TR3437

VISUAL-CONTENT ANALYSIS FOR MULTIMEDIA RETRIEVAL SYSTEMS

Proefschrift

BIBLIOTHEEK TECHNISCHE

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College voor Promoties aangewezen,
op dinsdag 7 december 1999 te 10.30 uur

door

Alan HANJALIĆ

Diplom-Ingenieur der Elektrotechnik
Friedrich-Alexander-Universität Erlangen-Nürnberg, Duitsland

geboren te Sarajevo, Bosnië-Herzegovina

Dit proefschrift is goedgekeurd door de promotoren:

Prof.dr.ir. J. Biemond
Prof.dr.ir. R.L. Lagendijk

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. J. Biemond	Technische Universiteit Delft, promotor
Prof.dr.ir. R.L. Lagendijk	Technische Universiteit Delft, promotor
Prof.dr.ir. P.F.A. van Mieghem	Technische Universiteit Delft
Prof.dr.ir. H.J. Sips	Technische Universiteit Delft
Prof.dr. I.T. Young	Technische Universiteit Delft
Prof.dr. A. Kohlrausch	Technische Universiteit Eindhoven
Dr. H. Zhang	Microsoft Research, Beijing, China

Het promotieonderzoek heeft gedeeltelijk plaatsgevonden in het EU-ACTS project SMASH (Storage for Multimedia Applications Systems in the Home).

Visual-Content Analysis for Multimedia Retrieval Systems
Hanjalic, Alan
Thesis Delft University of Technology – With ref. – With summary in Dutch

ISBN 90-9013189-2

Copyright © 1999 by A. Hanjalic

All rights reserved. No part of this thesis may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, any information storage and retrieval system, or otherwise, without written permission from the copyright owner.

Contents

INTRODUCTION **1**

1.1 INFORMATION RETRIEVAL SYSTEMS: FROM NEEDS TO POSSIBLE SOLUTIONS 1
1.2 SCOPE OF THE THESIS 6
1.3 THESIS OVERVIEW 7

STATISTICAL FRAMEWORK FOR SHOT-BOUNDARY DETECTION **11**

2.1 INTRODUCTION 11
2.2 PREVIOUS WORK ON SHOT-BOUNDARY DETECTION 16
  2.2.1 Discontinuity values from features and metrics 16
  2.2.2 Detection approaches 21
2.3 A ROBUST STATISTICAL FRAMEWORK FOR SHOT-BOUNDARY DETECTION 28
2.4 DETECTOR FOR ABRUPT SHOT BOUNDARIES 32
  2.4.1 Features and metrics 32
  2.4.2 A priori probability function 33
  2.4.3 Scalar likelihood functions 34
  2.4.4 PMI and the conditional probability functions 35
  2.4.5 Experimental validation 37
2.5 DISCUSSION 37

AUTOMATICALLY ABSTRACTING VIDEO USING KEY FRAMES **39**

3.1 INTRODUCTION 39
3.2 PREVIOUS WORK ON KEY-FRAME EXTRACTION 43
3.3 EXTRACTING KEY FRAMES BY APPROXIMATING THE CURVE OF VISUAL-CONTENT VARIATIONS 47
  3.3.1 Modeling visual content variations along a shot 48
  3.3.2 Distributing N key frames over the sequence 50
  3.3.3 Distributing key frames within a shot 52
  3.3.4 Experimental validation 55
3.4 KEY-FRAME EXTRACTION BASED ON CLUSTER-VALIDITY ANALYSIS 57
  3.4.1 Clustering 58
  3.4.2 Cluster-validity analysis 60
  3.4.3 Key frames from clusters 64
  3.4.4 Experimental validation 64
3.5 DISCUSSION 67

HIGH-LEVEL VIDEO CONTENT ANALYSIS **69**
4.1 INTRODUCTION 69
4.2 RELATED WORK 74
  4.2.1 Detecting different temporal events in a video *74*
  4.2.2 Detecting scene boundaries in a movie *75*
  4.2.3 Extracting the most characteristic movie segments *76*
  4.2.4 Automated recognition of video genres *76*
  4.2.5 News-program analysis *77*
  4.2.6 Methods for analyzing sports programs *81*
4.3 AUTOMATICALLY SEGMENTING MOVIES INTO LOGICAL STORY UNITS 82
  4.3.1 Hierarchical model of a movie structure *82*
  4.3.2 Definition of LSU *83*
  4.3.3 Novel approach to LSU boundary detection *86*
  4.3.4 Inter-shot dissimilarity measure *88*
  4.3.5 Experimental validation *90*
4.4 DETECTING ANCHORPERSON SHOTS IN NEWS PROGRAMS 94
  4.4.1 Assumptions and definitions *95*
  4.4.2 Finding a template *96*
  4.4.3 Template matching *97*
  4.4.4 Experimental validation *98*
4.5 DISCUSSION 101

TRENDS IN IMAGE CODING: THE "FOURTH CRITERION" **105**
5.1 INTRODUCTION 105
5.2 A CONCEPT OF AN ALTERNATIVE IMAGE CODEC 109
5.3 IMAGE CODEC BASED ON SIMPLIFIED VQ 113
  5.3.1 Code-book generation *113*
  5.3.2 Finding the block correspondences *117*
  5.3.3 Compressed image format specification *117*
5.4 PERFORMANCE EVALUATION 119
  5.4.1 CODEC performance regarding classical criteria *119*
  5.4.2 Reduction of the content access work *123*
5.5 DISCUSSION 126

DIRECTIONS FOR FUTURE RESEARCH **127**
6.1 WHERE ARE WE NOW? 127
6.2 WHERE TO GO? 128

REFERENCES **131**

SUMMARY **139**

SAMENVATTING **143**

ACKNOWLEDGMENTS **147**

CURRICULUM VITAE & BIBLIOGRAPHY **149**

Chapter 1

Introduction

1.1 Information retrieval systems: From needs to possible solutions

There is hardly a better way to describe the development stage of our civilization at the end of the second millennium than as the *information era,* and this has a quite obvious reason: never before was the impact of information on the human lifestyle and way of thinking as enormous as it is in the second half of this century. People are not only exposed to information all the time, this experience also becomes more intensive, which greatly contributes to broadening their views; they acquire knowledge and awareness about the environment and the world in general. This process globalizes society, and as such, creates new living and educational standards.

We can explain such an impact mainly as a consequence of an overwhelming *digital revolution,* which started some decades ago and has continuously gained in strength. On the one hand, the digital way of representing information opened completely new perspectives for further developments in information technology. It became possible to compress information, which resulted in a strong reduction of the time and channel capacity required for its transmission and of the space required for its storage. Information can be transmitted or manipulated without quality loss and it is possible to combine and transmit or process different types of information together, like audio, visual or textual: *multimedia* was born. On the other hand, digital hardware technology has rapidly developed and grown in the last decades, so that the performance-versus-price ratio of various digital systems, storage and transmission media steadily increased. All this has led to continuous advances in the quality of transmitted and received audiovisual information [Hua99a], in digital telecommunication networks providing high-speed information transfer ("information superhighway"), in fast digital signal processors and in compact high-capacity storage media like Digital Versatile Disc (DVD), which is seen by many as "the epitome of the digital age" [TNO97]. In view of such technological growth, it is not difficult to understand that an average information consumer easily raises his expectations regarding the amount, variety and technical quality of the received information, as well as of the systems for information receiving, processing, storage and re- or display. It will soon become quite usual that each household is equipped with receivers for Digital Video and Audio Broadcasting (DVB [ETS94] and DAB [ETS97]) providing together hundreds of high-quality audiovisual channels, accompanied by a broadband Internet connection, which gives access to countless on-line information archives all over the world.

However, it is beyond human capabilities to digest all the received information in an on-line manner. Large volumes of digital information obtained from digital TV/radio channels, Internet etc. will need to be stored temporarily, or if they are of long-term value, permanently. In this sense, we witness a strong development of *home digital multimedia archives* [SMA]. And, naturally, with an increasing information production even larger digital multimedia archives appear at service providers (e.g. TV and radio broadcasters, Internet providers, etc.). Thus, the issue of digital information storage steadily becomes more and more interesting and we can talk about emerging *digital libraries*. This term stands for a (large-scale) collection of stored digital information of any type (e.g. audio, visual, textual), made for either professional or consumer environments; examples of this are digital museum archives, Internet archives, video libraries available to commercial service providers and private information collections in the home, all of them being characterized by a quickly increasing capacity and content variety.

The development of digital libraries is not only related to technological advances in high-capacity storage media. The issue of efficiently retrieving the information stored in these libraries becomes of utmost importance as larger data volumes are stored. Actually, it can be said that the missing possibility to quickly access stored information degrades the high technological value of new high-capacity storage media and seriously jeopardizes the usability of the stored information. As nicely formulated in the preface of [Sme97], "anyone who has surfed the Web has exclaimed at one point or another that there is so much information available, so much to search and so much to keep up with". This citation describing a particular problem of finding a desired information on the World-Wide Web (WWW) can analogously be applied to a digital library of any type:

> *If information of interest is not easily accessible within a large digital library, that information can be of no use, in spite of its value and the fact that it is present in that library.*

Manually searching through GBytes of unorganized stored data is a tedious and time-consuming task. Consequently, with increasing information volumes the need grows for shifting the information retrieval to automated systems. There, algorithms are applied capable of performing any information retrieval task with the same reliability and with the same or even higher efficiency as when the retrieval is done manually.

Realizing this shifting in practice is not a trivial problem, especially in the case of images or video*. To explain this, we here analyze some characteristic retrieval tasks, such as "find me an image with a bird", "find me the movie scene where Titanic hits the iceberg", "find me the CNN business news report from 15 November 1999", "find me a 'western' movie in the database", "classify all the images according to the place where they were taken" or "find me all images showing Paris". These retrieval tasks are formulated on a *cognitive level,* according to the human capability of understanding the information content and analyzing it in terms of objects, persons, sceneries, meaning of speech fragments or the context of a story in general. Opposed to this, the only feasible analysis of a video or an image at the algorithmic or *system level* can be in terms of their *features,* such as color, texture, shape, frequency components, audio and speech signal characteristics, and using the algorithms operating on these features. Such algorithms are, for instance, image segmentation, detection of moving objects, extraction of textures and shapes, recognition of color compositions, determination of relations among different objects or analysis of the frequency spectrum of the audio or speech signal. These algorithms can be developed using the state-of-the-art in image and audio analysis and processing, computer vision, statistical signal processing, machine intelligence, pattern recognition and other related areas.

---

* Within the context of this thesis we refer to *video* as to a program in its entirety, consisting of an image sequence and the eventual accompanying audio/speech stream.

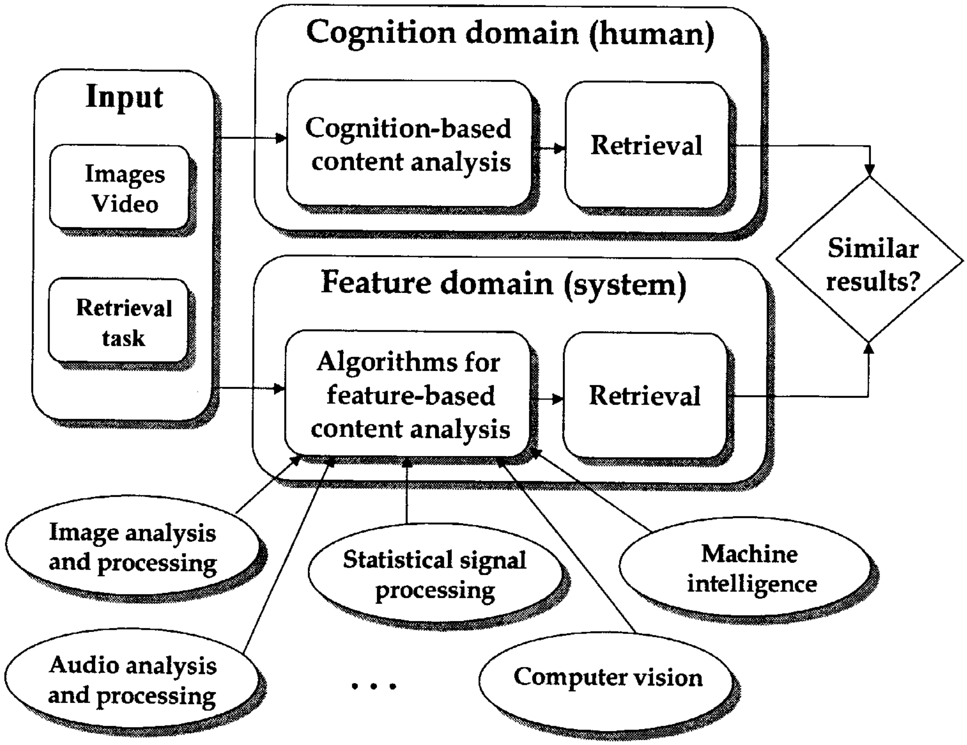

Cognition domain (human)
Input
Images
Video
Retrieval task
Cognition-based content analysis
Retrieval
Similar results?
Feature domain (system)
Algorithms for feature-based content analysis
Retrieval
Image analysis and processing
Statistical signal processing
Machine intelligence
Audio analysis and processing
. . .
Computer vision

Figure 1.1: *Cognitive vs. feature-based retrieval*

As illustrated in Figure 1.1, we can understand an automated feature-based content analysis as a system-level parallel to the cognition-based analysis. There the features are chosen and algorithms are developed in the way that the retrieval results are similar at the end of each branch of the scheme. Experience shows, however, that the parallelism in Figure 1.1 is not viable in all cases. We can explain this with the example of searching for an image containing a bird. While such a search performed by a human will always succeed, this cannot be said for the feature-based image analysis, simply because a complicated and large feature set describing the characteristics of a bird in general is required as well as complex algorithms operating on that feature set, which would enable the system to recognize the appearance of any arbitrary bird, in any possible pose and also in cases where parts of a bird are occluded. Finding a suitable feature set and developing related algorithms for such a retrieval task is very difficult, if not impossible. Consequently, the development of feature-based content analysis algorithms for the scheme in Figure 1.1 has not been directed to enable queries on the highest semantic level, such as the above example with a bird, but mainly towards extracting certain semantic aspects of the information which would allow for a reduction of the overall large search space. This tendency can be recognized in numerous algorithms proposed in recent literature, many of which will be explained in detail in further chapters of this thesis.

For instance, an algorithm in [Vai98] is able to classify with high accuracy images showing a city versus those showing a landscape. Further in [Vai99], a Bayesian framework is presented for semantic classification of outdoor vacation images. There, landscape images can be classified into those showing a sunset, a forest or mountains. Similar examples can also be found in the area of digital video libraries. The algorithms proposed in [Han99b] and [Yeu97] provide the possibility to detect episode or scene boundaries of a TV broadcast (movie, situation comedy, etc.). A methodology for detecting commercial breaks in a TV news program is presented in [Liu98]. There, the audio track of the broadcast is analyzed and commercial breaks are efficiently separated from the rest of the program because of their specific audio characteristics. An approach to detect commercial breaks in an arbitrary TV broadcast is presented in [McG99], based on parallel investigation of several feature types. Also a sophisticated feature-based analysis is applied in [Fis95] in order to classify video programs in different genres. Another class of approaches [DeM98], [Han97c], [Han00], [Pfe96] concentrates on *video abstraction*, i.e. compact representation of long video sequences by extracting and organizing a number of its most representative frames and segments.

Even the feature-based content-analysis techniques belonging to the current technological state-of-the-art and developed with the objective of search-space reduction can be used to build efficient tools for multimedia information retrieval, since they provide the user with reliable directions for browsing efficiently through a digital library and lead them quickly to the information of interest. The MPEG-7 standardization platform [ISO97] addresses ways to define standard sets of descriptors for multimedia information based on features which should provide further directions for the development of feature-based content-analysis algorithms. And with new solutions, the performance of information retrieval systems can only improve, leading to a further reduction of the search space and user's involvement during the search procedure. The material presented in this thesis, which is briefly outlined in the next two sections of this chapter, is a further contribution to this positive development.

1.2 Scope of the thesis

This thesis concentrates on a feature-based analysis of the *visual content* of images and video, enabling an easier image and video retrieval from large-scale multimedia digital libraries. The methodologies presented here are mainly the result of research performed for the European ACTS project SMASH (Storage for Multimedia Application Systems in the Home [SMA]).

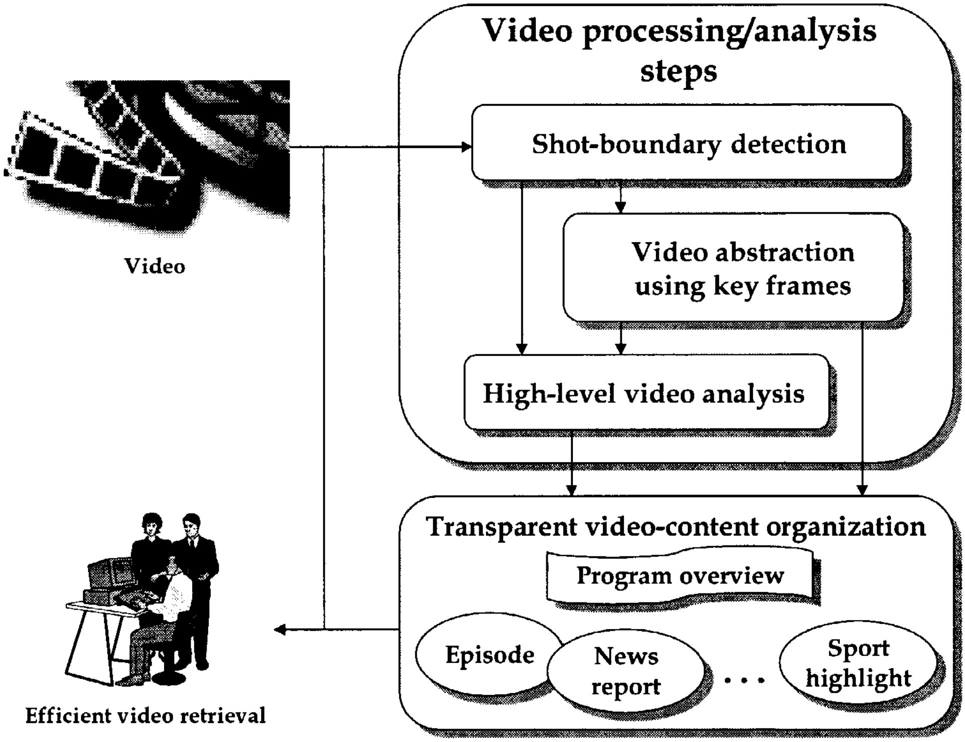

Video processing/analysis steps
Shot-boundary detection
Video abstraction using key frames
High-level video analysis
Video
Transparent video-content organization
Program overview
Episode
News report
. . .
Sport highlight
Efficient video retrieval

Figure 1.2: *Video-content analysis scheme enabling its efficient retrieval*

The scheme in Figure 1.2 presents a series of video processing/analysis steps which provide an organizational structure allowing efficient reviewing of the global video content (e.g. story flow of a movie, topic series of a news program, etc.) and a fast access to and retrieval of any arbitrary part of a video (e.g. an arbitrary movie episode, a news report on a certain topic, a highlight of a sport program, etc.). The scheme depicts a generally known video-analysis procedure which first breaks up a video into temporally homogeneous segments called *video shots*, then condenses these segments into a set of characteristic frames called *key frames* and finally performs a high-level analysis of a video content. This high-level analysis basically includes determining "semantic" relationships among shots (e.g. their grouping into news reports, movie episodes, etc.) using temporal characteristics of shots and suitable features of their key frames. As indicated in the scheme, beside of being used for high-level video analysis key frames also directly participate in forming the organizatorial video-content structure described above. There, they provide visual keys to different aspects of a video content. A large number of algorithms was presented in recent literature for all three mentioned processing/analysis steps, aiming at a robust and high-quality performance with as much automation as possible. We contribute to these efforts in the first part of this thesis and dedicate each of the Chapters 2 to 4 to one of the processing/analysis steps in Figure 1.2.

In the second part of this thesis we consider the fact that the prevailing amount of information reaching the digital libraries and being stored there will be in *compressed* form. This is because large and fast advances in the compression area are gladly employed to maximally utilize the available storage space in digital libraries, but also to increase the information-transmission rate and density. Consequently, compressed images and video need to be expected as inputs into feature-based content-analysis algorithms, which, however, must not influence the efficiency of these algorithms compared to the case where they operate on uncompressed data. The most important issue related to this efficiency is the possibility to easily reach all the necessary features in a compressed image/video. We address this issue in Chapter 5 for the case of image compression.

1.3 Thesis overview

Dividing a video sequence into *shots* is the first step towards video-content analysis and content-based video browsing and retrieval. A video shot is defined as a series of interrelated consecutive frames, taken contiguously by a single camera and representing a continuous action in time or space [Bor93]. As such, shots are considered to be the primitives for higher-level content analysis, indexing and classification, discussed in later chapters of this thesis. Chapter 2 presents a statistical framework for shot-boundary detection based on minimization of the average detection-error probability. We model the required statistical functions using a robust metric for visual content discontinuities (based on motion compensation) and take into account knowledge about the shot-length distribution and visual discontinuity patterns at shot boundaries. Major advantages of the proposed framework are its robust and sequence independent detection performance, as well as its capacity to detect different types of shot boundaries simultaneously. Chapter 2 is based on our work published in the Proceedings of the IEEE International Conference on Multimedia Computing and Systems 1999 [Han99d].

Abstracting a video by extracting a number of characteristic or *key frames* is useful for different applications in video libraries. The form and the size of the key-frame abstract needs, however, to be adapted to the structure of the video material, as well as to the targeted application. Chapter 3 presents two methods for extracting key frames, aiming at different applications in video-retrieval systems. The first method is characterized by the possibility to control the total number of key frames extracted for the entire sequence. While this number does not exceed the prespecified maximum, key frames are spread along a video such that the quality of capturing all relevant variations of its visual content is maximized and that a storyboard of a video is provided. The objective of the second approach to key-frame extraction presented in Chapter 3 is to minimize the size of the key-frame abstract while providing all the necessary aspects of the visual content of a video. This algorithm is designed to produce a set of key frames which capture the content of a video in a similar way as when key frames are extracted manually based on human cognition. Chapter 3 is based on our publications in *Image Databases and Multi-Media Search* (World Scientific Singapore, 1997) [Han97c] and IEEE Transactions on Circuits and Systems for Video Technology [Han00].

As already mentioned in Section 1.1, information retrieval from digital libraries by formulating queries on the highest semantic level is not realistic in view of the current technological state-of-the art. However, examples were also shown where certain semantic components can be recognized in the stored information and be used to organize the information in such a way that the overall large search space is reduced as far as possible. Using, for instance, the algorithm from [Vai98] for city-versus-landscape image classification, the time for finding an image showing the New York skyline can be considerably reduced since only relevant images, i.e. those showing cities, are submitted to the user. Although he still needs to browse through the city image collection and must search for the particular image of interest (New York), the number of images he needs to check is much smaller than the entire image library.

In Chapter 4 we first present an idea how to translate the above image-search example to the case of video retrieval, and especially retrieval of *movies*, which is a very important program category in video storage systems. We assume that a typical movie can be represented as a series of high-level semantic contexts called *episodes*, which correspond to different classes in an image database. If a movie is segmented into episodes, a search for different movie segments showing specific faces or sceneries can be performed only within the relevant episode, which reduces the overall search space and, therefore, also the retrieval time. We develop a feature-based algorithm for automatically segmenting movies into *logical story units*, which are the approximates for the actual movie episodes. Movie segmentation into logical story units is followed by the description of an algorithm for analyzing TV news programs at a high level. The algorithm detects the appearance of anchorperson shots, which can be considered as the first step in recovering the report structure of a news program at a later stage. The material presented in Chapter 4 is based on our work published in the Proceedings of SPIE/IS&T Electronic Imaging: Storage and Retrieval for Image and Video Databases VII, 1999 [Han99a] and in IEEE Transactions on Circuits and Systems for Video Technology [Han99b].

Chapter 5 addresses the issue of accessing the spatial (pixel-level) content in compressed images, such that the efficiency of image-database operations (e.g. image analysis, comparison, query) is not reduced due to compression. We present a novel image-compression methodology which was primarily developed to suit emerging applications in large-scale image databases. In this methodology, the access to spatial image content in the compressed domain is simplified first by having a highly simple decompression procedure, but also by the fact that some aspects of the spatial image content are directly accessible in the compressed image format. While we concentrated on reducing the *content access work*, we also took into account the requirements of keeping a good compression factor, providing a high perceptual quality of reconstructed images and keeping the overall computational costs in limits. Chapter 5 is based on our publication in the Proceedings of the IEEE International Conference on Image Processing, 1999 [Han99c]

In spite of all advances made in the area of image and video retrieval in the last years, we must remain aware of the fact that the development of robust retrieval systems is still in its infancy. In Chapter 6, we discuss the current development stage of these systems and give some important directions for pursuing further research in this area.

Chapter 2

Statistical Framework for Shot-Boundary Detection

2.1 Introduction

The basis of detecting shot boundaries in video sequences is the fact that frames surrounding a boundary generally display a significant change in their visual contents. The detection process is then the recognition of considerable *discontinuities* in the visual-content flow of a video sequence. The process of shot-boundary detection, having as input two frames k and $k+l$ of a video sequence, is illustrated in Figure 2.1. Here l is the interframe distance with a value $l \geq 1$.

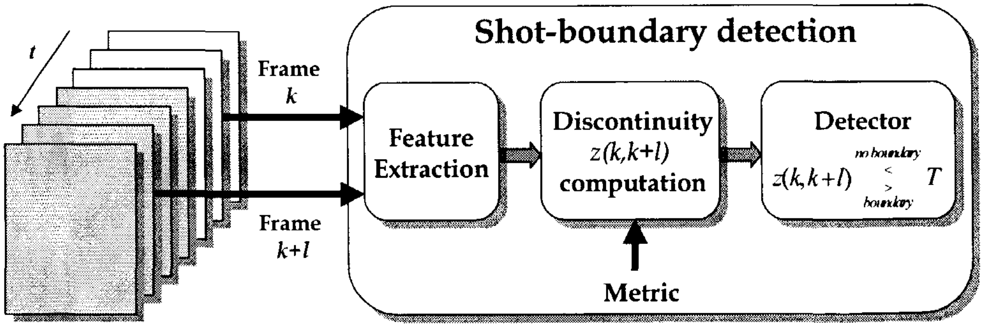

t
Frame k
Frame k+l
Shot-boundary detection
Feature Extraction
Discontinuity z(k,k+l) computation
Detector
$z(k,k+l) \underset{boundary}{\overset{no\ boundary}{\lessgtr}} T$
Metric

Figure 2.1: *Illustration of the process for detecting a shot boundary between frames k and k+l*

In the first step of the process, *feature extraction* is performed. Within the context of this thesis, extracted features depict various aspects of the visual content of a video. Then, a *metric* is used to quantify the feature variation from frame k to frame $k+l$. The discontinuity value $z(k,k+l)$ is the magnitude of this variation and serves as an input into the *detector*. There, it is compared against a *threshold* T. If the threshold is exceeded, a shot boundary between frames k and $k+l$ is detected.

