigma_k) \), based on Equation (7.3);

4. Determine which cluster is closest to \( f_{r_i} \) by calculating Maxsim. Let

\[
\text{Maxsim} = \max \left\{ \text{sim}(f_{r_i}, \sigma_k) \mid k \in \text{numCluster} \right\}.
\]  
(7.4)

If \( \text{Maxsim} < \sigma \), it means that \( f_{r_i} \) is not close enough to be put in any of the clusters, go to 5; otherwise, put \( f_{r_i} \) into the cluster which has Maxsim and go to 6.

5. \( \text{numCluster} = \text{numCluster} + 1 \). A new cluster is formed: \( f_{r_i} \rightarrow \sigma_{\text{numCluster}} \).

6. Adjust the cluster centroid: Suppose cluster \( \sigma_k \)'s old centroid is \( \epsilon_{r_k} \), \( D \) is the number of frames in it, and the new centroid is \( \epsilon_{r_k} \), then \( \epsilon_{r_k} = D/(D+1)\epsilon_{r_k} + 1/(D+1)f_{r_k} \). Go to 2.

After the clusters are formed, the next step is to select key frame(s). Only those clusters which are big enough are considered as key clusters (their sizes are bigger than the average size of the clusters, \( N/M \)), and a representative frame is extracted from this cluster as the key frame. For each key cluster, the frame which is closest to the cluster centroid is selected as the key frame. All the selected key frames are concatenated to summarize a video sequence.

The overall processing time comparison is shown in Table 13 for four different video sequences. The average speed of the proposed system is almost 9 times faster than real-time and almost 19 times faster than that of the baseline system implemented. The proposed system and the baseline system depend on the results of s
Table 13: Overall speed comparison between the proposed system and Huang's system.

<table>
<thead>
<tr>
<th>Test sequences</th>
<th>Time length</th>
<th>Proposed system (ratio)</th>
<th>Huang's system (ratio)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
<td>30 min.</td>
<td>3.38 min (8.88)</td>
<td>66.03 min (0.45)</td>
</tr>
<tr>
<td>Music video</td>
<td>4 min.</td>
<td>0.42 min (9.52)</td>
<td>8.92 min (0.48)</td>
</tr>
<tr>
<td>Documentary</td>
<td>10 min.</td>
<td>1.09 min (9.17)</td>
<td>20.96 min (0.48)</td>
</tr>
<tr>
<td>Silent</td>
<td>15 sec.</td>
<td>1.56 sec (9.62)</td>
<td>32.96 sec (0.46)</td>
</tr>
<tr>
<td>Average</td>
<td>11.06 min</td>
<td>1.22 min (8.59)</td>
<td>24.12 min (0.46)</td>
</tr>
</tbody>
</table>

shot segmentation algorithm before clustering, but the baseline system can compensate for this algorithm during key-frame extraction based on visual content similarities within a shot, where two different shots may be concatenated unintentionally by the segmentation algorithm. However, the baseline system is less effective with respect to the compactness of video summaries since it does not merge visually similar shots that are separated by other shots, such as the shot changes of dialogue shots where the camera switches from speaker to speaker.

7.5 Restrictions and Discussions

The proposed abstraction system is restrictive in several aspects. It can only be applied to MPEG compressed sequences, because it uses some MPEG features in the compressed domain for (sub)shot change detection. Since the algorithm for identifying shot transitions is based on the bi-directionally predicted macroblock numbers in B-frames, normal GOP structures such as “IBBPBB⋅⋅⋅BB” are preferred. This algorithm does not work well on MPEGs with “IPP⋅⋅⋅IP” GOP structures where only the anchor frames are processed. The algorithm is relatively independent of the quality of MPEG sequences, but very poor quality sequences still cause false alarms or dismissals. False alarms may be recovered by the refinement process, which performs a histogram and frame difference with DC-images at the nearest two I- or P-frames to each shot change point. However, dismissals from very low quality video cannot be completely avoided.