To be able to draw reliable conclusions about the presence or absence of a shot boundary between frames k and $k+l$, we need to use the features and metrics for computing the discontinuity values $z(k,k+l)$, that are as discriminating as possible. This means that a clear separation should exist between discontinuity-value ranges for measurements performed *within shots* and *at shot boundaries*. In the following, we will refer to these ranges as $\overline{R}$ and R, respectively. The problem of having unseparated ranges $\overline{R}$ and R is illustrated in Figure 2.2, where some discontinuity values within shot 1 belong to the overlap area. Such values $z(k,k+l)$ make it difficult to decide about the presence or absence of a shot boundary between frames k and $k+l$ without avoiding detection mistakes, i.e. *missed* or *falsely detected* boundaries.

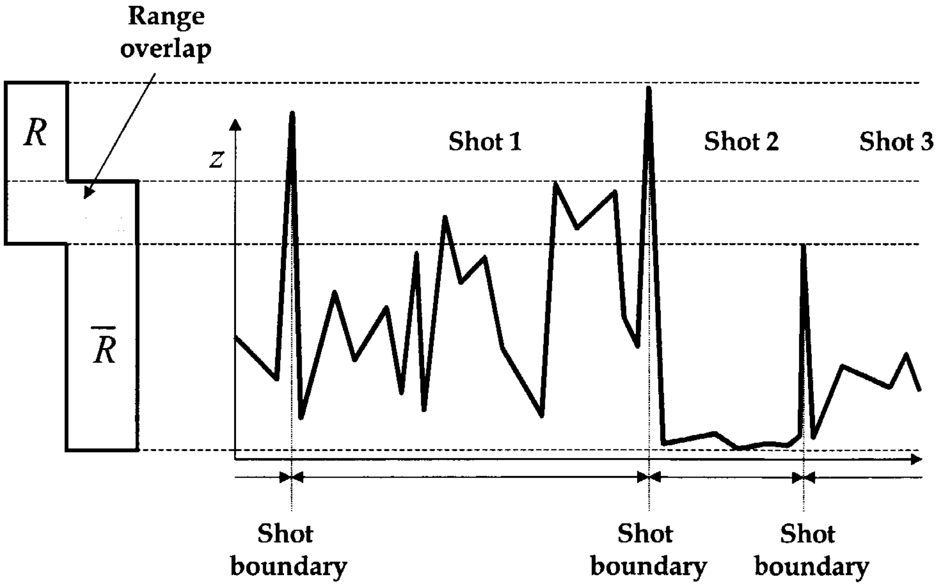


Figure 2.2: *The problem of unseparated ranges* $\overline{R}$ *and* R

We realistically assume that the visual-content differences between consecutive frames within the same shot are mainly caused by two factors: *object/camera motion* and *lighting changes*. Depending on the magnitude of these factors, the computed discontinuity values within shots vary and sometimes lie in the overlap area, as shown in Figure 2.2. Thus, an effective way to better discriminate between discontinuity values belonging to ranges $\overline{R}$ and R is to use features and metrics that are insensitive to motion and lighting changes. However, this is not the only advantage of using such features and metrics. Since different types of sequences can globally be characterized by their average rates and magnitudes of object/camera motion and lighting changes (e.g. high-action movies vs. stationary dramas), eliminating these distinguishing factors also provides a high level of *consistency* of ranges $\overline{R}$ and R across different sequences. If the ranges $\overline{R}$ and R are consistent, the parameters of the detection system (e.g. the threshold T) can first be optimized on a set of training sequences to maximize the detection reliability, and then the system can be used to detect shot boundaries in an arbitrary sequence without any human supervision, while retaining a high detection reliability.

As will be shown in the following section, *motion compensating* features and metrics can be found, capable of considerably reducing the influence of motion on discontinuity values. However, the influence of strong and abrupt lighting changes, induced by flashes or a camera directed to a light source, cannot be reduced in this way. For instance, one could try working only with chromatic color components, since the common lighting changes can mostly be captured by luminance variations. But this is not an effective solution in extreme cases, where all color components are changed. Strong and abrupt lighting changes can result in a series of high discontinuity values, which can be mistaken for the actual shot boundaries. In the remainder of this chapter we define possible causes for high discontinuity values within shots as *extreme factors*. These factors basically include strong and abrupt lighting changes, as well as some extreme motion cases, which cannot be captured effectively by motion compensating features and metrics.

While the influence of extreme factors on discontinuity values cannot be neutralized by choosing suitable features and metrics, it is possible to neutralize such influences by embedding *additional information* in the shot-boundary detector. For instance, the *temporal patterns* formed by consecutive discontinuity values can be investigated for this purpose. Then, the decision about the presence or absence of a shot boundary between frames *k* and *k+l* made by the detector is not only based on the comparison of the computed discontinuity value *z(k,k+l)* and the threshold *T*, but also based on the match between the pattern formed by consecutive discontinuity values surrounding *z(k,k+l)* and a known pattern that is specific for a shot boundary. This is illustrated in Figure 2.3.

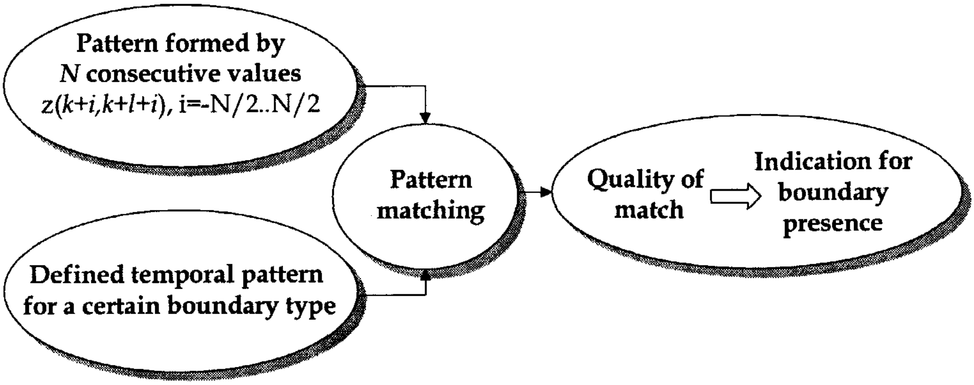

Pattern formed by
N consecutive values
z(k+i,k+l+i), i=-N/2..N/2
Pattern matching
Defined temporal pattern for a certain boundary type
Quality of match
Indication for boundary presence

Figure 2.3: *Matching of the temporal pattern formed by N consecutive discontinuity values and a temporal pattern characteristic for a shot boundary. The quality of match between two patterns provides an indication for boundary presence between frames k and k+l*

Different types of shot boundaries need to be taken into account during the detection process, where each of these types is characterized by its own characteristic temporal pattern. We can distinguish *abrupt boundaries*, which are the most common boundaries and occur between two consecutive frames *k* and *k*+1, from *gradual transitions*, such as fades, wipes and dissolves, which are spread over several frames.

Beside the information on temporal boundary patterns, the *a priori* information describing global knowledge about the visual-content flow can also be taken into account when detecting shot boundaries. An example of such information is the dependence of the probability for a shot boundary on the shot length. While being almost zero at the beginning of a shot, this probability rises with increasing shot length and converges to "1". In this way, the information on shot lengths is also highly efficient in preventing false detections due to extreme factors.

If we combine the usage of motion compensating features and metrics for computing the discontinuity values with embedding the additional information in the detector to reduce the influence of extreme factors on these values, we are thus likely to obtain highly reliable detection results. The scheme of such a detection procedure is illustrated in Figure 2.4.

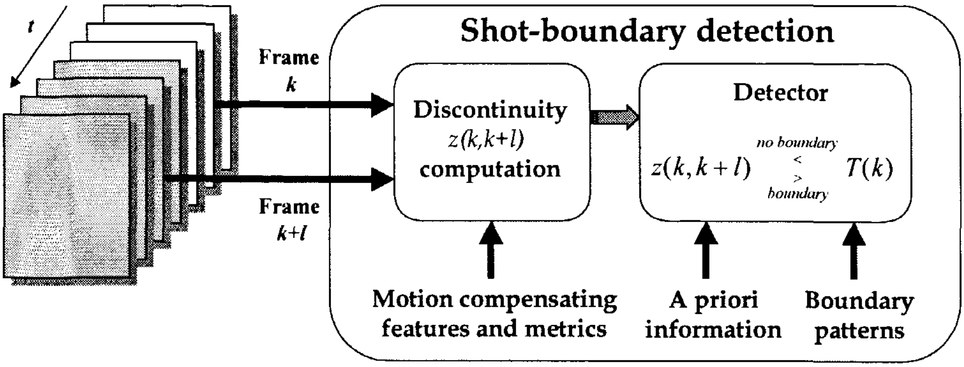

t
Frame k
Frame k+l
Shot-boundary detection
Discontinuity z(k,k+l) computation
Detector
z(k,k+l) no boundary < > boundary T(k)
Motion compensating features and metrics
A priori information
Boundary patterns

Figure 2.4: *A shot-boundary detector with improved detection performance regarding a reduction of the false-detection rate*

Compared to the detector in Figure 2.1, the threshold T does not remain constant but has a new value at each frame k. This is the consequence of the embedded additional information which regulates the detection process by continuously adapting the threshold to the quality of the pattern match for each new series of consecutive discontinuity values and the time elapsed since the last detected shot boundary. The remaining issue is to find the function $T(k)$ providing the optimal detection performance. Statistical detection theory provides means for solving this problem efficiently. Using the statistical properties of discontinuity values and the additional information embedded in the detector, we can compute the threshold function $T(k)$ such, that the average probability for detection mistakes is minimized.

After reviewing existing approaches to shot-boundary detection in Section 2.2, we develop in Section 2.3 a statistical framework for shot-boundary detection as shown in Figure 2.4, which addresses all the issues discussed above. Due to the consistent ranges $\overline{R}$ and R, a high generality of functions and parameters used is provided, so that our framework can operate without human supervision and is suitable for implementation into fully automated video-analysis systems. In Section 2.4 we apply the proposed detection framework to abrupt shot boundaries and evaluate the detection performance. Finally, a general discussion on the material presented in this chapter can be found in Section 2.5.

2.2 Previous work on shot-boundary detection

The problem of reliably detecting shot boundaries in a video has been the subject of substantial research over the last decade. In this section we give a concise overview of the relevant literature. The overview concentrates, on the one hand, on the capability of features and metrics to reduce the motion influence on discontinuity values. On the other hand, it investigates existing approaches to shot-boundary detection, involving the threshold specification, treatment of different boundary types and usage of additional information to improve the detection performance.

2.2.1 Discontinuity values from features and metrics

Different methods exist for computing discontinuity values, employing various features related to the visual content of a video. Characteristic examples of features used are pixel values, histograms, edges and motion smoothness. For each selected feature, a number of suitable metrics can be applied. Good comparisons of features and metrics used for shot-boundary detection with respect to the quality of the obtained discontinuity values can be found in overview papers [Fur95], [Aha96], [Bor96] and [Lie99].

The simplest way of measuring the discontinuity between two frames is to compute the mean absolute intensity change for all pixels of a frame [Kik92]. We first define $I(x,y)$ as the intensity of the pixel at coordinates (x,y) and compute the absolute intensity change of that pixel between frames k and $k+l$ as

$$D_{k,k+l}(x,y) = | I_k(x,y) - I_{k+l}(x,y) | \qquad (2.2.1)$$

The values (2.2.1) are then summarized over all pixels of the frame with dimensions X and Y, and averaged to give the discontinuity value, that is

$$z(k,k+l) = \frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} D_{k,k+l}(x,y) \qquad (2.2.2)$$

A modification of this technique is only counting the pixels that change considerably from one frame to another [Ots91]. Here, the absolute change of the intensity $I(x,y)$ is compared with the prespecified threshold T_1, and is only considerable if the measured absolute difference exceeds the threshold, that is

$$D_{k,k+l}(x,y)=\begin{cases}1 & if\,|I_k(x,y)-I_{k+l}(x,y)|>T_1\\ 0 & else\end{cases} \tag{2.2.3}$$

An important problem of the two approaches presented above is the sensitivity of discontinuity values (2.2.2) to camera and object motion. To reduce the motion influence, a modification of the described techniques was presented in [Zha93], where a 3x3 averaging filter was applied to frames before performing the pixel comparison. Much higher motion independence show the approaches based on motion compensation. There, a *block matching* procedure is applied to find for each block $b_i(k)$ in frame k a corresponding block $b_{i,m}(k+l)$ in frame $k+l$, such that it is most similar to the block $b_i(k)$ according to a chosen criterion (difference formula) D, that is:

$$D_{k,k+l}(i)=D\big(b_i(k),b_{i,m}(k+l)\big)=\min_{j=1..N_{Candidates}} D\big(b_i(k),b_{i,j}(k+l)\big) \tag{2.2.4}$$

Here, $N_{Candidates}$ is the number of candidate blocks $b_{i,j}(k+l)$, considered in the procedure to find the best match for a block $b_i(k)$. If k and $k+l$ are neighboring frames of the same shot, the values $D_{k,k+l}(i)$ can generally be assumed low. This is because for a block $b_i(k)$ almost the identical block $b_{i,m}(k+l)$ can be found due to a global constancy of the visual content within a shot. This is not the case if frames k and $k+l$ surround a shot boundary, since, in general, the difference between corresponding blocks in the two frames will be large due to a radical change in visual content across a boundary. Thus, computing the discontinuity value $z(k,k+l)$ as a function of differences $D_{k,k+l}(i)$ is likely to provide a reliable base for detecting shot boundaries.

An example of computing the discontinuity values based on the results of block-matching procedure is given in [Sha95a]. There, a frame k is divided into $N_{Blocks}=12$ nonoverlapping blocks and differences $D\big(b_i(k),b_{i,j}(k+l)\big)$ are computed by comparing pixel-intensity values within blocks. Then, the obtained differences $D_{k,k+l}(i)$ are sorted and normalized between 0 and 1 (where 0 indicates a perfect match), giving the values $d^s_{k,k+l}(i)$. These values are multiplied with weighting factors c_i and summarized over the entire frame to give the discontinuity values, that is

$$z(k,k+l)=\sum_{i=1}^{N_{Blocks}} c_i d_{k,k+l}^{s}(i) \tag{2.2.5}$$

A popular alternative to pixel-based approaches is using histograms as features. Consecutive frames within a shot containing similar global visual material will show little difference in their histograms, compared to frames on both sides of a shot boundary. Although it can be argued that frames having completely different visual contents can still have similar histograms, the probability of such a case is small. Since histograms ignore spatial changes within a frame, histogram differences are considerably more insensitive to *object* motion with a constant background than pixel-wise comparisons are. However, a histogram difference remains sensitive to *camera* motion, such as panning, tilting or zooming. If histograms are used as features, the discontinuity value is obtained by bin-wise computing the difference between frame histograms. Both grey-level and color histograms are used in literature, and their differences are computed by a number of metrics. A simple metric is the sum of absolute differences of corresponding bins, with N_{Bins} being the total number of bins, that is

$$z(k,k+l)=\sum_{j=1}^{N_{Bins}} |H_k(j)-H_{k+l}(j)| \tag{2.2.6}$$

when comparing grey-level histograms and

$$z(k,k+l)=\sum_{j=1}^{N_{Bins}} |H_k^R(j)-H_{k+l}^R(j)|+|H_k^G(j)-H_{k+l}^G(j)|+|H_k^B(j)-H_{k+l}^B(j)| \tag{2.2.7}$$

if color histograms are compared [Yeo95a]. In (2.2.6), $H_k(j)$ is the j-th bin of the grey-value histogram belonging to frame k. In (2.2.7), $H_k^R(j)$, $H_k^G(j)$ and $H_k^B(j)$ are the j-th bins of histograms of the R-, G- and B-color component of the image k. Another popular metric is the so-called χ^2-test, proposed in [Nag92] for grey-level histograms:

$$z(k,k+l)=\sum_{j=1}^{N_{Bins}} \frac{|H_k(j)-H_{k+l}(j)|^2}{H_{k+l}(j)} \tag{2.2.8}$$

However, according to experimental results reported in [Zha93], the metric (2.2.8) does not only enhance the discontinuities across a shot boundary, but also the effects caused by camera/object motion. Therefore, the overall detection performance of (2.2.8) is not necessarily better than that from (2.2.6), whereas it does require more computational power.

A metric involving histograms in the *HVC* color space [Fur95] (Hue – color type, Value – intensity, luminance, Chroma – saturation, the degree to which color is present) exploits the advantage of the invariance of Hue under different lighting conditions. This is useful in reducing the influence of common (weak) lighting changes on discontinuity values. Such an approach is proposed in [Arm93a], where only histograms of *H* and *C* components are used. These one-dimensional histograms are combined into a two-dimensional surface, serving as a feature. Based on this, the discontinuity is computed as

$$z(k,k+l)=\sum_{x=1}^{X}\sum_{y=1}^{Y}\{\,|\delta_{k,k+l}(x,y)|\times\Delta_{Hue}\times\Delta_{Chroma}\} \tag{2.2.9}$$

where $\delta_{k,k+l}(x,y)$ is the difference between the bins at coordinates (x,y) in *HC*-surfaces of frames *k* and *k+l*, and Δ_{Hue} and Δ_{Chroma} are the resolutions of Hue and Chroma components used to form the two-dimensional histogram surface.

Also the histograms computed block-wise can be used for shot-boundary detection, as shown in [Nag92]. There, both the images *k* and *k+l* are divided into 16 blocks, histograms $H_{k,i}$ and $H_{k+l,i}$ are computed for blocks $b_i(k)$ and $b_i(k+l)$ and the χ^2-test is used to compare corresponding block histograms. When computing the discontinuity as a sum of region-histogram differences, 8 largest differences were discarded to efficiently reduce the influence of motion and noise. An alternative to this approach can be found in [Ued91], where first the number of blocks is increased to 48, and then the discontinuity value is computed as the total number of blocks within a frame, for which the block-wise histogram difference exceeds a prespecified threshold T_1, that is

$$z(k,k+l)=\sum_{i=1}^{48}D\big(b_i(k),b_i(k+l)\big) \tag{2.2.10}$$

with

$$D\big(b_i(k), b_i(k+l)\big) = \begin{cases} 1 & \text{if } \frac{1}{N_{Bins}} \sum_{j=1}^{N_{Bins}} \frac{\big(H_{k,i}(j) - H_{k+l,i}(j)\big)^2}{H_{k,i}(j)} > T_1 \\ 0 & \text{else} \end{cases} \quad (2.2.11)$$

According to [Ots93], the approach from [Ued91] is much more sensitive to abrupt boundaries than the one proposed in [Nag92]. However, since emphasis is put on blocks, which change most from one frame to another, the approach from [Ued91] also becomes highly sensitive to motion.

Another characteristic feature that proved to be useful in detecting shot boundaries is edges. As described in [Mai95], first the overall motion between frames is computed. Based on the motion information, two frames are registered and the number and position of edges detected in both frames are compared. The total difference is then expressed as the total edge change percentage, i.e. the percentage of edges that enter and exit from one frame to another. Due to registration of frames prior to edge comparison, this feature is robust against motion. However, the computational complexity of computing the discontinuity values is also high. Let p_k be the percentage of edge pixels in frame *k*, for which the distance to the closest edge pixel in frame *k+l* is larger than the prespecified threshold T_1. In the same way, let p_{k+l} be the percentage of edge pixels in frame *k+l*, for which the distance to the closest edge pixel in frame *k* is larger than the prespecified threshold T_1. Then, the discontinuity value between these frames is computed as

$$z(k, k+l) = \max(p_k, p_{k+l}) \quad (2.2.12)$$

Finally, we discuss the computation of the discontinuity value *z(k,k+l)* using the analysis of the motion field measured between two frames. An example for this is the approach proposed in [Aku92], where the discontinuity value *z(k,k+*1) between two consecutive frames is computed as the inverse of *motion smoothness*. For this purpose, we first compute all motion vectors $\vec{v}(b_i(k), b_{i,m}(k+1))$ between frames *k* and *k*+1 and then check if they are significant by comparing their magnitude with a prespecified threshold T_1:

$$w_{i,1}(k) = \begin{cases} 1 & \text{if } |\vec{v}(b_i(k), b_{i,m}(k+1))| > T_1 \\ 0 & \text{otherwise} \end{cases} \quad (2.2.13a)$$

Then, we also take into consideration the frame k+2 and check if a motion vector between frames k and k+1 significantly differs from the related motion vector measured between frames k+1 and k+2. This is done by comparing their absolute difference with a prespecified threshold T_2:

$$w_{i,2}(k) = \begin{cases} 1 & \textit{if } |\vec{v}(b_i(k), b_{i,m}(k+1)) - \vec{v}(b_i(k+1), b_{i,m}(k+2))| > T_2 \\ 0 & \textit{otherwise} \end{cases} \tag{2.2.13b}$$

The sum of values (2.2.13a) for all blocks $b_i(k)$ is the number of significant motion vectors between frames k and k+1, and can be understood as a measure for object/camera velocity. Similarly, the sum of values (2.2.13b) is the number of motion vectors between frames k and k+1 that are "significantly" different from their corresponding vectors between frames k+1 and k+2, and can be understood as the measure for motion continuity along three consecutive frames of a sequence. Using these two sums, we can now compute the motion smoothness at frame k as

$$M(k) = \frac{\sum_{i=1}^{N_{Blocks}} w_{i,1}(k)}{\sum_{i=1}^{N_{Blocks}} w_{i,2}(k)} \tag{2.2.14}$$

The more motion vectors change accross consecutive frames, the lower is the motion smoothness (2.2.14). Finally, the discontinuity value can be obtained as an inverse of (2.2.14), that is

$$z(k, k+1) = \frac{1}{M(k)} = \frac{\sum_{i=1}^{N_{Blocks}} w_{i,2}(k)}{\sum_{i=1}^{N_{Blocks}} w_{i,1}(k)} \tag{2.2.15}$$

2.2.2 Detection approaches

Threshold specification

The problem of choosing the right threshold for evaluating the computed discontinuity values has not been addressed extensively in literature. Most authors work with heuristically chosen global thresholds [Nag91], [Ots91], [Arm93a]. An alternative is given in [Zha93], where the authors first measure the statistical distribution of discontinuity values within a shot. Then they model the obtained distribution by a Gaussian function with parameters μ and σ, and compute the threshold value as

$$T = \mu + r\,\sigma \tag{2.2.16}$$

where r is the parameter related to the prespecified tolerated probability for false detections. For instance, when r=3, the probability of having falsely detected shot boundaries is 0.1%. The specification of the parameter r can only explicitly control the rate of false detections. The rate of missed detections is implicit and cannot be regulated, since the distribution of discontinuity values measured on boundaries is not taken into account.

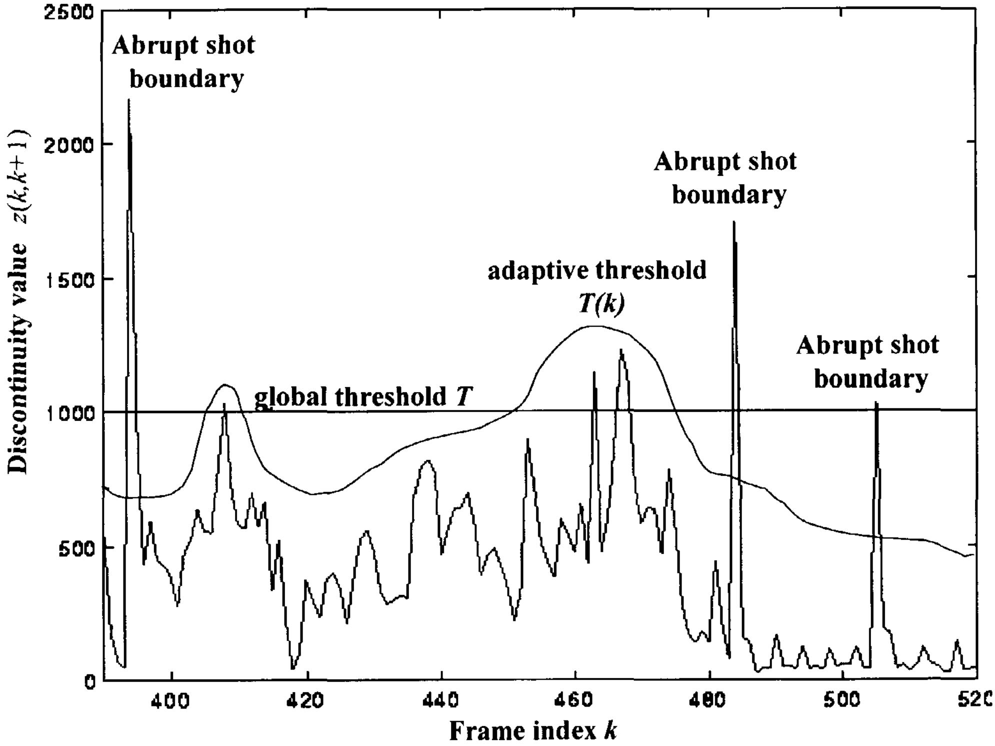


Figure 2.5: *Improved detection performance when using an adaptive threshold function T(k) instead of a global threshold T.*

However, even if they can be specified in a non-heuristic way, as shown by (2.2.16), the crucial problem related to the global threshold still remains, as illustrated in Figure 2.5. If the prespecified global threshold is too low, many false detections will appear in the shot, where high discontinuity values are caused by extreme factors, as defined in Section 2.1. If the threshold is made higher to avoid falsely detected boundaries, then the high discontinuity value corresponding to the shot boundary close to frame 500 will not be detected.

A much better alternative is to work with adaptive thresholds, i.e. with thresholds computed locally. The improved detection performance that results from using adaptive threshold function *T(k)* instead of the global threshold *T* is also illustrated in Figure 2.5. If the value of the function *T(k)* is computed at each frame *k* based on the extra information embedded in the detector (Figure 2.4), high discontinuity values computed within shots can be distinguished from those computed at shot boundaries. Three detection approaches applying adaptive thresholds can be found in recent literature.

A method for detecting abrupt shot boundaries using an adaptive threshold is presented in [Yeo95a]. There, the values *T(k)* are computed using the information about the temporal pattern that is characteristic for abrupt boundaries. The authors compute the discontinuity values with the interframe distance *l*=1. As shown in Figure 2.6, the *N* last computed consecutive discontinuity values are considered, forming a sliding window. The presence of a shot boundary is checked at each window position, in the middle of the window, according to the following criterion:

$$\begin{aligned} &\mathit{if} \quad z(k,k+1) = \max_{i=-\frac{N}{2},\dots,\frac{N}{2}} \left(\forall z(k+i,k+1+i)\right) \wedge z(k,k+1) \geq \alpha\, z_{sm} \\ &\Rightarrow \quad \textit{abrupt shot boundary} \end{aligned} \tag{2.2.17}$$

In other words, an abrupt shot boundary is detected between frames *k* and *k*+1 if the discontinuity value *z*(*k*,*k*+1) is the window maximum and α times larger than the second largest discontinuity value z_{sm} within the window. The parameter α can be understood as the *shape parameter* of the boundary pattern. This pattern is characterized by an isolated sharp peak in a series of discontinuity values. Applying (2.2.17) to such a series at each position of a sliding window is nothing else than matching the ideal pattern shape and the actual behavior of discontinuity values found within the window. The major weakness of this approach is the heuristically chosen and fixed parameter α. Because α is fixed, the detection procedure is too coarse and too inflexible, and because it is chosen heuristically, one cannot make statements about the scope of its validity.

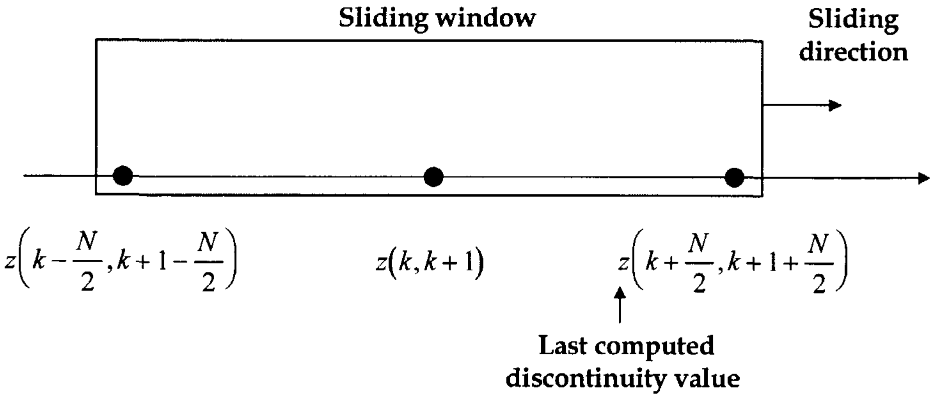

Sliding window
Sliding direction
$z\left(k-\frac{N}{2},k+1-\frac{N}{2}\right)$
$z(k,k+1)$
$z\left(k+\frac{N}{2},k+1+\frac{N}{2}\right)$
Last computed discontinuity value

Figure 2.6: *Illustration of a sliding window approach from [Yeo95a]*

In order to make the threshold specification in [Yeo95a] less heuristic, a detection approach was proposed in [Han97a] and [Han97b], which combines the sliding window methodology with the Gaussian distribution of discontinuity values proposed in [Zha93]. Instead of choosing the form parameter α heuristically, this parameter is determined indirectly, based on the prespecified tolerable probability for falsely detected boundaries. Zhang et al. observe in [Zha93] that the discontinuity values (there obtained by comparing color code histograms) can be regarded as a realization of an uncorrelated Gaussian process if no shot change or motion is present. This observation is extended in [Han97b] to any other temporal segment with a *uniform* content development, independent of the present amount of action. Within a single shot, the series of discontinuity values can then be modeled either as a single uncorrelated Gaussian process or as a temporal concatenation of multiple uncorrelated Gaussian processes. Shots themselves are separated by individual large-valued outliers, or peaks. Based on this a statistical model for the discontinuity values is defined that has the following properties:

- Each discontinuity value measured along a sequence can be assigned one state of a two-state model: the state "S" when it is within a Gaussian shot segment, and the state "D" when it is computed at shot boundaries. A state "S" can be followed by another state "S" or by a state "D". State "D" is always followed by state "S";

- Each state "S" has three parameters, determining the process that generates the discontinuity value $z(k,k+1)$ in that state, namely: the duration of the state L, the mean and the variance of the corresponding Gaussian process;

- State "D" has duration 1.

Figure 2.7 shows the defined statistical model of a fictive series of discontinuity values, with each Gaussian segment "S" represented by its mean value. The detection procedure is activated only if the discontinuity value in the middle of the sliding window is the window maximum. As shown in Figure 2.8a, it is assumed that the series of discontinuity values captured by the window and lying at each side of the window maximum can be described by one and the same Gaussian probability density function. We define these functions as $p_{left}(z,k)$ and $p_{right}(z,k)$. The new threshold value $T(k)$, illustrated in Figure 2.8b together with the defined Gaussian distributions, is computed as the solution of the following integral equation:

$$P=\frac{1}{2}\int_{T(k)}^{\infty}\left(p_{left}(z,k)+p_{right}(z,k)\right)dz \qquad (2.2.18)$$

Here, P is the given tolerable probability for falsely detected boundaries. As in [Zha93], the rate of missed detections cannot be regulated, since the distribution of discontinuity values measured on boundaries is not taken into account. Note that the form parameter α is "hidden" in the computed threshold value $T(k)$.

One way in which the additional information embedded in the detector can influence the process of shot-boundary detection much more effectively is using the *statistical detection theory*. One of the first applications of the statistical detection theory to signal analysis can be traced back to the work of Curran [Cur65]. A characteristic example of recent works in this area can be found in [Vas98]. There, the proposed statistical method for detecting abrupt shot boundaries includes the *a priori* information based on shot-length distributions, which can be assumed consistent for a wide range of sequences. However, this *a priori* information is the only type of information embedded in the detector, and is, by itself not sufficient to prevent false detections caused by extreme factors. A more robust statistical framework for shot-boundary detection is presented in Section 2.3 of this chapter.

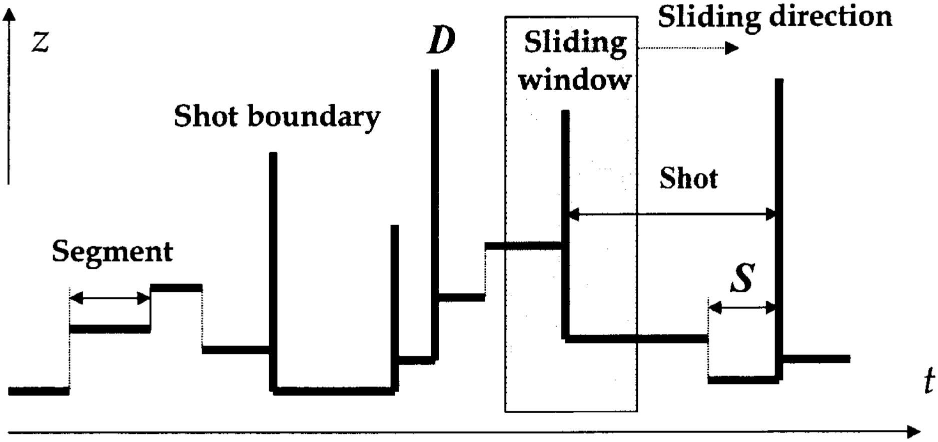
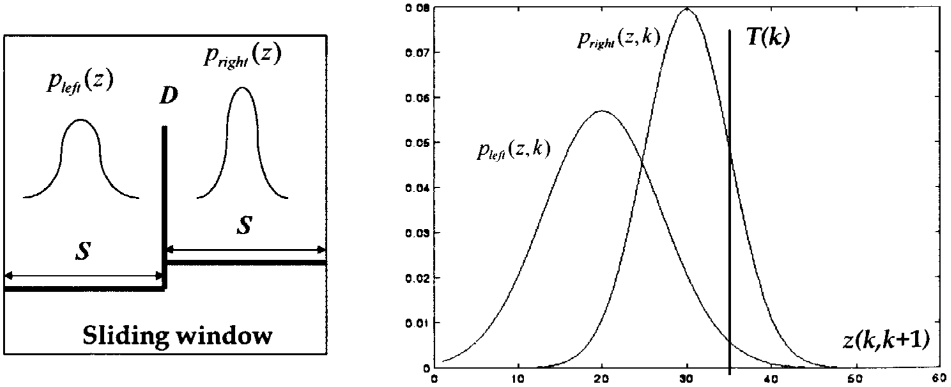


Figure 2.7: *Temporal segment structure of the series of consecutive discontinuity values computed along a sequence*



(a) (b)

Figure 2.8: *Moment situation within a sliding window. (a) A "D" state in the middle of the window surrounded by unbroken segments of "S" states, each of them described by one and the same Gaussian distribution. (b) The threshold T(k) together with Gaussian probability density functions of discontinuity values on both sides of the window maximum*

Different types of shot boundaries

Different boundary types were considered in most of the approaches presented in recent literature, although the emphasis was mostly put on the detection of abrupt boundaries. This preference can be explained by the fact that there is no strictly defined behavior for discontinuity values around and within gradual transitions. While the abrupt boundaries are always represented by an isolated high discontinuity value, the behavior of these values around and within a gradual transition is not unique, not even for one and the same type of transition. In the following we will present some recent approaches to detecting non-abrupt boundaries.

One of the first attempts for detecting non-abrupt boundaries can be found in [Zha93], where a so-called twin-comparison approach is described. The method requires two thresholds, a higher one, T_h, for detecting abrupt boundaries, and a lower one, T_l, for detecting gradual transitions. First the threshold T_h is used to detect high discontinuity values corresponding to abrupt boundaries, and then the threshold T_l is applied to the rest of the discontinuity values. If a discontinuity value is higher than T_l, it is considered to be the start of a gradual transition. At that point, the summation of consecutive discontinuity values starts and goes on until the cumulative sum exceeds the threshold T_h. Then, the end of the gradual transition is set at the last discontinuity value included in the sum.

In [Ham94], a model-driven approach to shot-boundary detection can be found. There, different types of shot boundaries are considered to be editing effects, and are modeled based on the video production process. Especially for dissolves and fades, different chromatic scaling models are defined. Based on these models feature detectors are designed and used in a feature-based classification approach to segment the video. The described approach takes into account all types of shot boundaries defined by the models.

One further method for detecting gradual transitions can be found in [Men95], which investigates the temporal behavior of the variance of the frame pixels. Since within a dissolve different visual material is mixed, it can be assumed that frames within a dissolve loose their sharpness. This can be observed in the temporal behavior of the frame variance, which starts to decrease at the beginning of the transition, reaches its minimum in the middle of the transition and then starts to increase again. A characteristic parabolic pattern of variance behavior is reported. The detection of the transition is then reduced to detecting the parabolic curve pattern in a series of measured variances. In order to be recognized as a dissolve, the potential pattern has to have a width and the depth that exceeds the prespecified thresholds.

In [Son98], a chromatic video edit model for gradual transitions is built based on the assumption that discontinuity values belonging to such a transition form a pattern consisting of two piece-wise linear functions of time; one decreasing and one increasing. Such linearity does not apply outside the transition area. Therefore, the authors search for close-to-linear segments in the series of discontinuity values by investigating the first and the second derivative of the slope in time. A close-to-linear segment is found if the second derivative is less than a prespecified percentage of the first derivative.

Although each of the described models is reported to perform well in most cases, strong assumptions are made about the behavior of discontinuity values within a transition. Furthermore, several (threshold) parameters need to be set heuristically. The fact that patterns which are formed by consecutive discontinuity values and correspond to a gradual transition can strongly vary over different sequences still makes the detection of gradual transitions an open research issue [Lie99].

2.3 A robust statistical framework for shot-boundary detection

In this section we develop the statistical framework for shot-boundary detection, which is in accordance to the scheme in Figure 2.4. In contrast to detection methodologies we discussed earlier, our statistical framework includes all aspects discussed until now relevant for maximum detection performance:

- In order to provide a high level of discrimination between ranges $\overline{R}$ and R, we compute the discontinuity values using motion compensating features and metrics.

- We use both the information on temporal boundary patterns and on shot-length distributions in the detector to compute the adaptive threshold $T(k)$. Here we apply the sliding window methodology and compute the threshold value at each window position.

- We apply statistical detection theory to build a robust boundary-detection framework. This theory provides means to effectively embed the extra information from the previous item and compute the threshold value *T(k)* using the criterion that the average probability for detection mistakes must be minimized.

In terms of the statistical detection theory, shot-boundary detection can be formulated as the problem of deciding between the two hypotheses:

- S - boundary present between frames k and $k+l$
- $\overline{S}$ - no boundary present between frames k and $k+l$

In order to take into account the information about temporal boundary patterns, we consider the N last computed consecutive discontinuity values together, in this way forming a sliding window. We define the vector $\underline{z}(k)$ as

$$\underline{z}(k)=\left\langle z(k-i,k+l-i),\quad i=-\frac{N}{2},\ldots,\frac{N}{2}\right\rangle \tag{2.3.1}$$

We also define the *likelihood functions* $p(\underline{z}|S)$ and $p(\underline{z}|\overline{S})$, which indicate at which degree an arbitrary series of discontinuity values $\underline{z}(k)$, defined by (2.3.1), belongs to series not containing any shot boundary and those containing a shot boundary, respectively, that is

$$p(\underline{z}(k)\,|\,S)=p\left(z\left(k-\frac{N}{2},k+l-\frac{N}{2}\right),\ldots,z\left(k+\frac{N}{2},k+l+\frac{N}{2}\right)\middle|\,S\right)$$

and (2.3.2)

$$p(\underline{z}(k)\,|\,\overline{S})=p\left(z\left(k-\frac{N}{2},k+l-\frac{N}{2}\right),\ldots,z\left(k+\frac{N}{2},k+l+\frac{N}{2}\right)\middle|\,\overline{S}\right)$$

In terms of statistical detection theory, the defined likelihood functions can be considered analogous to previously used ranges of discontinuity values $\overline{R}$ and R. Consequently, the requirements for a good discrimination between ranges can now be transferred to the likelihood functions $p(\underline{z}|S)$ and $p(\underline{z}|\overline{S})$. Further, we define the *a priori probability function P(S,k)*, which defines the probability that there is a boundary between frames k and $k+l$ based on the number of frames elapsed since the last detected shot boundary. As the criterion for deriving the rule for deciding between the two hypotheses, we choose minimizing the average probability for detection mistakes, given as

$$P_e(k) = \left(1 - P(S,k)\right) \int\limits_{\underline{Z}_S} p\left(\underline{z}(k) \mid \bar{S}\right) d\underline{z}(k) + P(S,k) \int\limits_{\underline{Z}_{\bar{S}}} p\left(\underline{z}(k) \mid S\right) d\underline{z}(k) \tag{2.3.3}$$

Minimization of (2.3.3) provides the following decision rule at the frame k:

$$\frac{p\left(\underline{z}(k) \mid S\right)}{p\left(\underline{z}(k) \mid \bar{S}\right)} \underset{S}{\overset{\bar{S}}{\lessgtr}} \frac{1 - P(S,k)}{P(S,k)} \tag{2.3.4}$$

which can be transformed into

$$f\left(z\left(k - \frac{N}{2}, k + l - \frac{N}{2}\right), \ldots, z\left(k + \frac{N}{2}, k + l + \frac{N}{2}\right)\right) \underset{S}{\overset{\bar{S}}{\lessgtr}} T(k) \tag{2.3.5}$$

We call $\underline{Z}_S$ and $\underline{Z}_{\bar{S}}$ the *discontinuity-value domains* belonging to the two hypotheses. The domain $\underline{Z}_{\bar{S}}$ contains all vectors $\underline{z}(k)$, for which the hypothesis $\bar{S}$ is chosen in (2.3.5), and vice versa. However, the N-dimensional likelihood functions (2.3.2) are difficult to compute. Therefore, we simplify the shot-boundary detector (2.3.4) in several respects, under the condition that the detection performance is not degraded:

- We keep the sliding-window concept, but use only the scalar likelihood functions $p(z|S)$ and $p(z|\bar{S})$ evaluated for the discontinuity value $z(k,k+l)$ lying in the middle of the window.

- Instead of capturing the dependencies between elements of the vector $\underline{z}(k)$ via their mutual likelihood functions $p(\underline{z}|S)$ and $p(\underline{z}|\bar{S})$, we pursue the following procedure. We first investigate the temporal pattern belonging to a certain boundary type. Each of these patterns is characterized by specific relationships among discontinuity values. A typical example is an isolated peak of an abrupt shot boundary, which can be fully captured by finding the ratio between the maximal and the second largest value in a discontinuity value series. The higher this ratio, the more probable is the presence of an abrupt shot boundary at the place of the maximal discontinuity value. The ratio between the maximal and the second largest discontinuity value can now be defined as *pattern-matching indication (PMI),* i.e. an indication that the pattern formed by consecutive discontinuity values is similar to the one that is characteristic for a certain boundary type, and therefore also as an indication of having a boundary of a certain type between frames *k* and *k+l*. Thus the PMI can be defined for any arbitrary type of shot boundary by the following generalized function:

$$\psi(k,k+l)=F\left(z\left(k-\frac{N}{2},k+l-\frac{N}{2}\right),..,z\left(k+\frac{N}{2},k+l+\frac{N}{2}\right)\right) \tag{2.3.6}$$

- At last, we define the conditional probability function $P_{Patt}(\psi(k,k+l)\,|\,S)$, which is the probability of having a shot boundary between frames *k* and *k+l*, based on matching of temporal patterns. It is computed at each window position, and serves as the modifier for the *a priori* probability *P(S,k)*. The lower the indication $\psi(k,k+l)$, the less likely is the presence of a shot boundary between frames and the lower are the values of $P_{Patt}(\psi(k,k+l)\,|\,S)$. In such cases the *a priori* probability is modified downwards. This modification becomes crucial if the *a priori* probability and the likelihood functions are in favor of the hypothesis *S*, whereby $\overline{S}$ is the proper hypothesis. In this way, boundaries detected falsely due to extreme factors can be eliminated. On the other hand, large values $\psi(k,k+l)$ indicate a similarity between the pattern formed by the elements of the vector $\underline{z}(k)$ and the pattern of a shot boundary. In such cases the probability that high discontinuity values are caused by extreme factors is small and the correction of the *a priori* probability by $P_{Patt}(\psi(k,k+l)\,|\,S)$ is not necessary.

On the basis of the simplifications described above, the general vector detection rule (2.3.4) has been now reduced to the scalar rule (2.3.7):

$$\frac{p(z(k,k+l)\,|\,S)}{p\left(z(k,k+l)\,|\,\overline{S}\right)}\ \underset{S}{\overset{\overline{S}}{\lesseqgtr}}\ \frac{1-P(S,k)P_{patt}(\psi(k,k+l)\,|\,S)}{P(S,k)P_{patt}(\psi(k,k+l)\,|\,S)} \tag{2.3.7}$$

Since a different function (2.3.6) is required for each boundary type, we cannot use one generalized detector (2.3.4) for detecting all shot boundaries, but need separate scalar detectors (2.3.7) operating in parallel, each being used for one specific type of shot boundary.

In Section 2.4 we develop the detector (2.3.7) for *abrupt* shot boundaries. We start with the computation of discontinuity values based on suitable features and metrics. This is followed by the definition of the *a priori* probability function $P(S,k)$ and by finding the scalar likelihood functions $p(z|S)$ and $p(z|\bar{S})$. At last, PMI function $\psi(k,k+l)$ and the conditional probability function $P_{Patt}(\psi(k,k+l)\,|\,S)$ are defined.

2.4 Detector for abrupt shot boundaries

Abrupt shot boundaries take place between two consecutive frames of a sequence. For this reason it is handy to work with discontinuity values, computed with interframe distance l=1.

2.4.1 Features and metrics

In order to maximize the discrimination of likelihood functions $p(z|S)$ and $p(z|\bar{S})$ we compute the discontinuity values by compensating the motion between video frames using a block matching procedure, described in Section 2.2.

Similarly as in [Sha95a], we divide frame k into N_{Blocks} nonoverlapping blocks $b_i(k)$ and search for their corresponding blocks $b_{i,m}(k+1)$ in frame k+1. The block-matching criterion used here is the comparison of average luminance values of blocks $b_i(k)$ and $b_{i,m}(k+1)$, that is

$$D\left(b_i(k), b_{i,j}(k+1)\right) = \left|Y_{ave}\left(b_i(k)\right) - Y_{ave}\left(b_{i,j}(k+1)\right)\right| \qquad (2.3.8)$$

After the corresponding blocks $b_{i,m}(k+1)$ have been found using the formula (2.2.4), we obtain the discontinuity value $z(k,k+1)$ by summarizing the differences between blocks $b_i(k)$ and $b_{i,m}(k+1)$ in view of block-wise average values of all three color components Y_{ave}, U_{ave} and V_{ave}, that is

$$z(k,k+1) = \frac{1}{N_{Blocks}} \sum_{i=1}^{N_{Blocks}} D\left(b_i(k), b_{i,m}(k+1)\right) \qquad (2.3.9)$$

with

$$\begin{aligned} D\big(b_i(k), b_{i,m}(k+1)\big) = & \left|Y_{ave}\big(b_i(k)\big) - Y_{ave}\big(b_{i,m}(k+1)\big)\right| + \\ & \left|U_{ave}\big(b_i(k)\big) - U_{ave}\big(b_{i,m}(k+1)\big)\right| + \\ & \left|V_{ave}\big(b_i(k)\big) - V_{ave}\big(b_{i,m}(k+1)\big)\right| \end{aligned} \tag{2.3.10}$$

2.4.2 A priori probability function

Studies by Salt [Sal73] and Coll [Col76], involving statistical measurements of shot lengths for a large number of motion pictures, have shown that the distribution of shot lengths for all the films considered matches the Poisson function well [Pap84]. Therefore, we integrate the Poisson function to obtain the *a priori* probability for a shot boundary between frames k and k+1, that is

$$P(S,k) = \sum_{w=0}^{\lambda(k)} \frac{\mu^w}{w!} e^{-\mu} \tag{2.3.11}$$

The parameter μ represents the average shot length of a video sequence, w is the frame counter, which is reset each time a shot boundary is detected, and $\lambda(k)$ is the current shot length at the frame k. Although in [Col76] and [Sal73] the Poisson function was obtained for motion pictures, we assume that this conclusion can be extended further to all other types of video programs. However, to compensate for possible variations in program characteristics, we adapt the parameter μ to different program types (movies, documentaries, music video clips, etc.) and sub-types (e.g. an action movie vs. drama). The adjustment of the parameter μ is easy and can be performed automatically, if the program type is known at the input into the video analysis system. In our experiments we kept μ constant at the value 70.

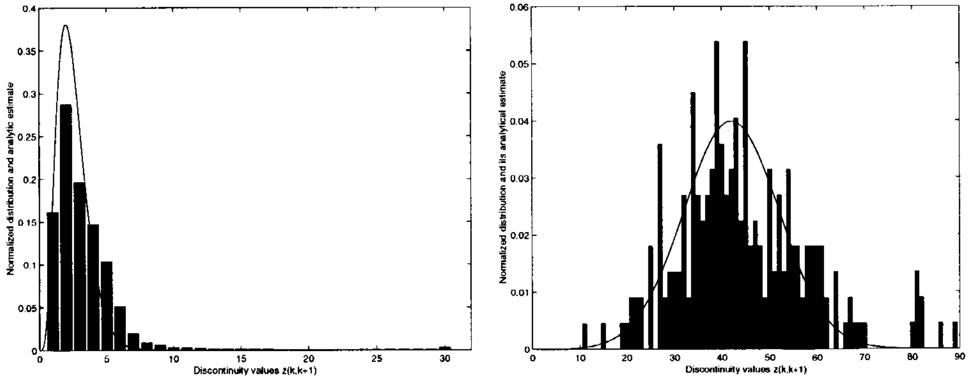
2.4.3 Scalar likelihood functions

We now perform a parametric estimation of scalar likelihood functions $p(z|S)$ and $p(z|\bar{S})$, to be used in the detection rule (2.3.6). In order to get an idea about the most suitable analytical functions used for such estimation, the normalized distributions of discontinuity values $z(k,k+1)$ computed within shots and at shot boundaries are obtained first, using several representative test sequences.

A normalized distribution of discontinuity values computed within shots is shown in Figure 2.9a. The shape of the distribution indicates that a good analytic estimate for this distribution can be found in the family of functions given as

$$p(z|\bar{S}) = h_1 z^{h_2} e^{-h_3 z} \tag{2.3.12}$$

The most suitable parameter combination (h_1, h_2, h_3) is then found experimentally, such that the rate of detection mistakes for the test sequences is minimized. The optimal parameter triplet is found as (1.3, 4, -2). The corresponding analytical function, serving as parametric estimate of the likelihood function $p(z|\bar{S})$, is also shown in Figure 2.9a.



a) *b)*

Figure 2.9: *(a) The normalized distribution of values z(k,k+1) computed within shots (discrete bins) and its analytic estimate (continuous curve), (b) normalized distribution of values z(k,k+1) computed at shot boundaries (discrete bins) and its analytic estimate (continuous curve)*

The analog procedure is applied to obtain the parametric estimate of the likelihood function $p(z|S)$. Figure 2.9b shows the normalized distribution of discontinuity values $z(k,k+1)$, computed at shot boundaries, for which the same set of test sequences as above is used. Judging by the form of the distribution, a Gaussian function

$$p(z|S) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2} \tag{2.3.13}$$

can be taken as a good analytic estimate of it. Again we found the optimal values for the pair of parameters (μ,σ) by experimentally minimizing the rate of detection mistakes for the set of test sequences. This pair of values was obtained as (42, 10), resulting in the Gaussian function presented in Figure 2.9b.

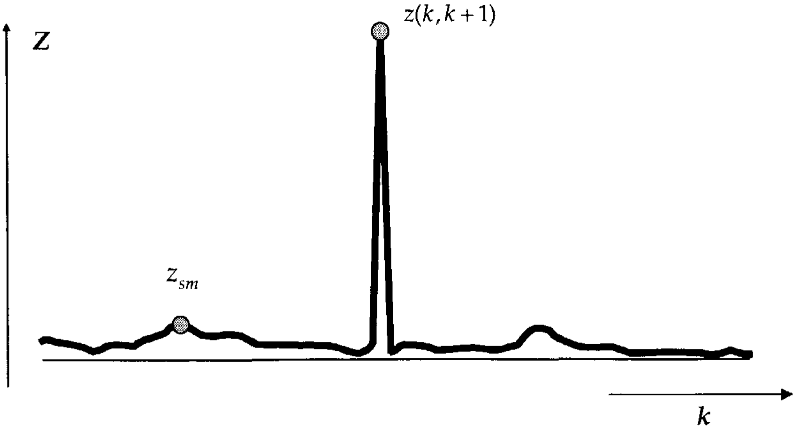


Figure 2.10: *Abrupt boundary pattern with characteristic parameters*

2.4.4 PMI and the conditional probability functions

Based on the discussion in the previous sections, we can state that the presence of an isolated sharp peak belonging to an abrupt shot boundary in the middle of the sliding window can efficiently be described by the ratio of the discontinuity value $z(k,k+1)$ in the middle of the window and the second largest discontinuity value z_{sm} within that window. A typical peak of an abrupt shot boundary with values $z(k,k+1)$ and z_{sm} is illustrated in

Figure 2.10. The corresponding PMI function to be used in the detector (2.3.7) is now given as

$$\psi(k,k+1)=\frac{z(k,k+1)}{z_{sm}} \tag{2.3.14}$$

The value of the PMI function (2.3.14) serves as the argument of the conditional probability function $P_{patt}(\psi(k,k+1)\,|\,S)$, defined as

$$P_{patt}(\psi(k,k+1)\,|\,S)=\frac{1}{2}\left(1+erf\left(\frac{\psi(k,k+1)-d}{\sigma_{erf}}\right)\right) \tag{2.3.15}$$

with

$$erf(x)=\frac{2}{\pi}\int_0^x e^{-t^2}dt \tag{2.3.16}$$

The parameters d and σ_{erf} are the "delay" from the origin and the spreading factor determining the steepness of the middle curve segment, respectively. The optimal parameter combination (d,σ_{erf}) is found experimentally such that the detection performance for the test sequences is optimized. The resulting optimal pair of parameters was found as (13, 5). The conditional probability (2.3.15) is illustrated in Figure 2.11.