116
In the classification of camera movements, this method has difficulty in detecting some movements when a camera moves diagonally (panning or tilting diagonally) and zooms a region away from image center. However, this kind of problem can be overcome by using more templates at the cost of processing time.

Despite its restrictions, the proposed compressed-domain approach is efficient and can be applied to large video databases for video summary. Additionally, although the proposed system uses the compressed features in the compressed domain as the shot detection and camera movement analysis measures, any reliable technique can be integrated into the system. For example, a color histogram difference with DC-images can be used when a sequence is of very low quality.

7.6 Summary and Remarks

This chapter proposes an abstraction structure using user interaction as the last step for video abstraction. For an overall system speed comparison, another system is implemented and compared with the proposed system based on speed. Performance results show that the proposed system is almost 9 times faster than real-time (playing time) and almost 19 times faster than the compared system.
CHAPTER 8

CONCLUSION

The focus of this research has primarily been on the speed and the compactness of compressed video abstraction. The application of several familiar processes to MPEG videos in the compressed domain has been investigated and developed. These processes developed include shot change detection, camera motion classification, clustering, and summary. Additionally, a fast human face detection algorithm has been incorporated into a video abstraction system. Each process developed has been compared with a competitive algorithm. One of the major advantages of compressed-domain processing is the avoidance of computational overhead.

In this work, a novel system that employs fast video analysis and synthesis was developed. Video abstraction has been attracting considerable research, and it is gradually coming to play an important role in the multimedia database area. Since the time this research was begun, only a small number of video abstraction systems have been proposed. The majority of existing work is related to shot change detection. Even shot change detection work in the compressed domain has difficulties in determining a threshold for decision making because the threshold is heavily dependent on the characteristics of individual video sequences. The fast algorithms developed in this work have been integrated into automated video abstraction system. The much faster than real-time processing rates achieved, with non-dedicated hardware and for a non-restricted class of video sequences, represent a truly significant advance for the video processing community.

To develop the proposed abstraction paradigm, it was necessary to investigate a number of different research topics, including image compression, motion estimation,
mathematical morphology, data mining, and human face detection. The algorithms that have been implemented for the purpose of comparisons are both practical and state-of-the-art.

Another independent contribution of this research is an efficient human face detection algorithm that is based on the skin-color in the compressed domain. The human face detection system consists of three stages: the skin detection stage, the template matching stage, and the confirmation stage. First, a DC-image is extracted from a compressed video sequence using DC coefficients of the DCT, and skin regions are separated from non-skin regions using the Bayesian decision rule with GMMs. Then, the location of a face is detected using a deformable template that considers shape and orientation of the face. In the final stage, if applicable, face candidates are verified by using the moments computed directly from the DCT coefficients in the face region. In particular, the coefficients are manipulated differently according to the orientation of the face. Efficiency is gained by selecting DCT-coefficients as few as possible in zigzag order. This algorithm can be applied to JPEG images without any modification as well.

The system developed uniquely includes the clustering algorithm. Useful properties of the SVD are introduced, and used to reduce the redundancies of video representatives. This algorithm combines both the SVD and k-means algorithm. The advantage of working with the algorithm is that as long as the repetitive key frames exist, a fast and compact clustering will be generated.

One of the biggest obstacles with which I was confronted when work on this topic began, was a lack of tools available for processing video sequences. Therefore, I was faced with the task of building my own toolbox and had to leave the comfort of the MATLAB environment. I have implemented an object-oriented MPEG decoder based on an existing MPEG decoder, and developed all of the components of the abstraction system in C++. Also, the display interface and additional functionality of the video
application were developed.