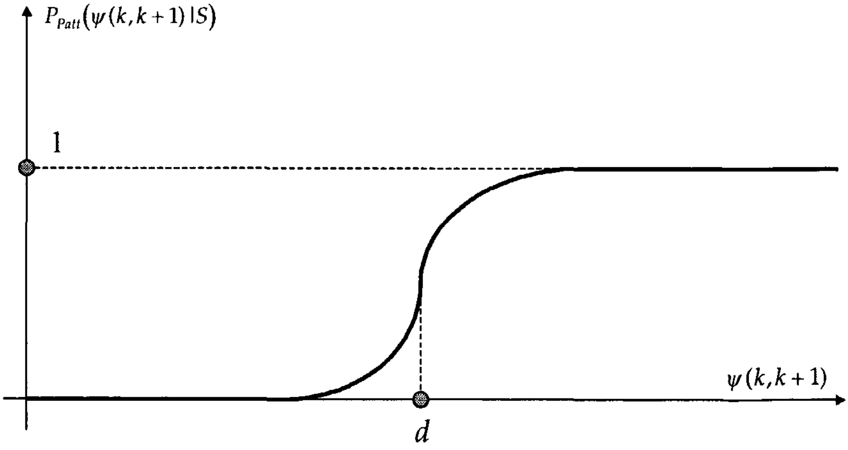


Figure 2.11: *The conditional probability function* $P_{patt}(\psi(k,k+1)\,|\,S)$

2.4.5 Experimental validation

Achieving a high detection performance was an important issue when we developed the statistical detection framework. To test the performance of the detector (2.3.7) for abrupt boundaries, we used 5 sequences that belong to 2 different categories of programs, movies and documentaries, and that were not previously employed for training the detection procedure. The results presented in Table 2.1 illustrate a high detection rate and no falsely detected boundaries. Furthermore, the obtained good results remain consistent over all sequences.

Test material	Length in frames	Total	Detected boundaries	Falsely detected boundaries
Documentary 1	700	3	3	0
Documentary 2	800	5	5	0
Documentary 3	900	6	6	0
Movie 1	10590	90	90	0
Movie 2	17400	95	94	0
Total	30390	199	198	0

Table 2.1: *Detection results for abrupt shot boundaries*

2.5 Discussion

Most existing approaches for shot boundary detection are based on explicitly given thresholds or relevant threshold parameters, which directly determine the detection performance. Due to such a direct mutual dependence, the detection performance is highly sensitive to specified parameter values. For instance, a threshold set to 2.3 will interpret a discontinuity value 2.31 as a shot boundary and a value 2.29 as a regular value within a shot. Beside the sensitivity, the problem of specifying such a precise threshold remains. And, consequently, the scope of the validity of such a precise threshold is highly questionable.

Manual parameter specification clearly cannot be avoided in any of the detection approaches. However, the influence of these parameters on the detection performance can be diminished and the detection can be made more robust if the parameters are used at lower levels of the detection framework, so only for the purpose of globally defining the framework components. Each component then provides the detector with nothing more than an indication of the presence of a boundary based on a specific criterion. The decision making about the presence of a shot boundary is then left solely to the detector, where all the indications coming from different sources are evaluated and combined. In this way, the importance of a single manually specified parameter is not as great as when that parameter is directly a threshold, and can therefore be assumed valid in a considerably broader scope of sequences. In the statistical detection framework presented in this chapter, this is the case with parameter sets (h_1, h_2, h_3) and (μ, σ), which are used to define the likelihood functions (2.3.12) and (2.3.13), as well as with parameters d and σ_{erf} used to formulate the conditional probability function (2.3.15).

The only parameter which needs to be adjusted depending on the type of sequence is μ, which is used in the formula (2.3.11) to define the *a priori* probability. However, setting the value for μ is easy, since it determines the average shot length characteristic for a certain program type. For instance, the μ value for movies can be set to a value within a range 80-100, and for music TV clips to 30-40. The adjustment of the μ value can be performed fully automatically if the program type information is available in the shot-boundary detection system. An example is a video analysis system, as illustrated in Figure 1.2, which operates directly on DVB streams. Here, each transmitted program compliant to DVB standard also contains a header, which – among other data – contains the program type (movie, documentary, music TV clip, etc.). Therefore, μ can be set easily by means of a simple look-up table.

Since the parameters used in our framework can either be assumed generally valid or be adjusted automatically, no human supervision is required during the detection procedure. At the same time, since the parameters are optimized for a general case, similar high detection performance can be expected for any input sequence. Both of these aspects make the developed framework suitable for an implementation in a fully automated sequence analysis system. The facts that the detection method presented in this chapter can operate on a wide range of video sequences without human supervision, and keep the constant high detection quality for each of them, are the major advantages the proposed detection framework has over the methods from recent literature.

Chapter 3

Automatically Abstracting Video using Key Frames

3.1 Introduction

A structured collection of selected video frames, or *key frames*, is a compact representation of a video sequence and is useful for various applications on a video. For instance, it can provide a quick overview of the video-database content, enable access to shots, episodes and entire programs in video-browsing and retrieval systems and be used for making a commercial for a video. Furthermore, a video index may be constructed based on visual features of key frames, and queries by example may be directed at key frames using image-retrieval techniques [Zha97a]. Also the higher-level video processing and analysis steps involving comparisons of shots can benefit from visual features captured in key frames [Yeu95a], [Yeu97], [Han99b]. To enable these applications, key frames can be extracted in various fashions, such as

- **Extracting the *most memorable* video frames**: It is in human nature to remember some most memorable segments of a video, e.g. a zoom of an actor in a funny pose, a slow camera pan along a beautiful landscape or an impressive action scene. A number of key frames can be extracted to represent each of these segments.

- **Summarizing the visual content of a video**: The visual content of a video can be "compressed" by first collecting fragments showing all of its relevant elements, such as landscapes, objects, persons, situations, etc., and by then searching for a limited number of frames to represent each of these elements. An alternative summarizing approach is to investigate the story flow of a video sequence and to represent each successive logical segment (event, episode) by suitable frames. Then, by concatenating these frames chronologically, a *storyboard* can be obtained giving a compact video overview [Pen94a].

Key frames can be extracted manually or automatically. Both possibilities are illustrated in Figure 3.1. If key frames are extracted manually, they comply with human cognition, that is, human understanding of a video content and human perception of representativeness and technical quality of a frame. For instance, each key frame can be extracted based on the role the persons and objects captured therein play in the context of the target application. From several candidate frames, the one being most representative (e.g. taken under the best camera angle) is chosen. Furthermore, it is expected that no blurred or "dark" frames are extracted, or those with coding artifacts, interlacing effects, etc.

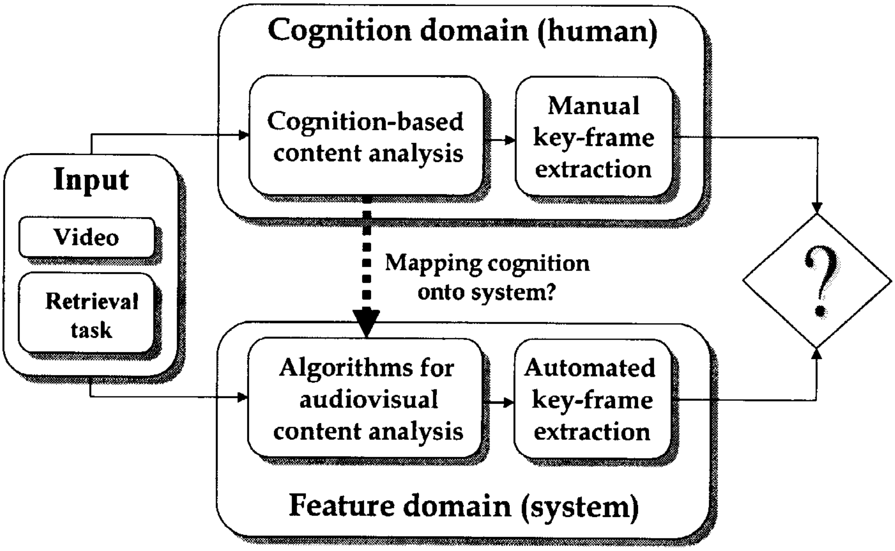

Cognition domain (human)
Cognition-based content analysis
Manual key-frame extraction
Input
Video
Retrieval task
Mapping cognition onto system?
?
Algorithms for audiovisual content analysis
Automated key-frame extraction
Feature domain (system)

Figure 3.1: *Manual vs. automated key-frame extraction*

In order to develop feature-based algorithms for automatically extracting key frames that have the same quality as those extracted manually, we must map the extraction criteria complying with human cognition onto the *machine criteria*. However, such mapping is highly problematic, not only technically (Chapter 1), but also due to the missing *ground truth* for the key-frame extraction.

If several users manually extract key frames from one and the same video and for the same target application, it can realistically be assumed that each of the obtained sets will be unique, concerning both the total number of frames contained therein and the specific frames extracted. One reason for this is the subjectivity of human perception of a video content. Especially when choosing the most memorable video segments and extracting the corresponding key frames are concerned, the dispersion among extraction results obtained by different users will be high [Paa97]. However, even if there is a consensus among users about which segments should be represented in the visual abstract, again different key-frame sets can be expected. A trivial example is a stationary shot showing an anchorperson in a news program. Such a shot can equally well be represented by any of its frames.

Based on the discussion above we conclude that automatically extracting key frames for the purpose of capturing the most memorable moments of a video sequence is a difficult problem, mainly due to the subjectivity of the definition what is *memorable*. Compared to this, the role of subjectivity in extracting key frames for making a visual summary of a video is significantly smaller. This can be explained by the fact that such a summary ideally contains *all* relevant visual-content elements (faces, objects, landscapes, situations, etc.) and not a subjective selection of these elements. In this way, we understand the key-frame based video summary as a unity of all possible subjective key-frame selections. This makes the extraction of "summarizing" key frames easier to automate. The only aspect which remains subjective and therefore difficult to take into account by automation is to choose a representative frame out of several equally acceptable candidate frames. However, as illustrated in the example that involves a stationary anchorperson shot, selecting any of the candidate frames does not considerably influence the quality of the resulting key-frame set. For this reason, instead of considering the possibility of selecting any frame out of equally acceptable candidates as a problem for automation, we hold that it is an additional degree of freedom in the automation of the key-frame extraction procedure.

We now define the objective of this chapter so as to provide methods for automatically extracting key frames which summarize the visual content of a video. Since the complex extraction criteria related to human cognition are difficult to map onto the system level, we circumvent this mapping by applying a practical extraction methodology which is based on reducing the visual-content redundancy among video frames. In the following, we define and discuss three different groups of key-frame extraction techniques belonging to this methodology: *sequential extraction in a local context* (SELC), *sequential extraction in a global context* (SEGC) and *non-sequential extraction* (NSE).

A typical video can be seen as a concatenation of frame series, each characterized by a high visual-content redundancy. These frame series can be entire video shots or shot segments. Then, the redundancy of the visual content found in such series can be reduced by representing each of them by one key frame. Taking again as an example a stationary shot showing an anchorperson in a news program, the frames of such a shot are almost identical and can be compressed to a single frame. Applied to the entire video sequence, key frames can then be seen as its (non)equally distributed sample frames [Pen94a]. This we call *sequential extraction in a local context* (SELC).

Since a SELC technique extracts key frames only in the local context, similar key frames may be extracted from different (remote) sequence fragments, which results in a redundancy within the obtained key-frame set. This indicates that by using some alternative techniques one can further reduce the number of extracted key frames while still keeping all the relevant visual information of a sequence. One of possibilities is to modify SELC approaches by taking into account all previously extracted key frames each time a new frame is considered. Then, a new key frame is extracted only if it is considerably different from all other already extracted key frames. We call such a technique *sequential extraction in a global context* (SEGC). Another alternative is a *non-sequential extraction* (NSE), where all frames of a sequence are taken and grouped together, based on the similarity of their visual content. The key-frame set is then obtained by collecting representatives of each of the groups.

If concatenated, the key frames obtained by means of a SELC technique represent a "red line" through the story of a video and closely provide a *storyboard*. However, for some applications involving video content, having a storyboard of that video is not required. This is the case with key-frame based video queries in standard image retrieval tools. In such applications, the redundancy among key frames makes the query database too large, slows down the interaction process and puts larger demands on storage space for keeping the key frames than actually necessary. In these cases, SEGC or NSE techniques are more suitable. While SELC and SEGC techniques allow for on-the-fly (on-line) key-frame extraction and are computationally less expensive than the NSE techniques, the NSE techniques consider the key-frame extraction as a postprocessing step and mostly involve complex clustering procedures. However, a higher complexity of NSE techniques is compensated by the fact that they are more sophisticated and, therefore, provide a higher representativity of key frames while keeping the number of key frames minimal.

After a review of existing approaches to automated key-frame extraction in Section 3.2, we present in Sections 3.3 and 3.4 two novel extraction methods. The first method belongs to the SELC group of approaches and aims at providing a good video summary, also including its storyboard, while keeping the total number of extracted key frames for the entire sequence close to the prespecified maximum. This controllability is, on the one hand, an important practical issue, regarding the available storage space and the interaction speed with a video database, but, one the other hand, it also means an additional constraint that needs to be taken into account during the key-frame extraction procedure. In contrast to the method in Section 3.3, the major objective of the method presented in Section 3.4 is minimizing the redundancy among video frames and providing a set of key frames which is similar to the one based on human cognition for a given video sequence. We can explain this objective with the example of a simple dialog sequence, where stationary shots of each of the two characters participating in a dialog are alternated. Since a user would summarize such a sequence by taking only two frames, one for each of the characters, this should be obtained automatically as well. The approach in Section 3.4 belongs to the NSE group; it is based on cluster validity analysis and is designed to work without any human supervision. A discussion to this chapter can be found in Section 3.5.

3.2 Previous work on key-frame extraction

A number of methods for automating the key-frame extraction procedure can be found in recent literature. As will be shown in this section, some of the methods are based on the criterion of reducing the visual-content redundancy among consecutive frames, as defined above. However, some characteristic key-frame extraction methods based on other criteria will be described as well.

A first attempt to automate key-frame extraction was done by choosing as a key frame the frame appearing after each detected shot boundary [Sha95b]. However, while one key frame is sufficient for stationary shots, in dynamic sequences it does not provide an acceptable representation of the visual content. Therefore, methods were needed to extract key frames that are in agreement with the visual-content variations along a video sequence. One of the first key-frame extraction approaches developed in view of this objective is presented in [Zha95a], with all details given in [Zha97b]. Key frames are extracted in a SELC fashion separately for each shot. The first frame of a shot is always chosen as a key frame. Then, similar methodology is applied as for detecting shot boundaries. The discontinuity value $z(F_{last},k)$ is computed between the current frame k of a sequence and the last extracted key frame F_{last} using color histograms as spatial features (Chapter 2). If this discontinuity value exceeds a given threshold T, the current frame is selected as a new key frame, that is

$$\begin{aligned} &Step\ 1\text{:}\quad F_{last} = 1 \\ &Step\ 2\text{:}\quad \forall k \in [2,S] \quad if\ z(F_{last},k) > T \Rightarrow F_{last} = k \end{aligned} \tag{3.2.1}$$

Here, S is the number of frames within a shot. The extraction procedure (3.2.1) is then adapted by means of the information on dominant or global motion resulting from camera operations and large moving objects, according to a set of rules. For a zooming-like shot, at least two frames will be extracted, at the beginning and at the end of a zoom. The first frame represents a global and the other one a more detailed view of a scene. In case of panning, tilting and tracking, the number of frames to be selected depends on the rate of visual-content variation: ideally, the visual content covered by each key frame has little overlap, or each frame should capture different object activities. Usually frames that have less than 30% overlap in their visual content are selected as key frames. A key-frame extraction method similar to (3.2.1) can also be found in [Yeu95a]. There, however, the motion information is not used.

Another SELC extraction approach is proposed in [Gun98], where the authors first compute the discontinuity value between the current frame k and the N previous frames. This is done by comparing the color histogram of the frame k and the average color histogram of the previous N frames, that is

$$z\left(k,\{k-1,..,k-N\}\right) = \sum_{j=k-1}^{k-N} \sum_{e=Y,U,V} \left| H_k^e(i) - \frac{1}{N}\sum_{j=k-1}^{k-N} H_j^e(i) \right| \tag{3.2.2}$$

If the discontinuity value (3.2.2) exceeds the prespecified threshold T, the current frame k is extracted as a new key frame F_{last}, i.e.

$$if \quad z(k,\{k-1,..,k-N\}) > T \quad \Rightarrow \quad F_{last} = k \tag{3.2.3}$$

A possible problem with the extraction methods described above is that the first frame of a shot is always chosen as a key frame, as well as those frames lying in shot segments with varying visual content. As discussed in [Gre97], when a frame is chosen that is close to the beginning or end of a shot, it is possible that that frame is part of a dissolve effect at the shot boundary, which strongly reduces its representative quality. The same can be said for frames belonging to shot segments of great camera or object motion (e.g. strong panning or a zoomed object moving close to the camera and hiding most of the frame surface). Such frames may be blurred, and thus in some cases not suitable for extraction. A solution to this problem can be found in [DeM98], where the authors first represent a video sequence as a curve in a high-dimensional feature space. The 13-dimensional feature space is formed by the time coordinate and 3 coordinates of the largest "blobs" (image regions), where 4 intervals (bins) are used for each luminance and chrominance channel. Then the authors simplify the curve using the multidimensional curve-splitting algorithm. The result is, basically, a linearized curve, characterized by "perceptually significant" points, which are connected by straight lines. A key-frame set of a sequence is finally obtained by collecting frames found at perceptually significant points. With a splitting condition that checks the dimensionality of the curve segment that is split, the curve can be recursively simplified at different levels of detail, that is with different densities of perceptually significant points. The final level of detail depends on the prespecified threshold, which evaluates the distance between the curve and its linear approximation. We consider the main problem of this approach to be evaluating the applicability of obtained key frames, as it is not clear which level and objective of video representation is aimed at. For instance, it is unlikely that the objective of the approach is to provide a good video summary, since there is no proof that extracted key frames lying at "perceptually significant points" capture all important aspects of a video. On the other hand, the connection between perceptually significant points and most memorable key frames according to user's cognition is not clear either.

An example of NSE key-frame extraction approaches can be found in [Zhu98]. There, all frames in a video shot are classified into M clusters, where this final number of clusters is determined by a prespecified threshold T. A new frame is assigned to an existing cluster if it is similar enough to the centroid of that cluster. The similarity between the current frame k and a cluster centroid c is computed as the intersection of two-dimensional HS histograms of the HSV color space (H - Hue, S - Saturation, V - Value). If the computed similarity is lower than the prespecified threshold T, a new cluster is formed around the current frame k. In addition, only those clusters that are larger than the average cluster size in a shot are considered as key clusters, and the frame closest to the centroid of a key cluster is extracted as a key frame.

Extraction of key frames in all approaches discussed above is based on threshold specification. The thresholds used in [Zha95a], [DeM98] and [Zhu98] are heuristic, while the authors in [Gun98] work with a threshold they obtained by means of the technique of Otsu [Sah88]. By adjusting the threshold, the total number of extracted key frames can be regulated. However, such regulation can be performed only in a global sense, meaning that a lower threshold will lead to more key frames, and vice versa. An exact or at least an approximate control of the total number of extracted key frames is not possible. First, it is difficult to relate a certain threshold value to the number of extracted key frames. Second, one and the same threshold value can lead to a different number of extracted key frames in different sequences. A practical solution for this problem is to make the threshold more meaningful and to relate it directly to the extraction performance. An example is the threshold specification in form of the maximum tolerable number of key frames for a given sequence. An NSE approach using this sort of thresholds can be found in [Sun97]. There, two thresholds need to be prespecified: r, controlling which frames will be included in the set and N, being the maximum tolerable number of key frames for a sequence. Key frame extraction is performed by means of an iterative partitional-clustering procedure. In the first iteration step, a video sequence is divided into consecutive clusters of the same length L. The difference is computed between the first and the last frame in each cluster. If the difference exceeds the threshold r, all frames of a cluster are taken as key frames. Otherwise, only the first and the last frame of the cluster are taken as key frames. If the total number of extracted frames is equal to or smaller than the tolerable maximum N, the extraction procedure is stopped. If not, a new sequence is composed out of all extracted frames and the same extraction procedure is applied. The biggest disadvantage of this method is the difficulty of specifying the threshold r, since it is not possible to relate the quality of the obtained key-frame set to any specific r value.

If the total number of extracted key frames is regulated by a threshold, the qualities of the resulting key-frame set and of the set obtained for the same sequence but based on human cognition are not necessarily comparable.

For instance, if the threshold is too low, too many key frames are extracted and characterized by a high redundancy of their visual contents. As a result of a threshold set too high, the key-frame set might be too sparse. Especially if the rate of visual-content change allows for only one optimal set of key frames for the best video representation, finding the threshold value providing such a key-frame set is very difficult.

Authors in [Avr98] and [Wol96] aim at avoiding this problem and propose threshold-free methods for extracting key frames. In [Avr98], the temporal behavior of a suitable feature vector is followed along a sequence of frames; a key frame is extracted at each place of the curve where the magnitude of its second derivative reaches the local maximum. A similar approach is presented in [Wol96], where local minima of motion are found. First, the optical flow is computed for each frame and then a simple motion metric is used to evaluate the changes in the optical flow along the sequence. Key frames are then found at places where the metric as a function of time has its local minima. However, although the first prerequisite for finding good key frames was fulfilled by eliminating threshold dependence of the extraction procedure, the two described methods have the same disadvantage as the method proposed in [DeM98], namely an unclear applicability of the resulting key frames.

3.3 Extracting key frames by approximating the curve of visual-content variations

In the key-frame extraction method presented in this section we aim at providing a good video summary while keeping the number of extracted key frames close to the prespecified maximum. This SELC method can be considered as an alternative to the approach from [Sun97]. However, it has the advantage that the number of thresholds is reduced to one; it is the maximum allowed number of key frames N for the entire sequence.

As illustrated in Figure 3.2, key frames are extracted for each shot of a sequence separately. This is done in two major phases. The first phase starts at the beginning of a shot i and lasts until the boundary to the shot i+1 is detected. During this time, the variation of the visual content is modeled along a shot i. The result of this phase is twofold. First, a curve is obtained which models the visual-content variations along shot i. Second, the total magnitude C_i of visual-content variations along a shot i is available at the moment the boundary between shots i and i+1 is detected.

The second phase starts at the moment the boundary to the shot i+1 is detected, and consists of two consecutive steps. In the first step, a fraction K_i of the prespecified N key frames is assigned to shot i, proportional to the computed value C_i, and such that the sum of key frames assigned to all shots of a sequence does not exceed the prespecified maximum N. The number N can be adjusted if we know *a priori* the type of the program to be processed. In the second step, a threshold-free procedure is applied to find optimal positions for the assigned number of key frames along a shot i. Such an optimal distribution is obtained iteratively, by means of a suitable numerical algorithm. In the following subsections, we will describe both extraction phases and all of their steps in more detail.

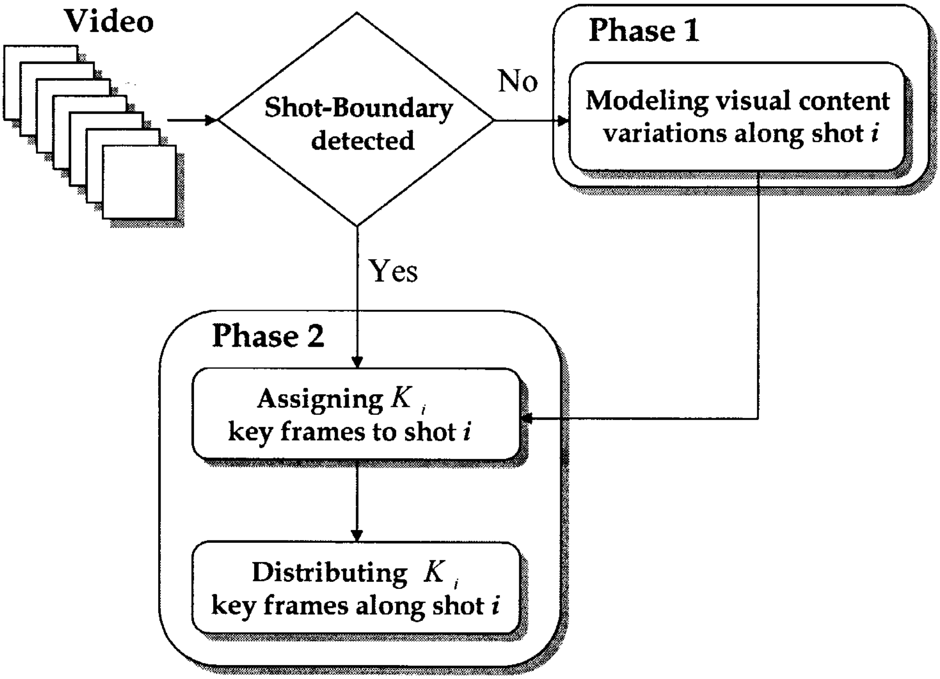

Video
Shot-Boundary detected
No
Phase 1
Modeling visual content variations along shot i
Yes
Phase 2
Assigning K_i key frames to shot i
Distributing K_i key frames along shot i

Figure 3.2: *Scheme of the key-frame extraction approach with controlled number of key frames*

3.3.1 Modeling visual content variations along a shot

In order to model the variations of the visual content along a shot i, we must consider *relevant* content variations, i.e. those that make the extraction of a new key frame necessary. For instance, object motion by a constant background is not as relevant for key-frame extraction as, for instance, camera panning, tilting and tracking. This is because the object motion alone does not result in a drastic change of the visual content, and does not need to be captured by several key frames. Opposed to this, a camera motion constantly introduces new visual material, which needs to be represented by more than one key frame. For efficiently capturing camera operations while excluding the sensitivity of key-frame extraction to object motion, we compute the discontinuity values $z(k,k+1)$ using color histograms and according to (2.2.7), but here in the YUV color space and for $l=1$:

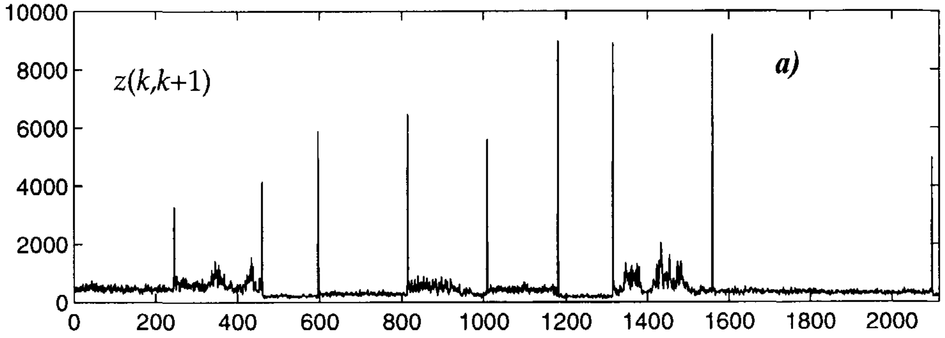
$$z(k,k+1)=\sum_{j=1}^{N_{Bins}}\left(|H_k^Y(j)-H_{k+1}^Y(j)|+|H_k^U(j)-H_{k+1}^U(j)|+|H_k^V(j)-H_{k+1}^V(j)|\right) \tag{3.3.1}$$