Many of the features incorporated into the video application are tutorial in nature and can be used as educational tools. Therefore, the next generation of students pursuing research in this area will be able to focus their efforts immediately on generating a better abstraction system rather than on becoming familiar with the most intimate details of MPEC. In addition to the new video abstraction system described in this work, a number of additional features are included in the video processing tool. For example, the video application also displays motion vector information, performs image transformations in the compressed domain, displays graphs of parameters used in each algorithm, and includes several different algorithms for shot change detection, human face detection, and video abstraction.

Another benefit of the system proposed in this work is that there are no restrictions placed on the video content to which it can be applied. Unlike most existing video systems that process a small sub-class of videos, the method proposed in this work processes a wide range of sequences. For example, video content that ranges from complicated movie scenes to head-and-shoulder shots and home movies as well as professional videos are processed with ease.

In summary of the overall proposed system, first, shot segmentation and camera movement classification are performed, regardless of the type of video sequence. Next, a simple key-frame extraction algorithm select the representative frames, which are put together using a clustering algorithm based on their visual similarities. Finally, these works lead to an implementation of a system for fast video abstraction with little redundancy according to multi-resolution architecture.

It is believed that the proposed abstraction system can be used as a significant tool for video analysis and summarization. Each module of this system can be used as a pre-filter for video indexing, retrieval, and browsing.
8.1 Contributions

Fast algorithms for video abstraction system have been developed to work directly with compressed data sequences. The primary advantage of the system is the speed and compactness for summarizing a video. The algorithms performed in the compressed domain reported that an average of overall processing time for video abstraction is almost 9 times faster than real-time (real play-time) on 302 × 240 sized videos with a Pentium II, 400 MHz PC.

The proposed event-based abstraction system consists of several modules developed for fast and content-based processing: shot change detection, camera motion classification, clustering, and human face detection.

In the shot change detection module, the types of macroblocks are used with an adaptive threshold regardless of the nature of the video sequence. The effectiveness of the adaptive threshold is verified with a statistical model. The robustness and accuracy have been compared and verified from the experimental results. In order to interpret the contents of a shot, an algorithm for camera motion classification has been developed and implemented in the compressed domain. This algorithm uses motion information, and performs template matching with six predefined basic templates. The main features of the proposed method are

- Camera motion analysis on the MVFs and their sub-MVF avoids heavy complexity spent on full decompression of an MPEG stream and optical flow calculations.

- The preprocessing step that includes the noisy vector filtering increases the reliability of the motion vectors in camera operation detection.

- The qualitative interpretation through the well-defined template works reliably even in the presence of moving objects of large size or noisy motion vectors since the selected templates consider the physical properties of the camera operations.
In the clustering module, useful properties of the SVD were introduced and applied to a new clustering algorithm for video abstraction. The clustering algorithm has advantages over others in terms of speed and compactness. For human face detection, a simple and fast algorithm was developed and compared with two successful algorithms based on neural network and skin-color statistics. Experimental results show that the proposed algorithm for face detection is also fast and robust.

Each module of the abstraction scheme has been implemented in C++ as a stand-alone application and incorporated into a system with a visual interface. The new video tool developed in this work can be extended for further study. In addition, it has many features that can be useful as a tutorial introduction to video processing. A screen-shot illustrating the developed system can be seen in Figure 51.

8.2 Future Works

There are numerous avenues of future work to pursue. The most important ones are listed below. The suggestions below will have a direct and immediate impact on the quality and efficiency of the current video abstraction system. Additional suggestions and details of those discussed here may be found in corresponding individual chapters.

1. The shot detection algorithm should be applied to uncompressed video sequences. The DC-image approach will be a good approach for accelerating processing time.

2. A special GOP structure, which has no bi-directionally predicted frame between anchor frames, should be considered for flexibility for the shot change detection algorithm.

3. Camera movement classification should be extended to detect more camera events such as panning or tilting diagonally and zooming away from the image.
Figure 5.1: A screen-shot illustrating the developed video tools in this work.

Experiments should be made in terms of accuracy and computational cost of using more templates.

4. An investigation of how many bins of histogram are optimal for visual key-frame clustering should be investigated in terms of accuracy and processing time.