Accumulating the discontinuity values (3.3.1) along a shot and taking the current cumulative value at each frame k results in the function $C_i(k)$, which we consider as the model for visual-content variations along a shot i:

$$C_i(k)=\sum_{j=f_{1,i}}^{k-1} z(j,j+1) \tag{3.3.2}$$

The frame $f_{1,i}$ is the first frame of the shot i and the summation process (3.3.2) is reset at the shot boundary. Since $z(k,k+1)$ can only have non-negative values, $C_i(k)$ is a non-decreasing function. It has a close-to-linear behaviour in shot segments with a uniform rate of visual content variations (e.g. a stationary segment or a constant camera motion) and changes in steepness wherever changes in the variation rate occur (e.g. camera motion after a stationary segment). Figure 3.3a shows the discontinuity values computed for nine shots of a typical movie sequence and Figure 3.3b the behavior of the corresponding functions $C_i(k)$. When the end of a shot is reached, we obtain with (3.3.3) the total magnitude of visual content variations along the shot i, with S_i being the number of frames in that shot:

$$C_i=C_i(S_i)=\sum_{j=f_{1,i}}^{S_i-1} z(j,j+1) \tag{3.3.3}$$



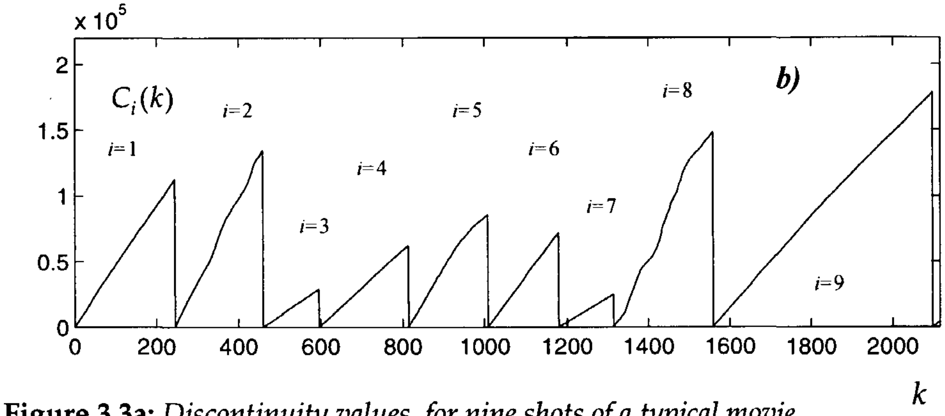


Figure 3.3a: *Discontinuity values for nine shots of a typical movie*
Figure 3.3b: *Functions (3.3.2) modeling the visual content variations*

3.3.2 Distributing N key frames over the sequence

After the total magnitude of visual content variations for the shot i has been obtained by means of (3.3.3), the total prespecified number N of key frames is distributed along all shots of a sequence proportional to values C_i. The higher C_i, the more diverse visual content is assumed in the shot i, which then requires more key frames in order to be represented well. As N_{shots} is the number of shots in the entire sequence, we assign K_i key frames to shot i according to the following ratio:

$$K_i = \frac{C_i}{\sum_{j=1}^{N_{shots}} C_j} N \tag{3.3.4}$$

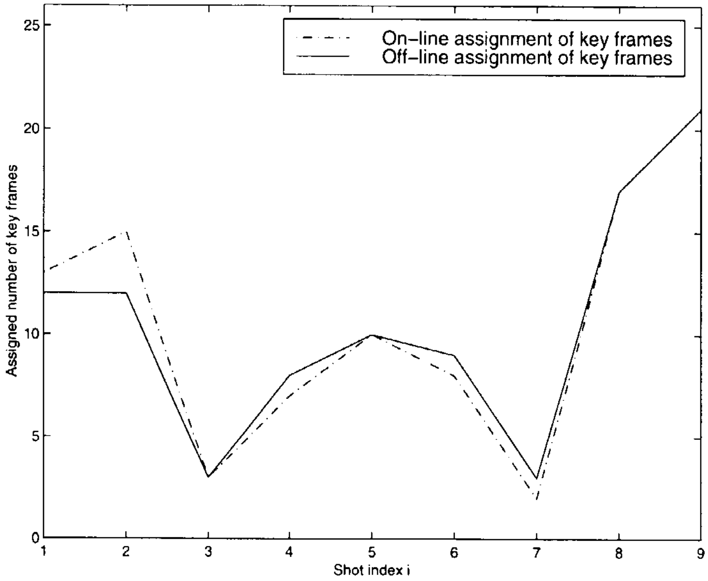


Figure 3.4: *Key-frame assignments according to the procedures (3.3.4) and (3.3.5)*

Equation (3.3.4) assumes that the values C_i are known for all shots of a sequence, so the denominator in (3.3.4) can now be computed. Since, in practice, on-line key-frame extraction is more appealing, we adapt the assignment rule (3.3.4) so that a suitable number of key frames can be assigned to a shot i immediately after the boundary between shots i and i+1 has been detected. The adapted assignment rule is given as follows:

$$K_i = \frac{C_i}{\sum_{j=1}^{N_{shots}} C_j} N = C_i \frac{N}{S} \frac{S}{\sum_{j=1}^{N_{shots}} C_j} \approx \operatorname{int}\left(C_i \frac{N}{S} \frac{\sum_{j=1}^{i} S_j}{\sum_{j=1}^{i} C_j} \right) \tag{3.3.5}$$

Here, S is the total sequence length and S_j is the length of the shot j. Compared to the off-line assignment (3.3.4), the rule (3.3.5) uses only the information available at the moment when K_i is computed. Since the total cumulative variations of the visual content along the entire sequence (denominator in (3.3.4)) is not known, we can only summarize until the shot i. This disadvantage is, however, compensated by taking into consideration also the time parameter, e.g. shot lengths. Thereby we assume that the ratio between the total sequence length and the values C_i for all shots of a sequence can be well approximated by the ratio between these two quantities, where both are only taken up to the current shot *i*.

Assignment results obtained using (3.3.4) and (3.3.5) may differ in the beginning, that is, for a low shot index *i*. However, with increasing *i* we expect the value (3.3.5) to converge towards the value (3.3.4). In order to show this, we chose to distribute an unusually large number of N=100 key frames along nine shots of the sequence illustrated in Figure 3.3a. Assignment results using both methods are presented in Figure 3.4.

3.3.3 Distributing key frames within a shot

In the final step, K_i assigned key frames need to be located within a shot. For the sake of notation and derivation, in the following we will consider *k* in (3.3.2) to be a continuous variable, although a practical implementation will use a discretized version. If we would interpolate $C_i(k)$ for non-integer values of *k* from neighboring values for integer *k*, i.e. $C_i(\lfloor k \rfloor)$ and $C_i(\lceil k \rceil)$, then $C_i(k)$ becomes a non-decreasing function. We will assume this property in the sequel. The underlying theory used here for distributing key frames along a shot is that K_i key frames should be distributed along a shot such that the visual content is summarized in the best possible way. Since the function $C_i(k)$ represents the variations of the visual content along the shot, it also provides the information about the amount of redundancy present in each of the shot segments. The steeper the function, the less redundancy is to be found among consecutive frames in that segment, and vice versa.

Consequently, properly distributing key frames along a shot is equivalent to finding a suitable way of representing the function $C_i(k)$ by K_i (non-) equidistant samples, where the sample density is dependent on function steepness and where each sample is a key frame representing a series of consecutive frames around it. A key frame F_j, j=1,...,K_i, lies in the middle of the interval (t_{j-1}, t_j) and represents all frames in that interval. We approximate the function $C_i(k)$ along this interval by its value at frame $k = F_j$, that is, by $C_i(F_j)$. By doing this for each key frame, a *step curve* is obtained, which closely approximates the function $C_i(k)$ along a shot *i*. Maximizing the quality of such an approximation is now equivalent to properly placing the horizontal line segments and defining their optimal lengths, which is, again, equivalent to properly positioning the key frames F_j in the middle of these segments. To achieve such optimal positioning, we choose to minimize the following L_1 error function:

$$g(F_1,\ldots,F_{K_i},t_1,\ldots,t_{K_{i-1}})=\sum_{j=1}^{K_i}\int_{t_{j-1}}^{t_j}\left|C_i(k)-C_i(F_j)\right|\mathrm{d}k \tag{3.3.6}$$

Note that t_0 and t_{K_i} are the (known) temporal starting and endpoints of the shot i. Figure 3.5 illustrates the meaning of (3.3.6). It shows the function $C_i(k)$ and how the key frames are distributed such that the area between this function and the approximating rectangles, defined by F_j and t_j, is minimized. The minimization of (3.3.6) is carried out in two steps. First, if we assume that the breakpoints t_{j-1} and t_j are given, then the partial integral

$$g(F_j)=\int_{t_{j-1}}^{t_j}\left|C_i(k)-C_i(F_j)\right|\mathrm{d}k \tag{3.3.7}$$

is minimized by taking as key frame the center of the interval considered:

$$F_j=\frac{t_{j-1}+t_j}{2}\quad\text{for}\quad j=1,2,\ldots,K_i \tag{3.3.8}$$

Note that this result is independent of the actual cumulative action function on this interval as long as $C_i(k)$ is a non-decreasing function. After substituting (3.3.8) into (3.3.6), we can minimize the resulting expression with respect to the breakpoints t_j. The resulting solution is given by the following set of K_i equations:

$$C_i(t_j)=\tfrac{1}{2}\left(C_i(F_j)+C_i(F_{j+1})\right)\quad\text{for } j=1,\ldots,K_i \tag{3.3.9}$$

The interpretation of this set of equations is that the breakpoint t_j is chosen such that the value of the function $C_i(k)$ at that breakpoint is the average of the $C_i(k)$ values at the key frames preceding and following that breakpoint. Together, (3.3.8) and (3.3.9) form the solution of the desired key-frame distribution according to the criterion (3.3.6). To solve the key-frame positions from (3.3.8) and (3.3.9), one can employ a recursive search algorithm. To this end, we rewrite (3.3.9) as follows:

$$C_i(F_{j+1}) = 2C_i(t_j) - C_i(F_j) \quad \text{for } j = 1, \ldots, K_i \tag{3.3.10}$$

If we start with assuming a breakpoint t_1, then we can compute key frame F_1 using (3.3.8). From (3.3.10) we can then compute breakpoint t_2 (substitute j=1 in (3.3.10)). Subsequently, from t_2 we can compute F_2 using (3.3.8), from which t_3 follows (substitute j=2 in (3.3.10)). In this way we can recursively compute for the assumed value of t_1 the value of t_{K_i}, which should be identical to the given length of the i-th shot. Depending on the mismatch between the computed and actual value, the position of the breakpoint t_1 can be adjusted. Note that this recursive search procedure is very close to the one often used for designing scalar quantizers [Llo57], [Max60].

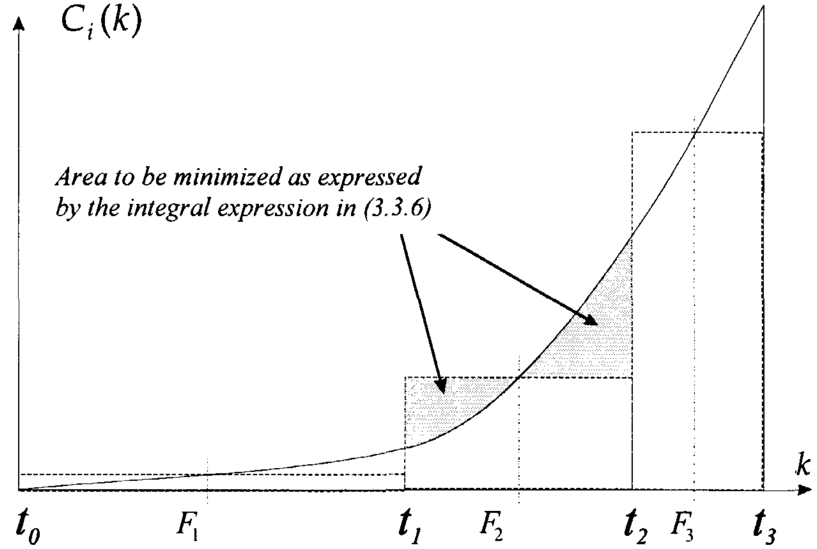

$C_i(k)$
Area to be minimized as expressed by the integral expression in (3.3.6)
k
t_0
F_1
t_1
F_2
t_2
F_3
t_3

Figure 3.5: *Illustration of the function* $C_i(k)$, *the distribution of key frames* F_j *and breakpoints* t_j

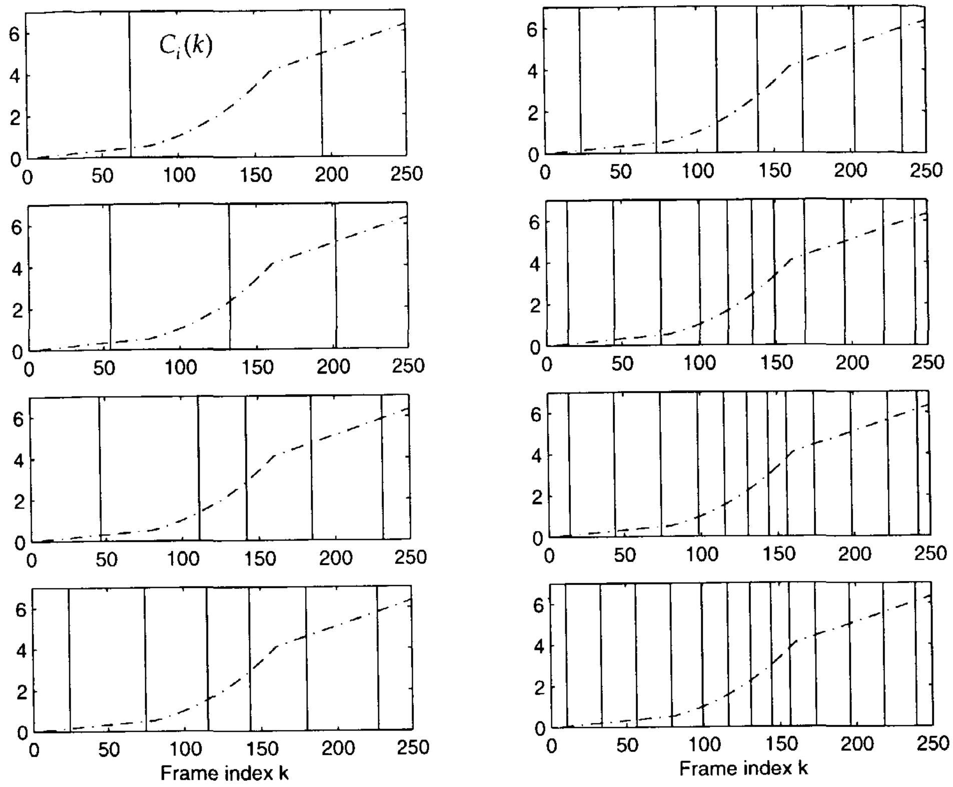


Figure 3.6: *Distribution of different number of key frames along a fictive video shot with a variable rate of visual-content variations.*

3.3.4 Experimental validation

The major issue in the key-frame extraction approach presented in this section is related to distributing a given number of key frames along a shot, such that the best possible summarization of the visual content of a shot is obtained. We therefore concentrate here on testing the optimization process (3.3.6) in the controlled situation. Visual content variations along an arbitrary shot i are modeled by two artificially produced functions $C_i(k)$. The form of the first function is given in the diagrams of Figure 3.6 and indicates that there is a constant low rate of visual content variation in the beginning of the shot, followed by an exponentially increasing variation rate, while the shot ends with a segment having the constant variation rate, but one that is higher than in the first shot segment. The exponential form of the second function, shown in diagrams of Figure 3.7, indicates a steadily increasing rate of visual-content change, for instance, in the case of an accelerated camera panning, tilting or tracking.

Independent of the number K_i of assigned key frames, the varying key-frame density along shot i should follow the visual-content variations modeled by the function $C_i(k)$. Furthermore, in shot segments with a constant variation rate, key frames are distributed homogeneously and all shot segments need to be represented. The last requirement should prevent that all or a large majority of key frames are concentrated on one small shot segment, while the rest of the shot's visual material is not captured by key frames.

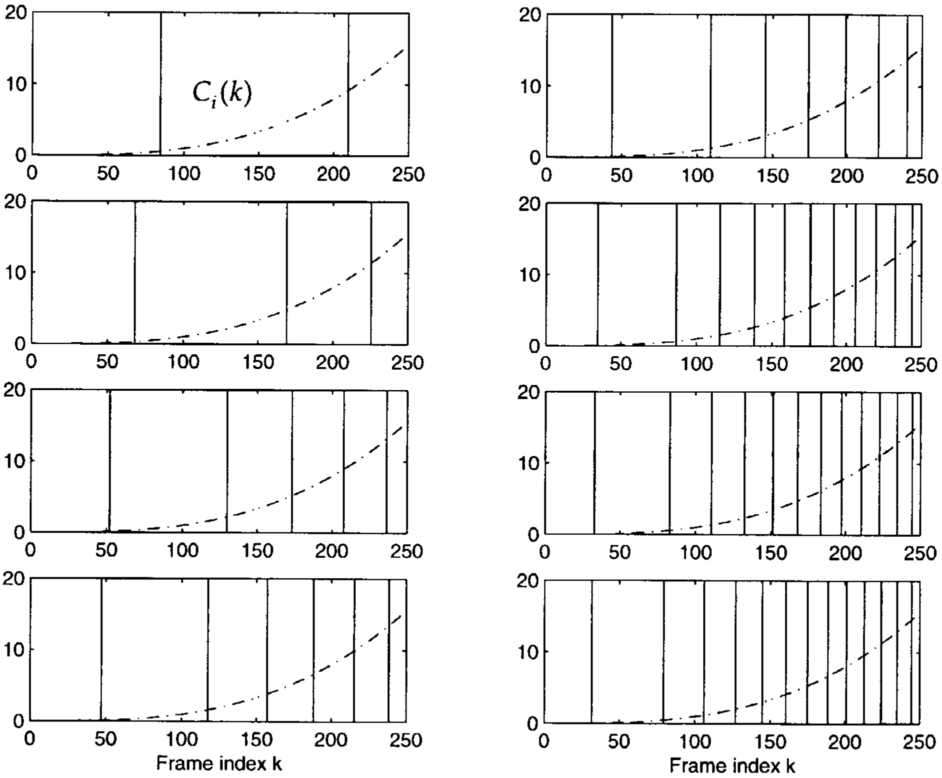


Figure 3.7: *Distribution of different number of key frames along a fictive video shot with exponential visual-content variations.*

We show in Figures 3.6 and 3.7 the results of distributing 2, 3, 5, 6, 7, 11, 12 and 13 key frames along a shot i for both modeling functions $C_i(k)$. In all cases, key frames – represented by vertical lines - were distributed as expected. On the one hand, it can be seen how in each case the concentration of key frames follows the dynamic of the function $C_i(k)$. On the other hand, from the fact that key frames are always distributed along the entire shot, it can be concluded that the visual content of all shot segments is well captured in each of the key-frame sets.

3.4 Key-frame extraction based on cluster-validity analysis

The objective of the key-frame extraction method presented in this section is to minimize the redundancy among video frames and provide a set of key frames for a given video sequence, which is similar to the one based on human cognition. While in the method from Section 3.3 key frames are extracted separately for each shot, the extraction procedure described here can be applied to a sequence containing an arbitrary number of shots. Furthermore, the method presented in this section does not require any human supervision or parameter (threshold) specification. This makes the extraction procedure very user friendly and it supplies the user with a stable quality of obtained key frames for any arbitrary sequence.

The visual-content redundancy is reduced here by applying a partitional clustering [Jai88] to all video frames. The underlying idea is that all frames with the same or similar visual content will be clustered together. Each cluster can be represented by one characteristic frame, which then becomes a key frame of a sequence, capturing all the visual material of that cluster. Since frames in different clusters contain different visual material, the redundancy among obtained key frames is low. At the same time, all variations of the visual material along a sequence is captured in its key-frame set.

Consequently, the problem of finding the optimal number of key frames for a given sequence is reduced to finding the optimal number of clusters in which the frames of a video can be classified based on their visual content. The main difficulty here is that the optimal number of clusters needs to be determined automatically. To solve this, we apply known tools and methods of cluster validity analysis and tailor them to our specific needs.

As illustrated in Figure 3.8, the extraction approach in this section consists of three major phases. First, we apply *N* times a partitional clustering to all frames of a video sequence. The prespecified number of clusters starts at 1 and is increased by 1 each time the clustering is applied. In this way N different clustering possibilities for a video sequence are obtained. In the second step, the system automatically finds the optimal combination(s) of clusters by applying the cluster-validity analysis. Here, we also take into account the number of shots in a sequence. In the final step, after the optimal number of clusters is found, each of the clusters is represented by one characteristic frame, which then becomes a new key frame of a video sequence.

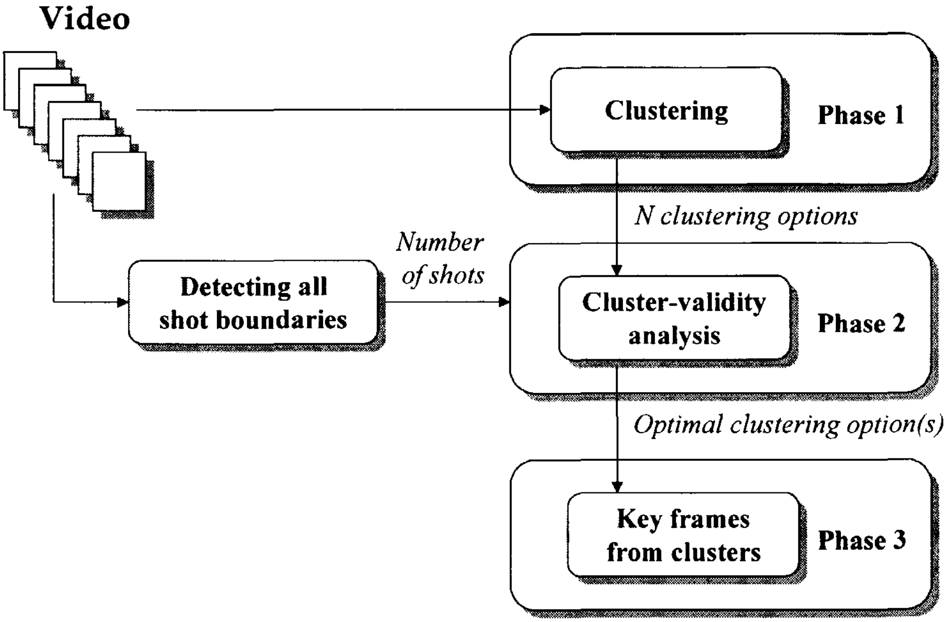


Figure 3.8: *Key-frame extraction scheme based on cluster validity analysis*

3.4.1 Clustering

The clustering process is performed on all video frames. For this purpose, each frame k of a video sequence is represented by a D-dimensional feature vector $\vec{\phi}(k)$, consisting of features $\varphi_v(k)$. The feature vector can be composed using texture, color, shape information, or any combination of those. Similarly as in the previous section, we wish to efficiently capture with key frames the changes introduced in the visual material, by e.g. camera panning, while the key frames must remain relatively insensitive to object motion. Therefore, we have chosen a D-dimensional feature vector, consisting of the concatenated $D/3$-bin color histograms for each of the component of the YUV color space. Furthermore, since $\vec{\phi}(k)$ is easily computable, we also compensate in this way for an increased computational complexity of the overall extraction approach due to the extensive cluster validity analysis, but still achieve an acceptable frame content representation. The feature vector used in this chapter is given as

$$\begin{aligned}\vec{\phi}(k) &= \langle \varphi_v(k) \mid v = 1,..,D \rangle = \\ &= \left\langle H_k^Y(1),..H_k^Y\left(\frac{D}{3}\right), H_k^U(1),..H_k^U\left(\frac{D}{3}\right), H_k^V(1),..H_k^V\left(\frac{D}{3}\right) \right\rangle\end{aligned} \tag{3.4.1}$$

By taking into account the *curse of dimensionality* [Jai82], we made the parameter D dependent on the sequence length. Now we compute it as $S/5$ [Jai82], whereby S is the number of frames to be clustered, and in this case also the number of frames in the sequence.

Since the actual cluster structure of the sequence is not known *a priori*, we first classify all frames of a sequence into 1 to N clusters. Thereby, the number N is chosen as the maximum allowed number of clusters within a sequence by taking into account the sequence length. Since each cluster corresponds to one key frame, the number N is equivalent to the maximum allowed number of key frames used in the previous section; here we use the same notation. Although N can be understood as a threshold parameter, its influence on the key-frame extraction result is minimal. This is because here we choose N much higher than the largest number of clusters to be expected for a given sequence. The longer the sequence, the higher is the potential number of clusters for classifying its video material. We found the variation of N with the number of sequence frames S defined by the function (3.4.2) suitable for the wide range of sequences tested:

$$N = N(S) = 10 + \operatorname{int}\left(\frac{S}{25}\right) \tag{3.4.2}$$

When we defined (3.4.2), we took into account that enough alternative options should be offered to the cluster validity analysis to obtain reliable results and that the number of options should increase with sequence length. On the other hand, the value N needs to be kept in limits, since the "noisy" clustering options become more probable with an increasing number of clusters and can negatively influence the cluster validity analysis.

After the clustering phase we perform a cluster-validity analysis to determine which of the obtained N different clustering options, i.e. which number of clusters, is the optimal one for the given sequence. In the following we will explain this procedure in full detail.

3.4.2 Cluster-validity analysis

For each clustering option characterized by n clusters ($1 \le n \le N$), we find the centroids c_i ($1 \le i \le n$) of the clusters by applying the standard *k-means* clustering algorithm to feature vectors (3.4.1) for all frames in the sequence. In order to find the optimal number of clusters for the given data set, we compute the *cluster separation measure* $\rho(n)$ for each clustering option according to [Dav79] as follows:

$$\rho(n) = \frac{1}{n} \sum_{i=1}^{n} \max_{1 \le j \le n \,\wedge\, i \ne j} \left(\frac{\xi_i + \xi_j}{\mu_{ij}} \right) \quad , \quad n \ge 2 \tag{3.4.3}$$

with the following parameters:

$$\xi_i = \left(\frac{1}{E_i} \sum_{v=1}^{E_i} \left| \vec{\phi}(v \mid v \in i) - \vec{\phi}(c_i) \right|^{\eta_1} \right)^{\frac{1}{\eta_1}} , \; \mu_{ij} = \left(\sum_{v=1}^{D} \left| \varphi_v(c_i) - \varphi_v(c_j) \right|^{\eta_2} \right)^{\frac{1}{\eta_2}} \tag{3.4.4}$$

The better all of the n clusters are separated from each other, the lower is $\rho(n)$ and the more likely is that the clustering option with n clusters is the optimal one for the given video material. The value ξ_i is called *dispersion* of the cluster i, while μ_{ij} is the *Minkowski metric* [Fri67] of the centroids characterizing the clusters i and j. For different parameters η_1 and η_2, different metrics are obtained [Dav79]. Consequently, the choice of these parameters has also a certain influence on the cluster-validity investigation. We found that the parameter setting η_1=1 and η_2=2 gave the best performance for our purpose. E_i is the number of elements in the cluster i. Note that the $\rho(n)$ values can only be computed for $2 \le n \le N$ due to the fact that the denominator in (3.4.3) must be nonzero. We now take all $\rho(n)$ values measured for one and the same sequence and for $2 \le n \le N$, and normalize them by their global maximum. Three different cases are possible for the normalized $\rho(n)$ curve, as illustrated in Figure 3.9a-c.