5. Reliability of clustering algorithm should be considered when there is no redundancy in key frames extracted from video analysis.

6. The exact position of the face should be investigated for the outputs of the face detection algorithm to connect with more advanced techniques such as human face recognition and human facial expression analysis. The algorithm can be
improved if more AC coefficients such as DC+2AC are used at cost of increasing computational complexity.

7. Robust skin-color detection can be re-thought. The stage of separating skin regions from background is the most important procedure of the proposed face detection algorithm. Although a simple Bayesian approach was applied to the algorithm, any reliable techniques (such as the Watershed transform [63]) can be considered.
APPENDIX A

THE HIERARCHICAL MPEG STRUCTURE

The hierarchical structure layers of MPEG are described below.

- The sequence layer is associated with the encoding parameters.
- The group Of Pictures layer allows for random access frame that is the access unit.
- The slice layer allows for error confinement.
- The macroblock layer is associated with motion compensation and prediction.
- The block layer is associated with the discrete cosine transform.

A.1 Sequence Layer

A sequence consists of all the frames that follow a sequence header until a term sequence_end_code. Encoding and displaying parameters are transmitted with the sequence header. The sequence header can be repeated in order to allow random access, but all the data elements of the repeated sequence header, except those concerning quantization matrices, must have the same values as in the first sequence header. The repeated sequence header must precede either an I-frame or a P-frame in the bitstream. In the case that random access is performed to a P-frame, it is possible that the decoded frames may not be correct.

Because the GOP layer is optional, the management of the frames is performed at the sequence layer. It is important to note that the frames are not coded in the order
in which they are displayed. In particular, the B-frames that use references "from the future" are always coded after the P-frame (or the I-frame) used for backward predictions. The frame reordering causes a delay in the coding and decoding processes. On the coder side, the delay is given by the number of B-frames that must wait for the following P-frame (or I-frame), while the decoder should wait for having full the two frame-memories before starting the display. As a special application, it is possible to code the sequence without any B-frames in the low_delay mode. In this case, the decoder needs only one frame-memory.

### A.2 Group Of Pictures Layer

A GOP consists of all the pictures that follow a GOP header before another GOP header. The GOP layer allows random access because the first picture after the GOP header is an intra-picture that means that it does not need any reference to any other picture. The GOP layer is optional; i.e., it is not mandatory to put any GOP header in the bitstream. In the header there is also the timecode of the first picture of the
GOP to be displayed.

The decoding process, as the GOP header is immediately followed by an intra-picture, can begin at that point of the bitstream. It is possible that some B-pictures, following such I-picture in the bitstream, have references coming from the previous GOP and cannot be correctly decoded. In this case, the GOP is called an open GOP because some references from the previous GOP exist; if a random access to such a GOP is performed, some B-pictures should not be displayed. A GOP is called a closed GOP when either there are no B-pictures immediately following the first I-picture or such B-pictures have not any references coming from the previous GOP (in this case a GOP header flag must be set).

The GOP length is the period (often expressed in frames) by which an intra-frame occurs. It must be noted that such a value cannot be found in the bitstream and it is unnecessary to the decoding process. Furthermore, it is not specified for any fixed period for the intra-frame. As the presence of the intra-frames is quite important for many applications, it is the encoder that has to provide them, while the decoder only has to work with all the valid bitstreams.

A.2.1 Picture

A picture consists of a field or a frame. It is identified in the bitstream by the picture header that specifies the picture_structure and the picture_coding_type.

The possible picture_coding_type are

- **Intra picture** refers to a picture called an I-picture that has no reference to any other picture, but it can be used as a reference for other pictures.

- **Predicted picture** refers to a picture called P-picture that may have references from another picture and can be used as a reference for other pictures. Compared to the I-picture, the P-picture may be coded with a greater efficiency.
• Bidirectional or interpolated picture refers to a picture called B-picture that may have references from two different pictures, but it cannot be used as a reference picture. This kind of picture may be coded with the greatest efficiency. In the display order, one reference picture is before the B-picture and one is after the B-picture, so a picture reordering is necessary.

A.2.2 Video Decoding Process at Picture Layer

In order to get information from the picture layer,

• Decode the picture header and all its possible extensions.

• Decode all the slices that compose the picture.

• If the picture is an I-picture or a P-picture, the decoded image must be kept in the picture memories (the oldest data will be deleted). The I-picture or the P-picture just decoded is not displayed until all the possible B-pictures preceding it in the display order are decoded and displayed. The correct display order is given by the temporal reference field of the picture header.

A.3 Slice Layer

A slice is a portion of an image of size \( 16 \times (n \times 16) \) pixels. Each slice is coded independently from the other slices of the picture. Therefore, the slice layer allows for error confinement because, when errors in the bitstream are detected, the decoder can try to continue the decoding process by looking for the next slice header.

A.3.1 Video Decoding Process at the Slice Layer

Decoding process is listed below;

• Decode slice.vertical.position.

• Decode quantizer.scale.code.
• Decode all the macroblocks that compose the slice.

A.4  Macroblok Layer

A macroblock is a portion of an image that consists of 16×16 picture elements (pixels). At the macroblock layer, motion compensation and prediction are performed and it is possible to change the quantization step. It must be noted that if the picture is an interlaced frame picture, the odd lines of the macroblock belong to the first field and the even lines to the second field.

A.4.1 Video Decoding Process at the Macroblock Layer

• Decode the macroblock mode and the possible quantizer.scale_code.

• If it is an intra-macroblock,
  o Decode the blocks which the macroblock consists of.

• If it is a non intra-macroblock,
  o Decode the prediction mode and the motion vectors.
  o Produce suitable prediction for the macroblock.
  o Decode the blocks which the macroblock consists of and obtain the prediction error values.
  o Add the prediction error values to the prediction.

A.5  Block Layer

A block is a matrix of 8×8 pixels. The blocks can be 8×8 adjacent luminance or chrominance samples or the corresponding DCT coefficients. The DCT is performed at the block layer.
A.5.1 Video Decoding Process at the Block Layer

In the last step for decoding a compressed sequence, the following functions are performed on the block layer;

- Variable length decoding: The bitstream codewords of the block are decoded to form a vector of quantized DCT coefficient.

- Inverse scan: The vector elements are put into a two-dimensional array, which is the block, following one of two possible patterns. The pattern is defined by the flag alternate.scan which is set at the picture layer. The scanning purpose is to optimize the entropy coding.

- Inverse quantization: DCT coefficients are converted to their original range of values.

- Inverse DCT: Eventually the IDCT is performed. Now the block elements represent either the image sample (intra-block) or the prediction-error (non-intra block).
APPENDIX B

MATHEMATICAL MORPHOLOGY

Although an in-depth review of morphological image processing is beyond the scope of this work, a brief introduction to the basic morphological tools employed in the research is presented.

Mathematical morphology is a tool for extracting image components that are useful for representation and description. The technique was originally developed by Matheron and Serra [53] at the Ecole des Mines in Paris. It is a set-theoretic method of image analysis providing a quantitative description of geometrical structures. At the Ecole des Mines, they were interested in analyzing geological data and the structure of materials. Morphology can provide boundaries of objects, their skeletons, and their convex hulls. It is also useful for many pre- and post-processing techniques, especially in edge thinning and pruning.

Generally speaking, most morphological operations are based on simple expanding and shrinking operations. The primary application of morphology occurs in binary images, though it is also used on grey level images. It can also be useful on range images. A range image is one where grey levels represent the distance from the sensor to the objects in the scene rather than the intensity of light reflected from them.

B.1 Set Operations

The two basic morphological set transformations are erosion and dilation. These transformations involve the interaction between an image \( A \) (the object of interest) and a structuring set \( B \), called the structuring element.