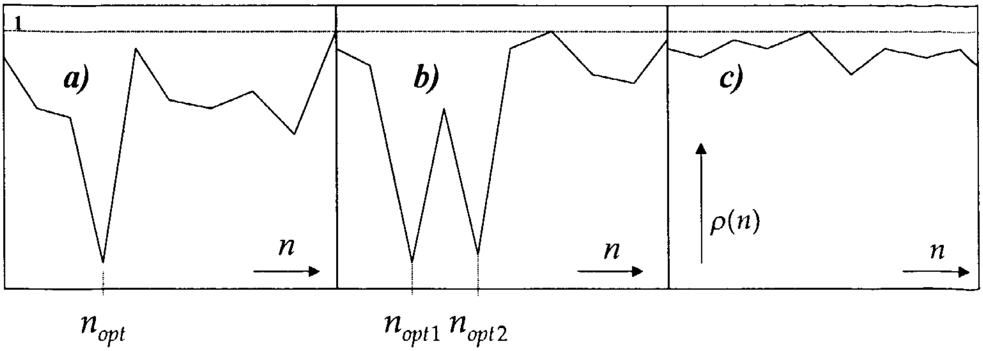


Figure 3.9: *Illustration of three possible cases for the normalized* $\rho(n)$ *curve.*

Case 1: The normalized $\rho(n)$ curve is characterized by a pronounced global minimum at $n = n_{opt}$, as shown in Figure 3.9a. This can be interpreted as the existence of n_{opt} clear natural clusters in the video material with $n_{opt} > 1$. In this case, we assume a set of n_{opt} clusters to be the optimal cluster structure for the given video sequence.

Case 2: The normalized $\rho(n)$ curve has s distinct low values. This means that it is possible to classify the given video material into s different numbers of clusters with a similar quality of content representation. An example of this is illustrated in Figure 3.9b for $s=2$ with options containing n_{opt1} or n_{opt2} clusters.

Case 3: All values of the normalized $\rho(n)$ curve are high and remain in the same range (around 1), as illustrated in Figure 3.9c. This case can be interpreted twofold: either there is no clear cluster structure within the given video material (e.g. an action clip with high motion) or the video sequence is stationary and it can be treated as one single cluster. In the remainder of this chapter we will consider a sequence as *stationary* if there is no or only non-significant camera or object motion (e.g. a zoom of a person talking, characterized by head and face motion). In general, if $\rho(n)$ curve is obtained as shown in Figure 3.9c, the decision about the optimal cluster structure is made depending on the detected number of shots in that sequence.

As a result of the above, the problem of finding the optimal cluster structure for any video sequence given by the normalized $\rho(n)$ values for $2 \le n \le N$ is reduced to recognizing the most suitable of the three above cases. To be able recognize this, we first must sort all the normalized values $\rho(n)$, $2 \le n \le N$, in the ascending order, resulting in a sorted set $\rho_{sorted}(m), 1 \le m \le N-1$. Then, we introduce the reliability measure $r(m)$, $1 \le m \le N-2$, defined as:

$$r(m) = \frac{\rho_{sorted}(m)}{\rho_{sorted}(m+1)} \tag{3.4.5}$$

Finally, we search for the value of the index m for which the function $r(m)$ has its minimum. Two possible results of the minimization procedure are given by the expressions

$$\min_{1 \le m \le N-2}(r(m)) = r(1) \tag{3.4.6a}$$

$$\min_{1 \le m \le N-2}(r(m)) = r(s), \quad s \ne 1 \tag{3.4.6b}$$

We will interpret these results for two different types of sequences, namely sequences containing several video shots and sequences corresponding to single video shots.

Sequences containing several video shots

We first analyze the situation involving sequences which contain more than one video shot. If there is a pronounced global minimum of the $\rho(n)$ curve at $n = n_{opt}$, as shown in Figure 3.9a, the reliability vector $r(m)$ has its global minimum at m=1. Therefore, the validity of (3.4.6a) is equivalent to the defined *Case 1*. Then, the optimal number of clusters is chosen as

$$n_{opt} = \min_{2 \le n \le N}(\rho(n)) \tag{3.4.7}$$

If the equation (3.4.6b) is valid, the scope of possible options is constrained either to *Case* 2 or to *Case* 3, where *Case* 3 can be considered less probable for the following two reasons: On one hand, the probability that there is a highly stationary content across several consecutive shots is low. On the other hand, enough distinction of the visual material belonging to different shots of the sequence can be expected, so that – if not only one –also several equally acceptable clustering options can be allowed. Therefore, we relate the validity of (3.4.6b) in case of complex sequences to the defined *Case* 2. That is, all cluster sets belonging to $\rho_{sorted}(i)$, $1 \leq i \leq s$, are taken as possible solutions for grouping the frames of a sequence.

Single video shots

The probability that one finds a natural cluster structure containing more than one cluster in sequences consisting of only one video shot is generally much smaller than finding one in sequences containing several shots. This is because changes of the visual content within a shot are continuous, mostly characterized by a camera/object motion without dominant stationary segments. For this reason, a large majority of $\rho(n)$ curves obtained for single video shots can be expected to correspond to the model in Figure 3.9c. Hence, it is crucial that a reliable distinction can be made between stationary shots and non-stationary shots of which the natural cluster structure is unclear, as this is the basis of obtaining a suitable abstract structure for single video shots.

If n_{opt} clusters are suggested by (3.4.7) for a given shot, and if that shot is stationary, the average intra-cluster dispersion $\bar{\xi}_{n_{opt}}$ computed over all n_{opt} clusters should be similar to the dispersion ξ_{one} computed for one cluster containing all frames of that shot. Otherwise, the dispersion ξ_{one} can be assumed to be considerably larger than $\bar{\xi}_{n_{opt}}$. In view of this analysis, we define the decision rule (3.4.8) to distinguish stationary shots from the non-stationary ones. For this purpose we first use (3.4.7) to find n_{opt} clusters for a given shot and compute the dispersion $\bar{\xi}_{n_{opt}}$. Then we also compute the dispersion ξ_{one} and compare both with $\bar{\xi}_{ref}$, which can be understood as the reference for the stationarity and is obtained by averaging dispersions measured for a large number of different stationary shots.

$$|\xi_{one} - \bar{\xi}_{ref}| \underset{stationary}{\overset{not\ stationary}{\gtrless}} |\bar{\xi}_{n_{opt}} - \bar{\xi}_{ref}| \tag{3.4.8}$$

If the shot is stationary, it is represented by only one cluster, including all frames of a shot. With non-stationary shots we proceed by checking the evaluations (3.4.6a-b). If the equation (3.4.6a) is valid, n_{opt} is chosen as the optimal number of clusters, indicating that clearly distinguishable natural clusters exist within the shot. If (3.4.6b) is valid, we can either assume that there are several clustering options for the given shot, or that no natural cluster structure can be recognized by the algorithm. The first possibility is relatively improbable because the range of content variations within a shot of an average length is limited. Therefore, the validity of (3.4.6b) for a single shot is related to an unclear cluster structure, which is difficult to represent. On the one hand, one single cluster is too coarse, since variations of the visual content are present. On the other hand, choosing too many clusters would lead to an over-representation of the shot. For these cases we found the smallest number of clusters proposed by (3.4.6b) as a good solution for this problem. Thus, from s clustering options suggested by (3.4.6b), we choose $n_{\min}$ clusters, defined by (3.4.9), to represent a single video shot with an unclear cluster structure:

$$n_{\min} = \min_{1<i<s}(n_i) \tag{3.4.9}$$

3.4.3 Key frames from clusters

Once a suitable cluster structure is found for the given video sequence, one representative frame is chosen from each of the clusters and taken as a key frame of the sequence. As being usual in the clustering theory, we choose for this purpose the cluster elements being closest to cluster centroids. We find the key frame F_i of the cluster i by minimizing the Euclidean distance between feature vectors (3.4.1) of all cluster elements k and the cluster centroid c_i, that is

$$F_i \Leftarrow \min_{1<k<E_i} \sqrt{\sum_{v=1}^{D} \left|\varphi_v(k) - \varphi_v(c_i)\right|^2} \tag{3.4.10}$$

3.4.4 Experimental validation

In order to test the video-abstraction method presented in this section, we concentrate here first on the evaluation of the proposed procedure for cluster-validity analysis, since both the key-frame sets and the preview sequences of a video abstract are directly dependent on the number and quality of obtained clusters.

We first tested the algorithm performance on sequences consisting of single video shots. For this purpose, we used 76 shots of a typical Hollywood-made movie and characterized them manually regarding the variations in their visual contents. The value of the parameter $\bar{\xi}_{ref}$ from (3.4.8) was obtained experimentally as 0.0228, for which we used a number of stationary shots of different lengths and containing different visual material, and can therefore be assumed generally valid. As illustrated in Table 3.1, each of the shots belonging to the test set is assigned a description of how its content varies in time. From this description, the most suitable number of clusters for grouping all the frames of a shot is derived and used as a ground truth. For instance, a stationary shot should get assigned 1 cluster, and a shot with Q distinct stationary segments should get assigned Q clusters. For 66 shots (87%) of the test set, their frames were clustered in the same way as given by the ground truth.

Shot 2:	Frames 42-286	stationary with minor object motion (1 cluster)
. . .		
Shot 10:	Frames 1582-1751	slight zoom (1 or 2 clusters)
. . .		
Shot 24:	Frames 4197-4358	two stationary camera positions (2 clusters)
. . .		
Shot 29:	Frames 5439-5776	three stationary camera positions (3 clusters)
. . .		
Shot 45:	Frames 7218-7330	slow camera panning (1 or 2 clusters)
. . .		
Shot 51:	Frames 8614-8784	stationary camera, followed by a strong zoom (2 clusters)
. . .		

Table 3.1: *A fragment of the test set for evaluating the performance of the cluster-validity analysis algorithm for single shots*

In order to test the performance of the cluster-validity analysis algorithm for sequences containing several shots, we established a controlled test environment involving a set of sequences with a clearly defined structure in terms of the possibilities for clustering their frames. For each of these sequences we estimated the suitable number of clusters for organizing their visual content and used this estimation as the ground truth. An indication of the algorithm performance can be found in Table 3.2 for the following test sequences used:

- **Sequence 1:** A dialog between two movie characters. Due to two fixed camera positions, two clearly defined clusters are expected, one for each of the characters.
- **Sequence 2:** Three movie characters in discussion, with the camera showing each of them separately and all together. Four clear clusters are expected.
- **Sequence 3:** Two major camera positions to be captured by two clear clusters.
- **Sequence 4:** A long sequence covering different visual material in a series. Five clear clusters are expected for sequence representation

Although for the fourth sequence a clear cluster structure containing 5 clusters was expected, the algorithm suggested two possible clustering options. However, this was still acceptable, since the 5 clusters found corresponded to the expected ones and the option with 6 clusters contained the same clusters and an additional one, capturing a segment with object motion.

Test sequences	Expected number of clusters	Expected cluster structure	Obtained number of clusters	Obtained cluster structure
Sequence 1	2	Clear	2	Clear
Sequence 2	4	Clear	4	Clear
Sequence 3	2	Clear	2	Clear
Sequence 4	5	Clear	5,6	Unclear

Table 3.2: *Algorithm performance for some video sequences containing more than one video shot*

Based on the results of cluster-validity analysis, key-frame sets and preview sequences were formed. For each of the obtained clusters, a key frame was extracted using (3.4.10). Besides of the fact that in each case the obtained cluster combination corresponded to the one given by the ground truth, we also found the resulting key-frame set providing a good representation of the video content. This implies that frames nearest to cluster centroids are suitable to be used as key frames, and that the cluster-validity analysis is here the crucial step in making the video abstract.

3.5 Discussion

After discussing the possibilities for automation of the key-frame extraction in the first section of this chapter, we presented in Sections 3.3 and 3.4 two methods by which key frames are automatically extracted for making a summary of a video's visual content. Both methods were developed such that the human intervention in dimensioning the extraction process is either limited to easily specified parameters or not necessary at all. In the method from Section 3.3, the maximal number N of key frames for the entire sequence is prespecified, while the approach from Section 3.4 is capable of functioning without human supervision. There, the value of the reference dispersion for stationary shots $\bar{\xi}_{ref}$ given in subsection 3.4.6 can widely be used for measurements on different sequences. Compared to the majority of key-frame extraction methods from recent literature, such a transparent parameter dependence makes the two approaches described in this chapter highly user-friendly.

Regarding the achieved visual-content representation, we first discuss in more detail the approach from Section 3.3. Two conclusions related to the ability of the method to summarize the visual content of a video can be drawn from the experimentally obtained key-frame distribution in Figures 3.6 and 3.7. First, the "sampling interval" between consecutive key frames is clearly dependent on the rate of visual content variation, i.e. on the steepness of the function $C_i(k)$. The higher the variation rate, the more key frames are used to capture the appearing new visual material. This indicates that all relevant elements of the visual content appearing in a shot will be represented in the resulting key-frame set. Second, although the sampling of the function $C_i(k)$ is generally not equidistant, key frames are always distributed such that the entire visual material of a shot is captured. This is opposed to an alternative where e.g. all K_i frames concentrate only on one shot segment. However, if the total number of key frames or any other threshold parameter is a constraint, it is difficult to prevent the cases of redundant key frames or to prevent ending up with too few key frames for a good sequence representation. Clear practical advantages of this method are the possibility of extracting key frames on-the-fly and of obtaining a good video summary and storyboard of a video, while keeping the amount of extracted information limited and closed to the prespecified one.

By using the extraction method presented in Section 3.4, one can obtain a very compact set of key frames for an arbitrary sequence, the quality of which is similar to a key frame set based on human cognition. Each frame selected using (3.4.10) can be assumed to have a high technical quality, since it corresponds to a cluster centroid, which is by definition the cluster element most similar to all other elements of that cluster. For that reason, having an "outlier" as a key frame, lying e.g. in a high-motion, in a blurred, dissolve or fade segment, is not as probable as having as a key frame a frame lying in a stationary, minimum-motion and maximum-clarity sequence segment. Although this method can be applied to a video segment of an arbitrary length, the segments of interest in this chapter are rather constrained to specific events, like for instance a dialog discussed before. The reason for this constraint is that long video segments are mostly characterized by an enormous variety in their visual contents, which is difficult to classify in a number of distinct clusters and, consequently, to represent by a limited number of key frames.

Chapter 4

High-Level Video Content Analysis

4.1 Introduction

Segmenting a video into shots, as discussed in Chapter 2, can be considered an elementary or a *low-level* video-analysis step. The reason for such a characterization is that this process, as well as the obtained results, do not depend on the actual content of the segmented video. In this chapter we concentrate on automatically analyzing a video at a higher level, at which *semantic video segments* can be distinguished.

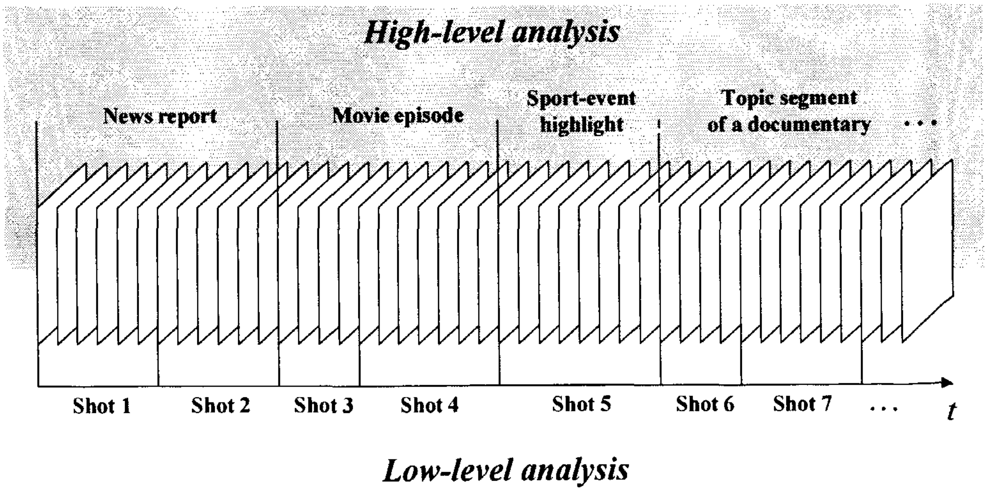


Figure 4.1: *Illustration of two different video-analysis levels*

As illustrated in Figure 4.1, the semantic video segments can be the reports in news programs, episodes in movies, highlights of sport events, topic segments of documentary programs, etc., and are concatenations of interrelated consecutive video shots. This indicates that the objective of high-level video content analysis can be formulated as finding *subsets* of all shot boundaries detected along a video, such that the series of consecutive shots, captured by shot boundaries belonging to these subsets, correspond to the semantic video segments of interest.

Autonomous systems able to analyze a video at a high (semantic) level can effectively be used to facilitate the user interaction with large volumes of video material stored in emerging digital video archives (libraries). Figure 4.2 illustrates how the results of high-level video analysis are used to organize the incoming or already stored video material, in order to provide access to semantic video segments of interest. The target applications of interest can be formulated as, e.g., search requests for all news reports on Bosnia, a movie episode containing the Alpine landscape or a favorite action scene, the "match point" of a tennis game, etc.

Similarly as in the case of key-frame extraction, the possibilities for automation of high-level video analysis are not unlimited. The first problem, as already discussed in Chapter 1, is that embedding the human ability of understanding the content of a video into an autonomous system is technically not feasible. A technically feasible solution to this problem is to find ways of relating the video semantics to some specific temporal behavior of suitable low-level features. There are numerous examples, which can indicate the possibilities for developing such methods. Some of them are described in sections 4.2, 4.3 and 4.4 of this chapter, such as detecting TV commercials in various programs, recovering the semantic structure of a news program or detecting the episodes in movies. However, since low-level features are powerless in some cases, for instance, when extracting video segments where a specific actor or the "Alpine landscape" appears, realistic objectives need to be set when choosing the target applications. Thus, instead of attempting to develop algorithms capable of finding the movie episode where "Alpine landscape" appears, alternative algorithms are aimed at, which first find all episode boundaries of a movie, represent them by a number of key frames and then submit the entire episode structure together with the key frames to a browsing tool. There, a user can easily get an overview of the movie content by looking at episode representation, recognize the "Alpine landscape" in one of the key frames and quickly retrieve the corresponding episode. As it will be shown in sections 4.2 and 4.3, detecting episode boundaries in a movie is possible by analyzing only the temporal consistency of low-level visual features of a movie.

The second problem concerns the missing ground truth for the results of high-level video analysis. These are the cases of, for instance, extracting the highlighting or most memorable video segments: due to a highly subjective human perception of the video content in such cases, the dispersion among the results obtained by a subjective analysis of one and the same sequence by several users will be high [Paa97]. However, in many other analysis cases, the problem of missing ground truth is not present, for instance, when detecting TV commercials in an arbitrary video or segmenting news programs into reports. Therefore, only the latter cases are considered in this chapter, although some examples of extracting semantic segments with a "questionable" ground truth will be described in Section 4.2.

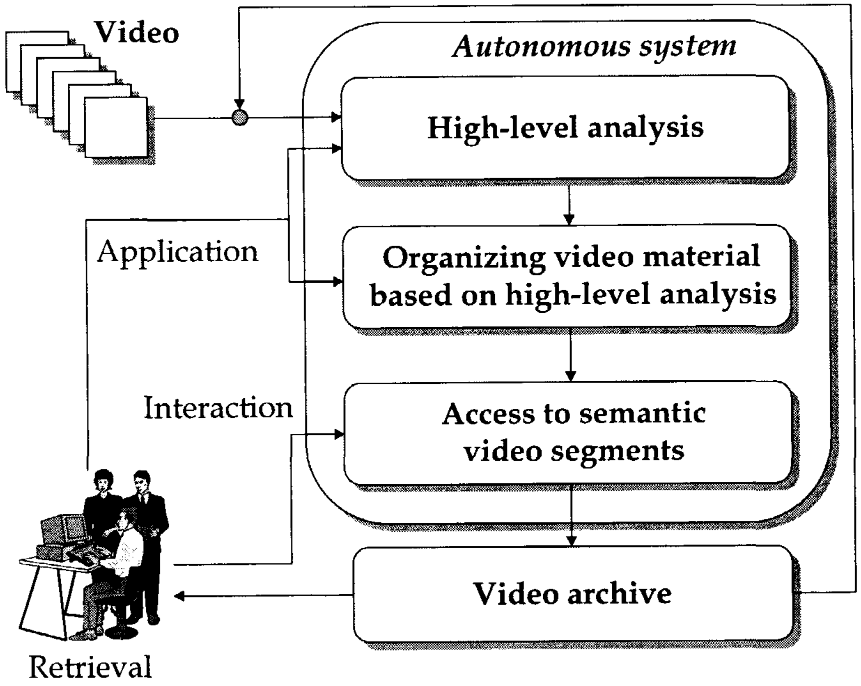

Video
Autonomous system
High-level analysis
Application
Organizing video material based on high-level analysis
Interaction
Access to semantic video segments
Video archive
Retrieval

Figure 4.2: *High-level video analysis and related operations embedded into an autonomous system*

An important issue which needs to be taken into account when automating the high-level video analysis is that no generally applicable analysis methods exist. One reason for this is that the semantic video segments of interest vary for different program types, user environments and applications. Furthermore, the characteristics of such segments, in terms of low-level features being most suitable for their detection, vary over different programs. Therefore, rather a specific analysis methodology can be developed for each particular program type. We set as the objective of this chapter to develop methods for automated high-level analysis of two specific video types: *movies* and *news* programs. In the remainder of this section, we will explain our motivation for choosing these two particular program types, and give an overview of methods for their analysis developed in Section 4.3 and Section 4.4 of this chapter.

With respect to the discussion in Chapter 1 of this thesis, we witness a strong development of home video libraries and expect that the digital storage of video material at home will soon overtake the current analog video cassette recording systems [Oka93], [dWi92], [dWi93], [Yan93], and that the volumes of stored data in home video archives will rapidly grow in time. Stored in these archives we find programs of various types, such as movies, news, documentaries, TV shows, sport broadcasts, etc. In view of their large popularity with private users, it can realistically be assumed that now and in the future, movies belong to the most frequently stored programs, covering the highest percentage of the stored data volumes in home video archives. Although the major user interest regarding a movie is simply to watch it, some other applications involving movie content may be desirable to users as well. Such applications include, for instance, retrieving and watching of selected movie scenes, searching for a shot where an actor appears in a funny pose and watching a short preview of a movie. Although they are still new for a common user, these applications can be expected to become more and more popular with the emerging and quickly developing technology [SMA]. The most important objective in this development is to provide methods and tools for automatically analyzing movie content and providing the user with semantic video segments of interest, with minimal user involvement in the analysis process. This is understandable, since users at home want to be entertained; they do not want to be burdened with programming or adjusting their video equipment, especially not if this burden exceeds the level reached by some current VCRs that can be programmed in various ways, but are already too complicated for an average user.

Regarding the movie analysis in this chapter, we follow the objective of automatically providing semantically meaningful entry points into a movie. These points are ideally the boundaries between consecutive movie *episodes*. We define an episode as a series of consecutive shots unified by the same *chronological time-frame* of the story. Since we can base our episode-boundary detection only on low-level features, it is unlikely that the detected boundaries always correspond to the actual episode boundaries. For this reason, the results of our approach generally do not reveal movie episodes but their approximates, which we define as *Logical Story Units* (*LSUs*). Compared to episodes, which are defined by their semantic contents, LSUs are defined in terms of specific spatio-temporal features which were found to be characteristic for an episode. As it will be explained later, we found the global temporal consistency of the visual content of an episode a powerful means for defining an LSU as an episode approximation.

A news broadcast, which is the other program type considered in this chapter, has been widely recognized as a highly interesting "storing object" in emerging large-scale digital video databases [Boy99], [Che97]. The main reason lies undoubtedly in the information content of news programs, which may be useful for applications in many professional areas (e.g. education, journalism, government) as well as for private needs. One could think of building up large information archives containing all available sorts of informative programs, e.g. news, documentaries, TV-debates, political or social discussions, reportages, etc. In such archives, news is at least as important as all other mentioned program types, since it concisely covers huge amounts of topics related to society, daily politics, sports, business, etc. The importance of news programs may even be larger, since not all daily events get a thorough coverage through e.g. a dedicated documentary. Collecting news over a longer time period from different broadcasters can therefore provide a solid top level for an information collection, whereby other informative programs on certain topics, if any, are linked to relevant news reports and serve as lower-level (more detailed) information sources. If large information archives are to be used efficiently, all the information segments need to be organized, either according to their topics or to any other specific criteria. Here, the issue of automating the news-program analysis and reducing human interaction is a great challenge, and becomes more and more important, if not crucial, as increasing information volumes are stored in video archives. Such tools should be capable of autonomously segmenting a news program into reports, recognizing the report topic or fulfilling of the specified application criteria, and should classify it with all other closely related reports, enabling, in this way, direct execution of search requests, such as "find me a business report in a CNN news program from 2.4.1997", or "give me everything what is available on car races".

In this chapter we concentrate on developing methods for automatically detecting anchorperson shots in an arbitrary news program. Since these shots are directly related to news reports and since, in most cases, they directly determine the report boundaries, their detection can be considered as an important step in automatically recovering the report structure of a news program, and also in reaching the overall topic-based organization structure of a news archive. When developing our method, we made use of a specific visual structure of an anchorperson shot, which can also be found repeatedly in different segments of a news broadcast.

Before we present a method for detecting LSU boundaries in movies in Section 4.3 and a method for detecting anchorperson shots in news programs in Section 4.4, in Section 4.2 we give a brief overview of some of the methods reported in recent literature, which indicate the current possibilities in using low-level features in extracting semantic aspects out of different types of video. The discussion belonging to this chapter can be found in Section 4.5.