Typically the structuring element \( B \) is a circular disc in the plane, but it can be
any shape. The image and structuring element sets need not be restricted to sets in
the 2-D plane, but can be defined in 1, 2, 3, or higher dimensions.

Let \( A \) and \( B \) be subsets of \( Z^2 \). The translation of \( A \) by \( x \) is denoted \( A_x \) and is
defined as

\[
A_x = \{ c : c = a + x, \text{ for } a \in A \}. \quad (2.1)
\]

The reflection of \( B \), denoted \( \hat{B} \), is defined as

\[
\hat{B} = \{ x : x = -b, \text{ for } b \in B \}. \quad (2.2)
\]

The complement of \( A \) is denoted \( A^c \), and the difference of two sets \( A \) and \( B \) is denoted
as \( A - B \).

### B.2 Dilation

Dilation of the object \( A \) by the structuring element \( B \) is given by

\[
A \oplus B = \{ x : \hat{B}_x \cap A \neq \emptyset \}. \quad (2.3)
\]

The result is a new set made up of all points generated by obtaining the reflection of
\( B \) about its origin and then shifting this reflection by \( x \).

Consider the example where \( A \) is a rectangle and \( B \) is a disc centered on the
origin. Note that if \( B \) is not centered on the origin this work will get a translation
of the object as well. Since \( B \) is symmetric, \( \hat{B} = B \). This definition becomes very

![Diagram](image.png)

**Figure 53:** \( A \) is dilated by the structuring element \( B \).

Intuitive when the structuring element \( B \) is viewed as a convolution mask.

126
B.3 Erosion

Erosion of the object \( A \) by a structuring element \( B \) is given by

\[
A \ominus B = \{ x : B_x \subseteq A \}.
\]  
(2.4)

\[ \text{Figure 54: } A \text{ is eroded by the structuring element } B \text{ to give the internal dashed shape.} \]

Dilation and erosion are duals of each other with respect to set complementation and reflection. That is,

\[
(A \ominus B)^e = A^e \oplus \overset{\circ}{B}.
\]  
(2.5)

To see this, consider first the left hand side:

\[
(A \ominus B)^e = \{ x : B_x \subseteq A \}^e.
\]  
(2.6)

Now, if \( B_x \) is contained in \( A \), then \( B_x \cap A^e = \emptyset \), and so

\[
(A \ominus B)^e = \{ x : B_x \cap A^e = \emptyset \}^e.
\]  
(2.7)

However, the complement of the set of all \( x \)'s that satisfy Equation is simply the set of all \( x \)'s such that \( B_x \cap A^e \neq \emptyset \). Thus,

\[
(A \ominus B)^e = \{ x : B_x \subseteq A^e \neq \emptyset \} = A^e \oplus \overset{\circ}{B}.
\]  
(2.8)

B.4 Applications of Morphological Operations

Erosion and dilation can be used in a variety of ways, in parallel and series, to give other transformations including thickening, thinning, skeleton, and many others. Two
very important transformations are opening and closing. Intuitively, dilation expands an image object and erosion shrinks it. Openings generally smooth contours in an image, breaking narrow isthmuses and eliminating thin protrusions. Closings tend to narrow smooth sections of contours, fuse narrow breaks and long thin gulfs, eliminate small holes, and fill gaps in contours.