4.2 Related work

4.2.1 Detecting different temporal events in a video

We start this section by discussing the method of capturing and characterizing a video by temporal events, such as dialogues, actions and story units [Yeu97]. The method consists of two major steps. In the first step, the *semantic labeling* of all video shots in a video is performed by applying *time-constrained clustering*. There, shots of a video are clustered based on their visual similarity and mutual temporal locality. In other words, two visually similar shots are not clustered together if they are too far from each other. G_i is the i-th cluster, the shots x, y and w are elements, d the distance between shots in terms of their visual similarity, T the maximum allowed temporal distance between two shots within the same cluster and δ the maximum allowed visual dissimilarity between two shots. The clustering procedure can be defined as follows:

- $\max\limits_{w \in C_i} d(x,w) \le \delta \quad , \forall x \in G_i$ (4.2.1a)
- $\max\limits_{y \in C_i} d_t(x,y) \le T \quad , \forall x \in G_i$ (4.2.1b)
- $d(x,w) > \delta$ or $d_t(x,w) > T$, $\forall x \in G_i$, $\forall w \in G_j$, $j \neq i$ (4.2.1c)

Assuming that all shots of a sequence are clustered in N clusters G_i, all shots within one and the same cluster get assigned a *label*. Then, by replacing each shot of a video by its corresponding label, we can represent the entire video as a series of labels, that is, as

$$A\,B\,C\,A\,D\,G\,H\,B\,A\,C\,D\,K\,H\,D\,B\,A\,C\ldots \tag{4.2.2}$$

In the second step, the label sequence (4.2.2) is investigated for different prespecified patterns appearing therein and corresponding to dialogs, actions, etc. For instance, dialog patterns are found as interchanging labels such as

$$\underbrace{A\,B\,A}_{\text{Dialog}}\,X\,Y\,Z\,\underbrace{A\,B\,A\,B\,A\,B}_{\text{Dialog}}\,C\,D\,\underbrace{E\,F\,E\,D\,E}_{\text{Dialog}}\,G\,H\,I\ldots \tag{4.2.3}$$

Or, as another example, action patterns are characterized by a series of shots with contrasting visual contents, expressed by no or only a minimal repetition of shot labels, that is

$$A\,B\,C\,D\,E\,F\,B\,G\,H\,I\ldots \tag{4.2.4}$$

4.2.2 Detecting scene boundaries in a movie

In [Ken98], the authors consider a movie as a series of consecutive *scenes* and propose an approach for finding probable boundaries of scenes. The approach is based on investigating the *coherence* measured along a series of consecutive shots and representing the consistence of the visual material contained therein. We first introduce the *recall* between shots s_m and s_n as $\mathrm{SRecall}(s_m, s_n)$ being proportional to the function $Sim(s_m, s_n)$ describing their visual similarity and the function $TR(s_m, s_n)$ taking into account their lengths and their relative temporal positions within a video, that is

$$\mathrm{SRecall}(s_m, s_n) = Sim(s_m, s_n)\,TR(s_m, s_n) \tag{4.2.5}$$

Then we define the total recall of all the shots older than the boundary by all the shots newer than the boundary as

$$\mathrm{Recall}(s_i, s_{i+1}) = \sum_{m<i}\sum_{n>i+1} \mathrm{SRecall}(s_m, s_n) \tag{4.2.6}$$

The coherence at the boundary between shots s_i and s_{i+1} is now computed as the total recall $\text{Recall}(s_i, s_{i+1})$ normalized by the maximum potential recall $\text{Ideal}(s_i, s_{i+1})$ possible at that boundary, that is

$$\text{Coh}(s_i, s_{i+1}) = \frac{\text{Recall}(s_i, s_{i+1})}{\text{Ideal}(s_i, s_{i+1})} \qquad (4.2.7)$$

The maximum potential recall $\text{Ideal}(s_i, s_{i+1})$ is computed similarly as $\text{Recall}(s_i, s_{i+1})$, except that $Sim(s_m, s_n)$ in (4.2.5) is fixed at its maximum value of 1.

The significant local minima of the coherence curve measured along a sequence indicate the potential scene boundaries. A methodology for high-level video segmentation based on similar principles as the one from [Ken98] was published in [Han99b] and [Han99c] and is explained in detail in Section 4.3.

4.2.3 Extracting the most characteristic movie segments

As discussed in the introduction to this chapter, it is very difficult to develop methods which automate the detection of semantic content elements for which no clear ground truth is defined. Therefore, in literature not many approaches can be found dealing with this problem. In [Pfe96] the *most characteristic* movie segments are extracted for the purpose of automatically producing a movie *trailer* (a short summary). Movie segments to be included in such a trailer are selected by investigating the specific visual and audio features and by taking those segments which are characterized by *high motion* (action), *basic color composition* similar to average color composition of a whole movie, *dialog-like audio track*, and *high contrast*. It is claimed that this method yields good quality movie abstracts, since "all important places of action are extracted" [Pfe96].

4.2.4 Automated recognition of video genres

The method for detecting video types (genres) presented in [Fis95] is a good example of an attempt to obtain some conclusions related to an extremely high abstraction level of a video by simply investigating its low-level features. The proposed approach consists of three steps. In the first step, the syntactic properties of a digital video, such as color statistics, shot-boundaries, motion vectors, simple object segmentation and audio-statistics, are analyzed. The results of the analysis are used in the second step to derive *video-style attributes*, such as shot lengths, camera panning and zooming, types of shot boundaries (abrupt ones vs. dissolves, fades, etc.), object motion and speech vs. music, which are considered to be the distinguishing properties for video genres. In the final step, an "educated guess" is made about the genre to which the video belongs, based on a mapping of the extracted style attributes with those corresponding to different prespecified genres. Experiments were reported using a number of sequences which were to be classified in one of the following genres: news, car races, tennis, commercials and animated cartoon. It is interesting to see in which way the style attributes were related to a particular genre. For instance, for a news program, the appearance of interchanging low- vs. high-motion video segments is investigated. There, low-motion segments correspond to anchorperson shots, which are separated by high-motion report segments. Also, a distinction is made between the anchorperson and some other "talking head" through the requirement that the periodically appearing low-motion segments need to be visually similar. This is done by computing and block-wise comparing the histograms of three subsequent low-motion segments. On the other hand, tennis is a good example of how audio can be used for detecting a video genre. As reported in [Fis95], a tennis game has a highly pronounced structure of the audio stream, characterized by interchanging "bouncing-ball" and speaker phases.

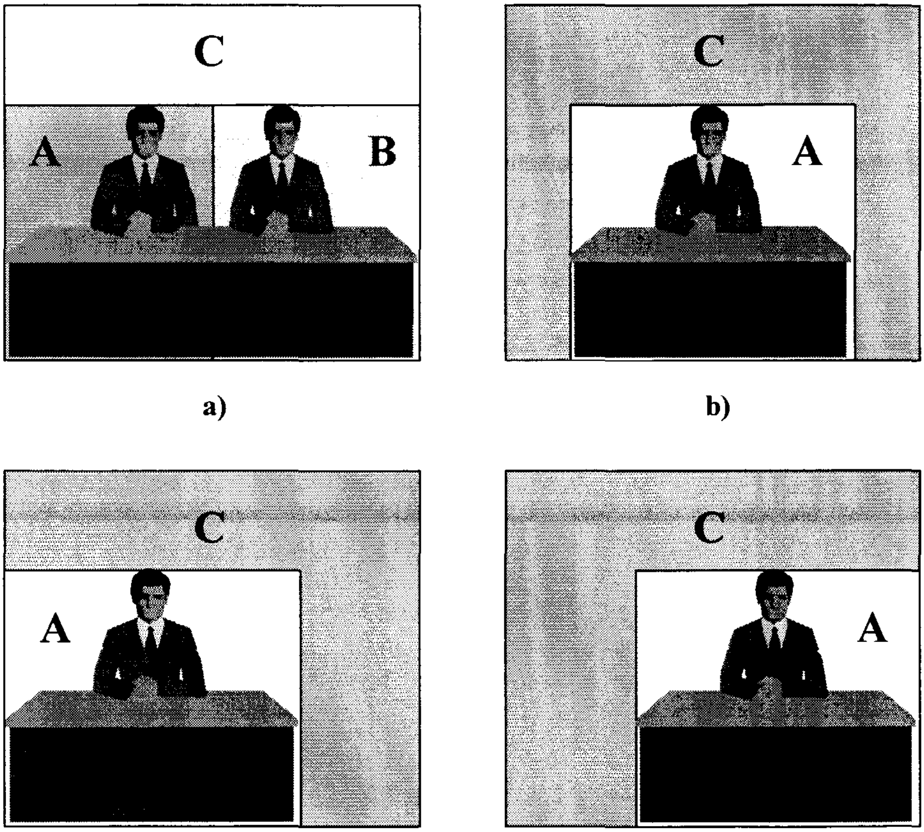
4.2.5 News-program analysis

We now move to high-level analysis of news programs. Due to their defined "container" structure, these programs are popular targets for developing content-analysis algorithms. The guiding objective when developing such algorithms is that these must automatically recognize the report structure of news programs and reach a topic-based organization of the news material on the system level with maximally reduced human interaction. While some of the proposed methods address this objective directly, many of them concentrate only on certain semantic aspects of a news program, which can be used at some later stages to reach the above objective. Examples are given in [Fur95], [Ari96], where the detection of anchorperson shots within a news program is performed.

Anchorperson shot detection using temporal shot characteristics

The approach to anchorperson-shot detection, presented in [Fur95], consists of three major steps. In the first step, potential anchorperson shots are found based on the fact that these shots are more or less stationary, compared to other shots within a news program. So, a shot is considered a candidate anchorperson shot if the following two expressions are valid, with μ and σ^2 being the mean and variance of discontinuity values $z(k,k+1)$, measured along a shot:

$$\begin{aligned} \mu &< T_1 \\ \sigma^2 &< T_2 \end{aligned} \tag{4.2.8}$$



a)

b)

c)

d)

Figure 4.3: *Spatial structure models of four different types of anchorperson shots*

The second step is performed by taking a candidate anchorperson shot and analyzing the temporal changes in regions *A*, *B* and *C*, indicated on four characteristic types of anchorperson shots in Figure 4.3. Changes between consecutive frames along these shots are expected in frame regions where the speakers are, that is, in regions *A* and *B*. Opposed to this, no motion should be registered in regions *C*. These conditions can mathematically be formulated as follows:

$$\begin{array}{lcl} \mu_A > T_3 > 0 & & \sigma^2{}_A > T_4 > 0 \\ \mu_B > T_3 > 0 & \text{and} & \sigma^2{}_B > T_4 > 0 \\ \mu_C \approx 0 & & \sigma^2{}_C \approx 0 \end{array} \tag{4.2.9}$$

If a candidate anchorperson shot fulfills the conditions (4.2.9) for any combinations of regions, as indicated in Figure 4.3, it can be considered as an anchorperson shot.

Obviously, this procedure can also provide the information on the anchorperson-shot type (e.g. one of the four types from Figure 4.3). In view of this, after the first anchorperson shot is found, it is used to find all anchorperson shots of the same type among the remaining candidates. This is done by computing the average frame of the detected anchorperson (*model image*) and by comparing it to average frames of candidate shots. The second step of the procedure is repeated until model images of all anchorperson-shot types appearing in a news broadcast are computed and all anchorperson shots have been detected.

Anchorperson-shot detection using planar graphs

The method for anchorperson-shot detection proposed in [Ari96] uses the results of shot boundary detection to detect the appearance of an anchorperson. The first frame of each detected shot is extracted and considered as a "cut point". Then, a planar graph is formed with the cut points as nodes and the shots as edges connecting each two nodes. Under the assumption that each news report starts and ends with the same anchorperson shot, a loop structure can be assumed within the obtained graph, where each loop corresponds to a report starting from one node (anchorperson) and ending in the same node. This is illustrated in Figure 4.4. Then, in order to detect all starting frames of anchorperson shots, nodes forming the loop points need to be detected. For this purpose, a threshold is defined and all nodes with distances smaller than the threshold are considered as starting frames of anchorperson shots.

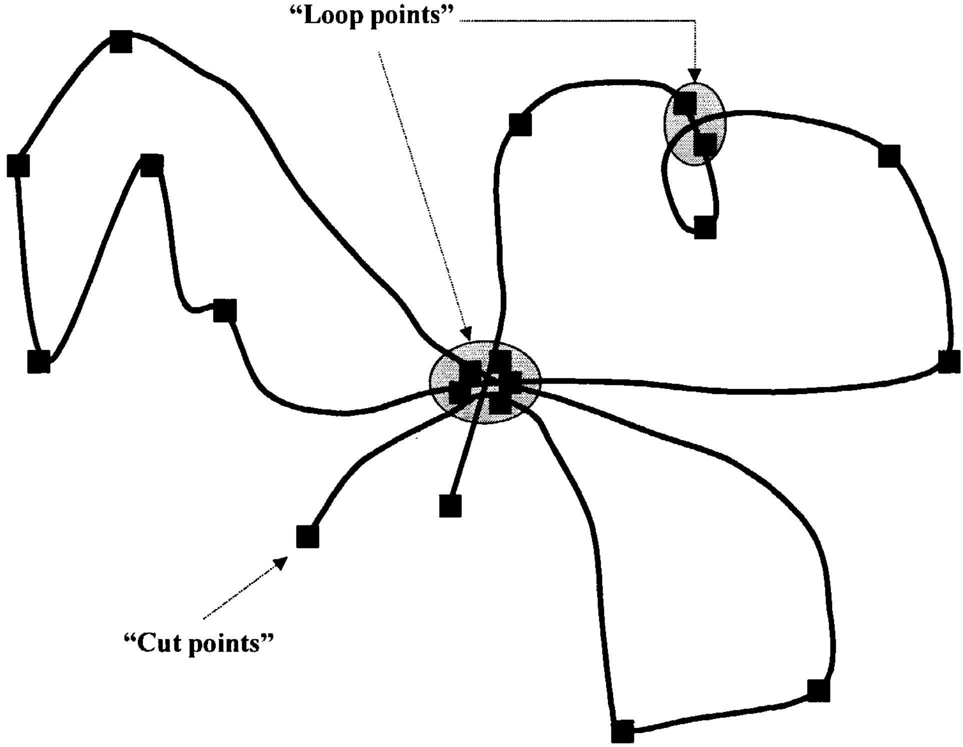


Figure 4.4: *"Cut points" and "loop points"*

Recovering news-program structure by combining different media

An attempt to automatically recover the entire semantic structure of a news broadcast can be found in [Hua99b]. The proposed approach for high-level news analysis is based on utilizing cues from different media and has the objective of recovering semantic segments from broadcast news at different levels of abstraction. The authors observe a hierarchy of a typical news program, which consists of four semantic levels. At the lowest level, a news program can be split into news material and commercials. Then, within the news material, anchorperson shots can be separated from shots taken outside the studio. Here, the anchorperson shot usually introduces and summarizes a report, which is followed by detailed reporting from a site. At the next level, anchorperson shots and related shots from different sites can be merged into reports.

In order to recover the first hierarchy level, news is separated from commercials by registering the changes in the audio-waveform, which are mainly caused by the background music in the commercials. In the second step, the news material is classified into segments corresponding to anchorperson shots and the rest using text-independent speaker recognition techniques. These techniques make it possible to distinguish an anchorperson segment from background speech coming from other sources (non-anchorperson shots) as well as from various audio segments (e.g. music in commercials). This step is meant to use the detected anchor's identity to hypothesize about a set of report boundaries that consequently partition the continuous text into adjacent blocks of text, each corresponding to a single report. In further steps this helps in obtaining higher levels of hierarchy by grouping the text blocks into reports.

4.2.6 Methods for analyzing sports programs

Instead of movies and news, some authors considered sports programs when they developed high-level video-analysis methods. The analysis approach presented in [Sau97] uses spatio-temporal features to classify the video material of a basketball sequence in segments such as wide-angle and close-up views, fast breaks, steals, potential scores, number of ball possessions and possession times. For instance, shots are classified as wide-angles and close-ups, by an investigation of their motion intensity. While wide-angle shots are taken from a distance and are relatively stationary, close-up shots are highly dynamic, since the camera only shows a small portion of a scene and usually follows an object. The term "fast break" is defined as a "fast" movement of the ball from one end of the court to the other. In order to detect fast breaks, one accumulates the magnitude of the motion vectors along a sequence in such a way that the accumulation is reset to zero each time the motion changes direction. If the camera follows the ball during a fast break, a long and persistent pan is registered in these segments. Therefore, the search for fast breaks is actually the search for extremely long segments in the accumulation curve between two reset points. By exploring specific camera motion and lengths of corresponding video segments, one can also characterize steals and ball-possessions.

Also, as referred to in [Sau97], a system is developed in [Gon95], that can automatically parse TV soccer broadcasts. There, the standard layout of a soccer field was used to classify the video material into nine different categories, such as "around the left penalty line" or "near the top right corner".

4.3 Automatically segmenting movies into Logical Story Units

As already discussed in Section 4.1, we here present an approach for high-level movie analysis which was developed with the objective to provide semantically meaningful entry points into a movie. Although we envision such entry points as boundaries between consecutive movie episodes, detecting episode boundaries with great precision is difficult if only spatio-temporal features are used. Approximates of movie episodes captured by the boundaries detected using our approach are defined here as Logical Story Units. We start this section by justifying the episode boundaries as the meaningful semantic entry points into a movie. Then, we choose appropriate low-level features and define the LSU-boundary procedure such that the detected boundaries are as close to the actual episode boundaries as possible.

4.3.1 Hierarchical model of a movie structure

We first define a hierarchical model of a movie structure, which consists of three hierarchy levels, namely

- Shots
- Events
- Episodes

While shots are elementary "technical" temporal units of a video in general, we define an *event* as the smallest semantic segment of a movie. Such an event can be a dialog, an action scene or, generally, any series of shots unified by *location* or *dramatic incident*. However, an event does not need to be an unbroken series of consecutive shots; it can also alternate with another event. This is often used in the process of movie generation to represent several events taking place in parallel. Several alternating events are, all together, a good example of the highest semantic segment, which we define in this chapter as an *episode*. There, all events are unified by the same *chronological time frame* of the story and form a rounded context, which is in a certain sense separated from the neighboring contexts.

An episode does not need to be related to several events; it can also concentrate on a single event. Since no shot within a movie is isolated but semantically it always belongs to a certain part of the story, each shot can be said to belong to one or to another episode. This implies that a movie can be understood as a *concatenation of episodes*. The hierarchical model of the movie structure, involving shots, events and episodes, is illustrated in Figure 4.5. There, we denote the fragment *i* of the event *j* by E_i^j. The model shows how an episode is built up around one movie event or around several of them taking place in parallel. Thereby a shot can either be a part of an event or it can serve for its "description" by, e.g., showing the scenery where the next or the current event takes place, showing a "story telling" narrator in typical retrospective movies, etc. In view of such a distinction, we further refer to shots of a movie as either *event shots* or *descriptive shots*.

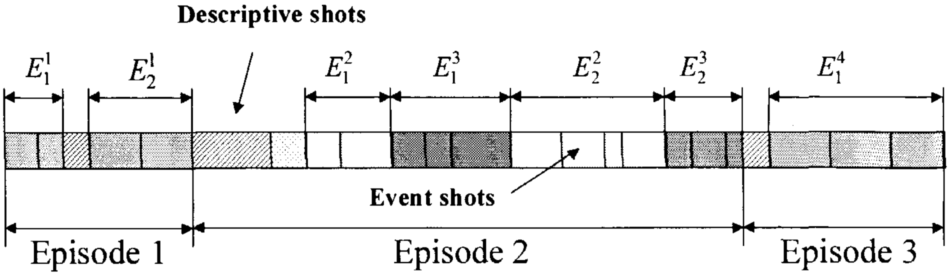


Figure 4.5: *Episodes 1 and 3 cover only one event and have a simple structure. Episode 2 covers two events, presented by their alternating fragments.*

Based on the above definitions, it can be said that if a movie is segmented into episodes, each boundary between two consecutive episodes provides an entry point into a new global segment of a story, having a rounded context and therefore being suitable for retrieval separately from the rest of the movie.

4.3.2 Definition of LSU

We now define the procedure of detecting the LSU boundaries such that they closely approximate the actual episode boundaries. In order to do this, we first analyze the characteristics of an episode and investigate the possibilities to efficiently capture them using suitable features.

It can realistically be assumed that an event is related to a specific location (scenery) and to certain movie characters. In other words, every now and then within an event similar *visual content elements* (scenery, background, people, faces, dresses, specific patterns, etc.) appear, and some of them even appear repeatedly. Since an episode is built around events, the same can be assumed for an episode as well; it is either related to only one event or to several of them alternating in time:

> **Assumption:** *An episode can generally be characterized by a global temporal consistency of its visual content, that is, by good matches of its visual-content elements found anywhere within a certain limited time interval.*

According to this assumption, approximate episode boundaries can be found by investigating the temporal behavior of visual low-level features. In this sense, we define the LSU as follows:

> *An LSU is a series of temporally contiguous shots which is characterized by overlapping links that connect shots with similar visual content elements.*

Since the definition of an LSU is based only on an assumption about the episode characteristics, which is not always fulfilled, the LSU boundaries do not exactly correspond to the episode boundaries in some cases. We will now explain some of the most characteristic problematic cases in view of the LSU definition and the movie-structure model in Figure 4.5.

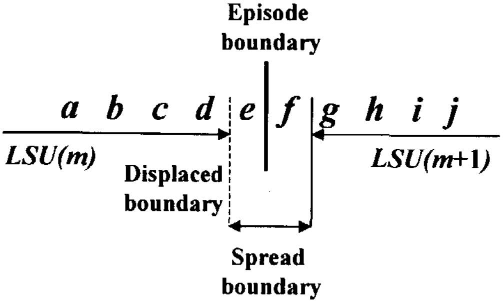


Figure 4.6: *Possible differences between an LSU and an episode boundary.*

For this purpose, we first investigate a series of shots *a* to *j*, as illustrated in Figure 4.6. Let the boundary between episodes *m* and *m*+1 lie between shots *e* and *f*. We now assume that the shot *e*, although belonging to the episode *m*, has a different visual content than the rest of the shots in that episode. This can be the case if, e.g., *e* is a descriptive shot, which generally differs from event shots. Consequently, the content consistency could be followed by overlapping links in the *LSU(m)* up to shot *d*, so that the LSU boundary is found between shots *d* and *e*. If the shot *e* contains enough visual elements also appearing in the episode *m*+1 so that a link can be established, *e* is assumed to be the first shot of the *LSU(m*+1) instead of shot *f*. This results in a *displaced* episode boundary, as shown in Figure 4.6. However, if no content-consistency link can be established between shot *e* and any of the shots from the episode *m*+1, another LSU boundary is found between shots *e* and *f*. Suppose that *f* is a descriptive shot of the episode *m*+1, containing a different visual content than the rest of the shots in that episode, so again no content-consistency link can be established. Another LSU boundary is found between shots *f* and *g*. If the linking procedure can now be started from shot *g*, it is considered to be the first shot of the new *LSU(m*+1). In this case, not a precise *LSU boundary* is found but one that is spread around the actual episode boundary, where all places where the actual episode boundary can be defined are taken into consideration. Consequently, the shots *e* and *f* are not included in the LSUs, as shown in Figure 4.6.

We now proceed to define the LSU analytically, using the illustration of the LSU definition in Figure 4.7. The basis of the definition of an LSU given above is that a visual dissimilarity between two video shots can be measured. For now we assume that the dissimilarity $D(k,k+l)$ between the shots k and $k+l$ is quantitatively available. Then, three different cases can be distinguished, depending on the relation of the current shot k and the m-th LSU.

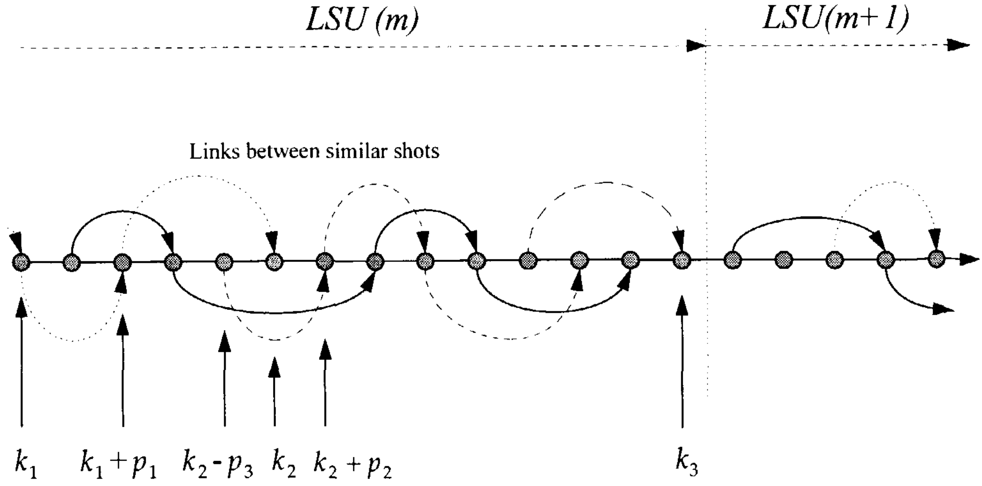


Figure 4.7: *Illustration of LSUs characterized by overlapping links connecting similar shots*

Case 1: Visual content elements from shot k_1 reappear (approximately) in shot k_1+p_1. Then, shots k_1 and k_1+p_1 form a linked pair, illustrated in Figure 4.7 by the arrow. Since shots k_1 and k_1+p_1 belong to the same *LSU(m)*, consequently all intermediate shots also belong to *LSU(m)*:

$$[k_1,k_1+p_1]\in LSU(m) \quad \text{if } p_1 \Leftarrow \min_{l=1,\ldots,c} D(k_1,k_1+l) < T(k_1). \tag{4.3.1}$$

Here c is the number of subsequent shots (look-ahead distance) with which the current shot is compared to check the visual dissimilarity. The threshold function $T(k)$ specifies the maximum dissimilarity allowed within a single LSU. Since the visual content is usually time-variant, the function $T(k)$ also varies with the shot under consideration.

Case 2: There are no subsequent shots with sufficient similarity to shot k_2, i.e. the inequality in (4.3.1) is not satisfied. However, one or more shots preceding shot k_2 link with shot(s) following shot k_2 (see Figure 4.7). Then, we enclose the current shot by a pair of shots that belongs to *LSU(m)*, i.e.

$$\begin{aligned}&[k_2-p_3,k_2+p_2]\in LSU(m)\\ &\text{if } (p_3,p_2>0) \Leftarrow \min_{i=1,\ldots,r}\ \min_{l=-i+1,\ldots,c} D(k_2-i,k_2+l) < T(k_2).\end{aligned} \tag{4.3.2}$$

Here r is the number of shots to be considered preceding the current shot k_2 (look-back distance).

Case 3: If for the current shot k_3 neither (4.3.1) nor (4.3.2) is fulfilled, and if shot k_3 links with one of the previous shots, then shot k_3 is the last shot of *LSU(m)*. This can also be seen in Figure 4.7.

4.3.3 Novel approach to LSU boundary detection

The objective is to detect the boundaries between LSU's, given the definition of an LSU and the concept of linking shots described by Cases 1-3 from the previous section. In principle one can check equations (4.3.1) and (4.3.2) for all shots in the video sequence. This, however, is computationally intensive and also unnecessary. According to (4.3.1), if the current shot *k* is linked to shot *k+p* (link between shots (a) and (b) in Figure 4.8), all intermediate shots automatically belong to the same LSU, so they need not to be checked. Only if no link can be found for shot *k* (shot (c) in

Figure 4.8), it is necessary to check whether at least one of *r* shots preceding the current shot *k* can be linked with a shot *k+p* (for *p>0,* as stated in (4.3.2)). If such a link is found (link between shots (d) and (e) in Figure 4.8), the procedure can continue at shot *k+p*; otherwise shot *k* is at the boundary of *LSU(m)* (shot (e) in Figure 4.8). The procedure then continues with shot *k*+1 for *LSU(m*+1).