The opening of \( A \) by \( B \), denoted \( A \circ B \), is given by the erosion of \( A \) by \( B \), followed by a dilation by \( B \), that is,

\[
A \circ B = (A \ominus B) \oplus B. \tag{2.9}
\]

Opening is like "rounding from the inside": the opening of \( A \) by \( B \) is obtained by taking the union of all translates of \( B \) that fit inside \( A \). Parts of \( A \) that are smaller than \( B \) are removed. Thus,

\[
A \circ B = \bigcup \{B_x : B_x \subset A\}. \tag{2.10}
\]

Closing is the dual operation of opening and is denoted by \( A \bullet B \). It is produced by the dilation of \( A \) by \( B \), followed by an erosion by \( B \): This is like "smoothing from the outside". Holes are filled in and narrow valleys are "closed". Just as with dilation and erosion, opening and closing are dual operations. That is

\[
(A \bullet B)^c = (A^c \circ B^c). \tag{2.11}
\]

The opening operation satisfies the following properties:
Figure 56: The closing of A by the structuring element B.

1. \( A \circ B \) is a subset of \( A \).

2. If \( C \) is a subset of \( D \), then \( C \circ B \) is a subset of \( D \circ B \).

3. \( (A \circ B) \circ B = A \circ B \).

Similarly, for the closing operation:

1. \( A \) is a subset of \( A \bullet B \).

2. If \( C \) is a subset of \( D \), then \( C \bullet B \) is a subset of \( D \bullet B \).

3. \( (A \bullet B) \bullet B = A \bullet B \).

Property 3, in both cases, is known as idempotency. It means that any application of the operation more than once will have no further effect on the result. The morphological filter \( (A \circ B) \bullet B \) can be used to eliminate "salt and pepper" noise. Salt and pepper noise is random and uniformly distributed on small noisy elements which are often found in corrupted real images. It typically appears as black dots or small blobs on a white background, and white dots or small blobs on a black object. The background noise is eliminated at the erosion stage, under the assumption that all noise components are physically smaller than the structuring element \( B \). Erosion on its own will increase the size of the noise components on the object. However, these are eliminated at the closing operation.

129
The important thing to note is that morphological operations preserve the main geometric structures of the object. Only features "smaller than" the structuring element are affected by transformations. All other features at "larger scales" are not degraded. This is not the case with linear transformations, such as convolution.

The above operations are applied to a binary image obtained by the skin-tone detection algorithm described in Section 6.2 as shown in the Figure 57. The boundary of a set $A$, denoted $\partial A$, can be obtained by first eroding $A$ with $B$, where $B$ is a suitable structuring element, and then performing the set difference between $A$ and its erosion. That is

$$\partial A = A - (A \ominus B).$$

Typically, $B$ is a $3 \times 3$ matrix of 1's. Region filling can be accomplished iteratively using dilations, complements, and intersections. Suppose this work has an image $A$ containing a subset whose elements are 8-connected boundary points of a region. Beginning with a point $p$ inside the boundary, the objective is to fill the entire region with 1's. Since, by assumption, all non-boundary points are labeled 0, the algorithm begins by assigning 1 to $p$, and construct

$$X_k = (X_{k-1} \ominus B) \cap A^c, \text{ for } k = 1, 2, \cdots,$$

where $X_0 = p$, and $B$ is the "cross" structuring element. The algorithm terminates when $X_k = X_{k-1}$. The set union of $X_k$ and $A$ contains the filled set and its boundary. Likewise, connected components can also be extracted using morphological operations. If $Y$ represents a connected component in an image $A$ and a point $p$ in $Y$ is known, then the following iterative expression yields all the elements of $Y$:

$$X_k = (X_{k-1} \ominus B) \cap A, \text{ for } k = 1, 2, \cdots,$$

where $X_0 = p$ and $B$ is a $3 \times 3$ matrix of 1's. If $X_k = X_{k-1}$, the algorithm has converged, and it sets $Y = X_k$. An important step in representing the structural
shape of a planar region is to reduce it to a graph. This is very commonly used in robot path planning. This reduction is most commonly achieved by reducing the region to its skeleton. The skeleton of a region is defined by the medial axis transformation (MAT). The MAT of a region $R$ with border $B$ is defined as follows: for each point $p$ in $R$, the algorithm finds its closest neighbor in $B$. If $p$ has more than one such closest neighbor, then $p$ belongs to the medial axis (or skeleton) of $R$. Some examples with the usual Euclidean metric are shown in Figure 58. Direct implementation of the MAT is computationally prohibitive. However, the skeleton of a set can be expressed in terms of erosions and openings. Thus, it can be shown that

$$S(A) = \bigcup_{k=0}^{K} S_k(A), \quad (2.15)$$

where

$$S(A) = \bigcup_{k=0}^{K} \{A \ominus kB\} - [(A \ominus kB) \cap B], \quad (2.16)$$

$B$ is a structuring element, $(A \ominus kB)$ indicates $k$ successive erosions of $A$, and $K$ is the last iterative step before $A$ erodes to an empty set. Thus $A$ can be reconstructed from its skeleton subsets $S_k(A)$ using the equation

$$S(A) = \bigcup_{k=0}^{K} (S_k(A) \oplus kB), \quad (2.17)$$

where $S_k(A) \oplus kB$ represents $k$ successive dilations of $S_k(A)$. 

131
Figure 57: Results of morphological operations on a binary image by a disk element structure of size 2; (a) original image, (b) erode operation, (c) dilate operation, (d) opening operation, and (e) closing operation.

Figure 58: The skeletons of three simple regions.
REFERENCES


135


[55] Shahararay, B., "Scene change detection and content-based sampling of video

[56] Shahararay, B. and Gibbon, D. C., "Automatic generation of pictorial tran-
scriptions of video programs," in Proc. of IISST/SPIE, vol. 2417, pp. 512-519,
1995.

[57] Shin, T., "Hierarchical scene change detection in an mpeg-2 compressed video

[58] Smoliar, S. W. and Zhang, H. J., "Content-based video indexing and re-

[59] Sudhir, G. and Lee, J. C. M., "Video annotation by motion interpretation
using optical flow stream," J. Vis. Commun. Image Represent., vol. 4, pp. 354-
368, Dec. 1996.

[60] Taniguchi, Y., "An intuitive and efficient access interface to real-time incoming


[63] Vincent, L. and Soille, P., "Watersheds in digital spaces: an efficient algo-

[64] Wang, H. and Chang, S. F., "A highly efficient system for automatic face

[65] Wolf, W., "Key frame selection by motion analysis," in ICASSP '96, vol. 2,
p. 1228-1231, 1996.

[66] Xiong, W. and Lee, J. C. M., "Efficient scene change detection and camera
motion annotation for video classification," Computer Vision Image Understand-

[67] Y. Li, T. Z. and Tretter, D., "An overview of video abstraction techniques,

[68] Y. Z. Huang, Y. Rui, T. S. H. and Mehrotra, "Adaptive key-frame ex-


Sangkeun Lee was born on September 10, 1970 in Chung-Nam, Korea (Republic Of Korea). After receiving a Bachelor of Science degree and a Master of Science degree in Electrical Engineering at the Chung-Ang University in 1996 and in 1999, respectively, he began his graduate work for Ph.D. as a graduate research assistant under the supervision of Dr. Monson H. Hayes at the Georgia Institute of Technology with a specialization in digital signal processing. His research was conducted in the Center for Signal and Image Processing (CSIP) in Atlanta, Georgia, and focused on video analysis and abstraction in the compressed domain. Sangkeun defended his Ph.D. dissertation in October 2003 and began working as a Senior Member of Software Engineering at SBA in Irvine, California in November 2003. In December 2003, he received the Ph.D. degree in Electrical Engineering at Georgia Tech.

Non-academic highlights of his graduate years include his wedding to Hyunkryung Back in June 1999 and his participation on Southeast United State Kendo federation (SEUSKF) team that was All United State Kendo Federation (AUSKF) championship in July 2002. In 2001, he was 3rd pace in SEUSKF team tournament. In 2002, he was 2nd place and 3rd place in SEUSK team and individual tournaments, respectively. In 2003, he passed 4th dan only in Kendo practice.