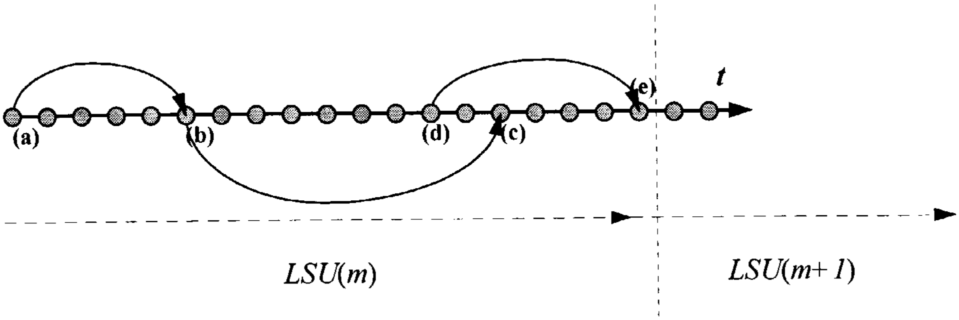


Figure 4.8: *Illustration of the LSU boundary-detection procedure. The shots indicated by (a) and (b) can be linked and are by definition part of LSU(m). Shot (c) is implicitly declared part of LSU(m) since the shot (d) preceding (c) is linked to a future shot (e). Shot (e) is at the boundary of LSU(m) since it cannot be linked to future shots, nor can any of its r predecessors.*

In order to determine whether a link can be established between two shots, we need the threshold function *T*(*k*). We compute this threshold recursively from already detected shots that belong to the current LSU. For this purpose we define the *content inconsistency value u*(*k*) of shot *k* as the minimum of *D*(*k*,*n*) found in (4.3.1) (or in (4.3.2) if (4.3.1) does not hold), that is

$$u(k) = \begin{cases} D(k_1, k_1 + p_1) & \text{if (4.3.1) holds} \\ D(k_2 - p_3, k_2 + p_2) & \text{if (4.3.2) holds} \end{cases} \tag{4.3.3}$$

Then the threshold function *T*(*k*) we propose is:

$$T(k) = \frac{\alpha}{N_k + 1}\left(\sum_{i=1}^{N_k} u(k-i) + u_0\right) \tag{4.3.4}$$

Here α is a fixed parameter whose value is not critical between 1.3 and 2.0. The parameter N_k denotes the number of links in the current LSU that have led to the current shot *k*, while the summation in (4.3.4) comprises the shots defining these links. Essentially the threshold *T*(*k*) adapts itself to the content inconsistencies found so far in the LSU. It also uses as a bias the last content inconsistency value u_0 of the previous LSU for which (4.3.1) or (4.3.2) is valid.

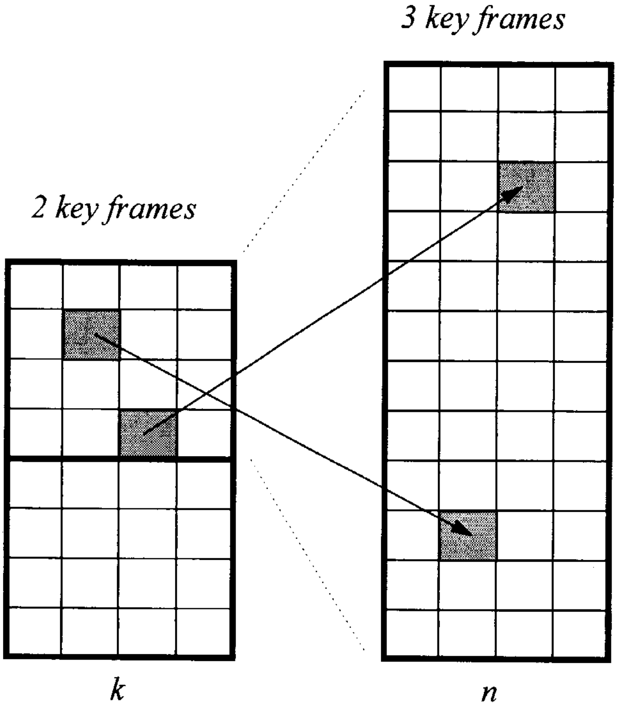


Figure 4.9: *Comparison of shot k with shot n by matching HxW blocks from each key frame of shot image k with shot image n. Shot k had 2 key frames and shot n had 3 key frames.*

4.3.4 Inter-shot dissimilarity measure

The LSU detection algorithm and the computation of the threshold function require the use of a suitable dissimilarity function *D*(*k*,*n*). We assume that the video sequence is segmented into shots, and that each detected shot is represented by one or multiple key frames so that its visual information is captured in the best possible way.

For each shot, all key frames are merged in one large variable-size image, called the *shot image*, which is then divided into blocks of *HxW* pixels. Each block is now a simple representation of one visual-content element of the shot. Since we cannot expect an exact shot-to-shot match in most cases, and because the influence of those details of a shot's visual content which are not interesting for an LSU as a whole should be as small as possible, we choose to use only those features that describe the blocks *globally*. In view of this we only use the average color in the $L^*u^*v^*$ uniform color space as a block feature.

For each pair of shots *(k,n)*, with $k<n$, we would now like to find the mapping between the blocks b_k and b_n, each being an *HxW* block from the shot image *k* and *n*, respectively, such that

- each block b_k in a key frame of shot image *k* has a unique correspondence to a block b_n in shot image *n*. If a block b_n has already been assigned to a block b_k of a key frame belonging to shot image *k*, no other block of that key frame may use it. All blocks b_n are only available when a new key frame of shot *k* is to be matched. Figure 4.9 illustrates this in more detail.
- the average distance in the $L^*u^*v^*$ color space between corresponding blocks of the two shot images is minimized:

$$\min_{\text{all possible block combinations}} \sum_{\text{all blocks } b} d(b_k, b_n) \tag{4.3.5}$$

where

$$d(b_k, b_n) = \sqrt{\left(L^*(b_k) - L^*(b_n)\right)^2 + \left(u^*(b_k) - u^*(b_n)\right)^2 + \left(v^*(b_k) - v^*(b_n)\right)^2} \tag{4.3.6}$$

and where all possible block combinations are given by the first item. Unfortunately this is a problem of high combinatorial complexity. We therefore use a suboptimal approach to optimize (4.3.5). The blocks b_k of a key frame of shot *k* are matched in the unconstrained way in shot image *n*, starting with the top-left block in that key frame, and subsequently scanning in the line-fashioned way to its bottom-right block. If a block b_n has been assigned to a block b_k, it is no longer available for assignment until the end of the scanning path. For each block b_k the obtained match yields a minimal distance value, $d_1(b_k)$. This procedure is repeated for the same key frame in the opposite scanning fashion, i.e. from bottom-right to top-left, yielding a difference mapping for the blocks b_k and a new minimal distance value for each block, denoted by $d_2(b_k)$. On the basis of these two different mappings for a key frame of shot k and corresponding minimal distance values $d_1(b_k)$ and $d_2(b_k)$ per block, the final correspondence and actual minimal distance $d_\mathrm{m}(b_k)$ per block is constructed as follows:

- $d_\mathrm{m}(b_k) = d_1(b_k)$, if $d_1(b_k) = d_2(b_k)$ (4.3.7a)
- $d_\mathrm{m}(b_k) = d_1(b_k)$, if $d_1(b_k) < d_2(b_k)$ and $d_1(b_k)$ is the lowest distance value measured for the assigned block in the shot image n (one block in shot image n can be assigned to two different blocks in a key frame of shot k: one time in each scanning direction) (4.3.7b)
  $d_\mathrm{m}(b_k) = \infty$, otherwise. (4.3.7c)
- $d_\mathrm{m}(b_k) = d_2(b_k)$, if $d_2(b_k) < d_1(b_k)$ and $d_2(b_k)$ is the lowest distance value measured for the assigned block in the shot image n (4.3.7d)
  $d_\mathrm{m}(b_k) = \infty$, otherwise. (4.3.7e)

where ∞ stands for a fairly large value, indicating that no objective best match for a block b_k could be found. The entire procedure is repeated for all key frames of a shot k, leading to one value $d_\mathrm{m}(b_k)$ for each block of a shot image k. Finally the average of the distances $d_\mathrm{m}(b_k)$ of the B best-matching blocks (those with lowest $d_\mathrm{m}(b_k)$ values) in the shot image k is computed as the final inter-shot dissimilarity value:

$$D(k,n) = \frac{1}{B} \sum_{\substack{B \text{ best matching} \\ \text{blocks}}} d_m(b_k) \tag{4.3.8}$$

The reason for taking only the B best-matching blocks is that two shots should be compared only on a global level. In this way, we allow for inevitable changes within the LSU, which, however, does not degrade the global continuity of its visual content.

4.3.5 Experimental validation

We illustrate the performance of the proposed LSU boundary-detection approach with the example of two full-length movies which belong to quite different categories in view of their dynamics and the variety of their contents. The objective of the evaluation is to compare the obtained LSU boundaries with the actual episode boundaries and to investigate the consistency of results for both different types of movies.

Establishing the ground truth

In order to evaluate the performance of our segmentation procedure, we need reference episode boundaries, serving as a ground truth. Generally, such reference boundaries can be obtained if the information about the movie generation process is available, i.e. the movie script. Since such information was not available for our tests, the first step in the evaluation procedure was to obtain a set of reference boundaries which (closely) correspond to the ground truth. This was done by a number of test subjects, who manually segmented both movies in units which they believed to be episodes. The obtained segmentation results differed mainly in the number of episode boundaries that were detected; this was especially noticeable in the complex movie segments and can be explained by the fact that each subject perceived that episode to be constructed differently. On the basis of manual segmentation results, we defined two different classes of episode boundaries

- *Probable boundaries* – registered by all test subjects
- *Potential boundaries* – registered by some of the test subjects

In total, 19 probable and 17 potential boundaries were detected for the first movie and 26 probable and 16 potential boundaries for the second one. Since the *probable* boundaries were those all test subjects had selected, we considered them to be fundamental, and relevant for evaluating our detection method. This is not the case with *potential* boundaries, and they are, therefore, not considered in the boundary set belonging to the ground truth.

Parametrizing the LSU-boundary detection procedure

After establishing the ground truth, we had our algorithm perform the automatic segmentation of the movies for different values of parameters B and α. Thereby, we limited the range of the parameter α only to [1.4-1.5], while B, here expressed as a percentage of the total number of blocks in a shot, varied in the range 40-70%. We learned that taking less than 30% of the blocks makes the inter-shot comparison too coarse. On the other hand, more than 70% makes the comparison too detailed. Although both parameters determine the sensitivity of the detection procedure and, consequently, also the number and positions of detected boundaries, parameter B is more interesting since it defines the limits of inter-shot comparison, concerning both the amount of detail taken into account and how "global" this comparison should be. On the other hand, we left the parameters c and r, defined in (4.3.1) and (4.3.2), constant at values c=8 and r=3, since the segmentation results were fairly insensitive to the setting of these parameters. We represented each shot by two subsampled key frames, taken from the first and last shot segment. Dimensions of key frames were 88x72 and 80x64, and the parameters H and W determining the size of the blocks to compute (4.3.8) were chosen correspondingly, as 8.

Evaluation

We now evaluate the performance of the detection algorithm for each parameter pair (B, α). In view of the possible tolerable displacements between an LSU and the corresponding episode boundary (Figure 4.6), we consider here an automatically obtained LSU boundary as properly detected if it was close enough to the one detected manually. For this purpose we set the maximum tolerable distance to 4 shots. Any other automatically detected boundary was considered to be *false*. Also, if no LSU boundary was detected within 4 shots of the actual episode boundary, it was considered *missing*.

In order to quantitatively estimate the quality of the automated boundary detection for a certain parameter combination (B, α), we used the following expression:

$$Q = \frac{\text{Properly detected probable boundaries}}{1 + \text{Falsely detected boundaries}} \tag{4.3.9}$$

The parameter Q denotes the quality of the boundary detection, depending on the number of properly detected LSU boundaries and the number of falsely detected ones for a given parameter combination. As it will be shown by the obtained experimental results, the quality parameter Q is rather sensitive to the number of falsely detected boundaries. This was also the main intention when we defined the function (4.3.9), since the objective of the detection procedure, presented in this section, is to provide semantically meaningful entry points into a movie. Such points can only be found at properly detected boundaries, while the number of falsely detected ones needs to be kept low. After computing the quality parameter Q for each parameter combination (B, α) belonging to ranges defined above, we sorted all values of Q and ranked them in descending order. The parameter combination having the largest Q gets the rank "1". Parameter combinations having the same value of Q are assigned the same rank.

B, α	MOVIE 1				MOVIE 2				Overall Quality Ranking
	Detected probable boundaries (out of 19)	Detected potential boundaries (out of 17)	Falsely detected boundaries	Quality Ranking	Detected probable boundaries (out of 26)	Detected potential boundaries (out of 16)	Falsely detected boundaries	Quality ranking	
40, 1.4	11	2	0	(2)	18	6	4	(6)	(4)
40, 1.5	9	1	0	(3)	18	5	3	(4)	(3)
50, 1.4	12	3	0	(1)	19	4	2	(3)	**(1)**
50, 1.5	11	1	0	(2)	18	4	3	(4)	(2)
60, 1.4	14	4	1	(4)	19	4	2	(3)	(3)
60, 1.5	12	4	1	(6)	19	5	1	(2)	(4)
70, 1.4	14	6	2	(7)	21	4	1	(1)	(5)
70, 1.5	13	4	1	(5)	20	7	4	(5)	(6)

Table 4.1: *LSU boundary-detection results for different parameter settings. Bold numbers indicate the parameter combination providing the optimal overall detection performance. Combinations with the same Q values have been assigned the same ranking.*

The first column of Table 4.1 shows all parameter combinations (B, α) used in the experiments. The other columns show for each of the movies the number of probable and potential boundaries that were detected, the number of false alarms and the ranking for each parameter combination according to the computed detection quality Q. In the final step, ranks of all pairs (B, α) obtained for both movies have been added up and the obtained results have been sorted in ascending order. The parameter combination with the lowest sum of two ranks was assigned the overall rank "1" and considered as the optimal combination for both movies.

As shown by the overall ranking list in the last column of the table, the best performance for both movies is obtained when 50% of blocks are considered for computing the overall inter-shot difference value and when the threshold multiplication factor α is 1.4. It can also be observed that the quality of a parameter combination decreases the more it differs from the optimal parameter set. This is mainly due to the influence of parameter B: if less blocks are taken into account when (4.3.8) is computed, the inter-shot comparison becomes too global, resulting in an unacceptably low number of detected boundaries. On the other hand, the large number of blocks considered in (4.3.8) can make the boundary detection too sensitive, resulting in an increased number of falsely detected boundaries.

For the chosen optimal parameter combination B=50% and α =1.4, the average percentage of detected *probable* boundaries is 69%, with only 5% of false detections. This is compatible with the requirement that, while as many boundaries as possible are properly detected, the number of falsely detected boundaries should be kept low, since they do not correspond to semantically meaningful entry points into the movie. However, absolutely seen, the obtained total percentage of 69% of properly detected boundaries for the optimal parameter combination is low. This is mainly the consequence of insufficient changes of visual features at certain episode boundaries or, in other words, of having two consecutive episodes each containing mutually similar visual content.

Table 4.1 also shows that the efficiency of the algorithm concerning the detection of probable and potential boundaries is not the same. The higher percentage of probable boundaries that were detected can be explained by the fact that those boundaries were characterized by a radical change of the scenery, which could easily be recognized by the algorithm. On the other hand, most of the potential boundaries were marked by some of the users in highly complex parts of the movies, where clearly distinguishing different episodes was a difficult task. Since our assumption about the temporal consistency of the visual content within an episode, i.e. its change at an episode boundary, was often not fulfilled in such complex movie segments, no good detection performance could be expected there.

4.4 Detecting anchorperson shots in news programs

A typical news report consists of one or several consecutive segments, each of them containing one or several concatenated video shots and belonging to one of the following categories:

- An anchorperson shot
- A news shot series (e.g. a series of shots taken by a reporter on a site, outside the studio)

Although the commercial segments can also be found in many news broadcasts, we do not consider them here, since they can easily be detected and separated from the actual news program by using any of the approaches proposed in recent literature (e.g. [Liu98]). In order to recover the next semantic level of a news-program structure, we must first classify the entire news material into one of the above two categories. Such classification is required since the beginning and the end of an anchorperson shot represents a potential report boundary which cannot be determined otherwise, e.g. by just analyzing the audio track of a news broadcast. After the classification is completed, the reports can be formed by merging related anchorperson shots and news shot series. A method to recover the report structure in this way can be found in [Hua99b] (explained in Section 4.2).

In this section, we concentrate on the problem of automatically detecting *anchorperson shots* in an arbitrary news program and propose a new approach for performing this operation. Compared to already existing anchorperson-shot detection methods described in Section 4.2, we believe our method can yield an increase in detection robustness, mainly due to the minimized usage of different thresholding parameters and, at the same time, maximal exploitation of inherent properties of the news program structure, related to anchorperson shots.

4.4.1 Assumptions and definitions

We base our anchorperson shot detection approach on the assumption that an anchorperson shot is the only type of video shots in a news program that has multiple matches of most of its visual content along the entire news program. Other (news) shots may match well only in their closest neighborhood (e.g. within a single report) where they can eventually find enough similar visual features. Such an assumption is realistic due to specific visual characteristics of anchorperson shots and their regular appearance along a news sequence. We also assume that the first anchorperson shot k_{ap} in a news program containing S video shots certainly appears within the interval $[1, N]$, where $N<S$ is assumed to be around 5 shots. In order to make the detection as robust as possible, we took into account different types of anchorperson shots, including non-stationary ones. We introduce now the following definition:

> *Anchorperson shots are visually characterized by studio background and by one or two news readers sitting at the desk, appearing separately or together, also with some possible variations of a camera angle and the magnitude of a zoom. These shots can be static or dynamic (containing some camera operations like zooming or panning). They all generally contain a certain (high) percentage of the same or similar visual features.*

During the detection procedure we compare video shots based on their *key frames*. Hereby, we assume that, prior to the anchorperson shot detection procedure, a news sequence has already been segmented into video shots, and that each shot is represented by a visual abstract consisting of a limited number of key frames. The proposed anchorperson shot detection approach consists of two steps:

- A threshold-free procedure of finding the sequence-specific template for anchorperson shots,
- Using the template to detect all anchorperson shots in a sequence by applying adaptive thresholding.

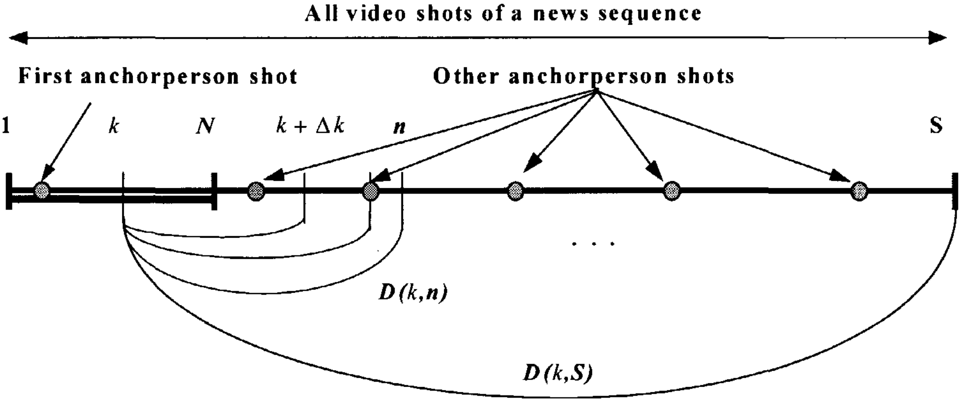


Figure 4.10: *Obtaining a dissimilarity values set for the shot k*

4.4.2 Finding a template

Based on assumptions made above, we start the procedure for finding the anchorperson- shot template by matching each shot $k \in [1, N]$, $N<S$, with all other news shots $n \in [k + \Delta k, S]$, as shown in Figure 4.10. In this way, a set of dissimilarity values $\{D(k, k + \Delta k), \ldots, D(k,S)\}$ is obtained for each shot k. The dissimilarity measure used here to compute values $D(k,n)$ compares two shots on basis of their abstracts (key frames) and is the same as the one used in the previous section. The "security" interval $[k, k + \Delta k]$ serves to avoid a possible good match of a news shot with its surrounding shots and, consequently, to separate the shot k_{ap} even stronger from the rest. For each

shot $k \in [1, N]$ we now take the P best matches (lowest values) from the set of dissimilarities and average them to compute the overall matching value. The shot with the lowest overall matching value is assumed to be an anchorperson shot, and is used as the template for finding all other anchorperson shots of a news sequence. With $k_{\min,i,j}$ being the j-th of the P shots, between which and the shot k_i the lowest dissimilarity D is measured, we find the shot k_{ap} using the following expression:

$$k_{ap} \Leftarrow \min_i \sum_{j=1}^{P} D(k_i, k_{\min,i,j}) \tag{4.4.1}$$

4.4.3 Template matching

After the template has been found, again the inter-shot dissimilarity metric *D(k,n)* is used on all shots of a sequence to test which are anchorperson shots. Low dissimilarity values will be obtained when the template is matched with another anchorperson shot. For each shot *k* of a sequence we now define its similarity with the template shot as

$$s(k) = \frac{1}{D(temp, k)} \tag{4.4.2}$$

whereby *D(temp,k)* is the dissimilarity between the template *temp* and the shot *k*. In order to perform the detection of anchorperson shots automatically, we use the similar adaptive threshold *T(k)* as in the previous chapter, defined here as the function of the similarity (4.4.2):

$$T(k) = \frac{\alpha}{N_k + 1}\left(\sum_{i=1}^{N_k} s(k-i) + s_0\right) \tag{4.4.3}$$

Here α is again a fixed threshold parameter, as in the previous section. The parameter N_k denotes the number of shots until *k* and since the last detected anchorperson shot. It also uses the similarity value s_0 computed before the last detected anchorperson shot as a bias. For each shot *k*, a value *s(k)* is available as well as the threshold value *T(k)*. An anchorperson shot is detected when *s(k)>T(k)*.

4.4.4 Experimental validation

We now illustrate the performance of the developed algorithm on the example of two news sequences produced by different broadcasting companies and having the following global characteristics:

- *Sequence 1*: 12 minutes long, 5 anchorperson shots, one news reader, first appearance in the first sequence shot,
- *Sequence 2*: 25 minutes long, 17 anchorperson shots, two news readers, first appearance in the third sequence shot.

We represented each video shot by two subsampled key frames with sizes 165x144 for *Sequence 1* and 180x144 for *Sequence 2*. The parameter setting for both sequences was $N=5$, $P=3$, $\Delta k=25$ and $\alpha=3.1$. For computing the inter-shot differences (4.3.8) we chose the dimensions of the blocks in shot images $H=W=8$ and found 70% of all blocks in a shot image to be a good value for B. With this parameter setting we will now evaluate each of the two steps separately.

	RELATIVE DISTANCE $\delta(\psi,\lambda)$	TOTAL NUMBER OF ANCHORPERSON SHOTS	DETECTED ANCHORPERSON SHOTS	FALSE DETECTIONS
Sequence 1	73 %	5	5	0
Sequence 2	17 %	17	17	1

Table 4.2: *Reliability evaluation of the template finding procedure and AP detection results*

On both sequences we applied the template-finding procedure and managed to find the proper template for each of them. Figure 4.11 shows the matching results of two template-candidates along the *Sequence 2*. We then measured the relative distance

$$\delta(\psi,\lambda)=100\left(\frac{\psi}{\lambda}-1\right)\% \tag{4.4.4}$$

between the chosen minimum overall matching value λ corresponding to the template, and the second smallest matching value ψ corresponding to the major other competitor shot for template selection. The larger the relative distance, the more reliable is the found template. Table 4.2 shows in its second column these relative distances for both sequences. The lower relative distance in the second sequence is most probably the result of the particular sequence structure, which shows an introduction for the coming reports after the first anchorperson shot. This introduction contains very similar visual information as the shots in the later parts of that sequence, which partially violates the assumptions made at the beginning of this section.

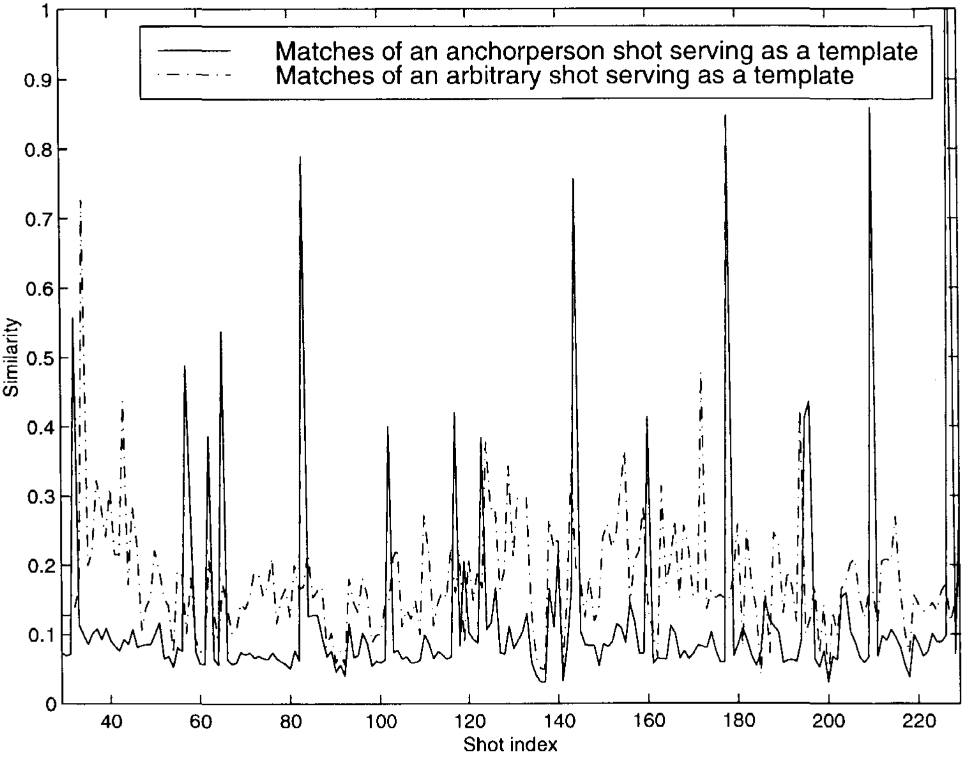


Figure 4.11: *Results of the matching procedure for two different templates* $k \in [1, N]$ *and shots* $[k + \Delta k, S]$

We then matched the found templates along the corresponding sequences to detect all anchorperson shots. The results of the template-matching procedure are given in the third and fourth column of Table 4.2 in terms of missed and false detections. Only one shot of *Sequence 2* was falsely interpreted as the anchorperson shot. This shot featured an interview between the news reader and a reporter outside the studio. Both the news reader and the reporter were positioned within their "windows" and the background of the screen in terms of its color composition fully corresponded to the studio background found in regular anchorperson shots. Similar color compositions were, thus, the most probable reason for a falsely interpretation of this shot as a regular anchorperson shot.

An idea about the robustness of the method presented in this section can be obtained by analyzing the types of anchorperson shots detected by each of the templates, and the visual content of a template itself. The first sequence contained three different variations of an anchorperson shot with one news reader. In some cases, the news reader was on the left side, zoomed in or zoomed out, with a news icon in the top right corner. In one of the shots, the news reader was in the middle of the screen and no news icon was present. This shot was also chosen as the template for *Sequence 1*.

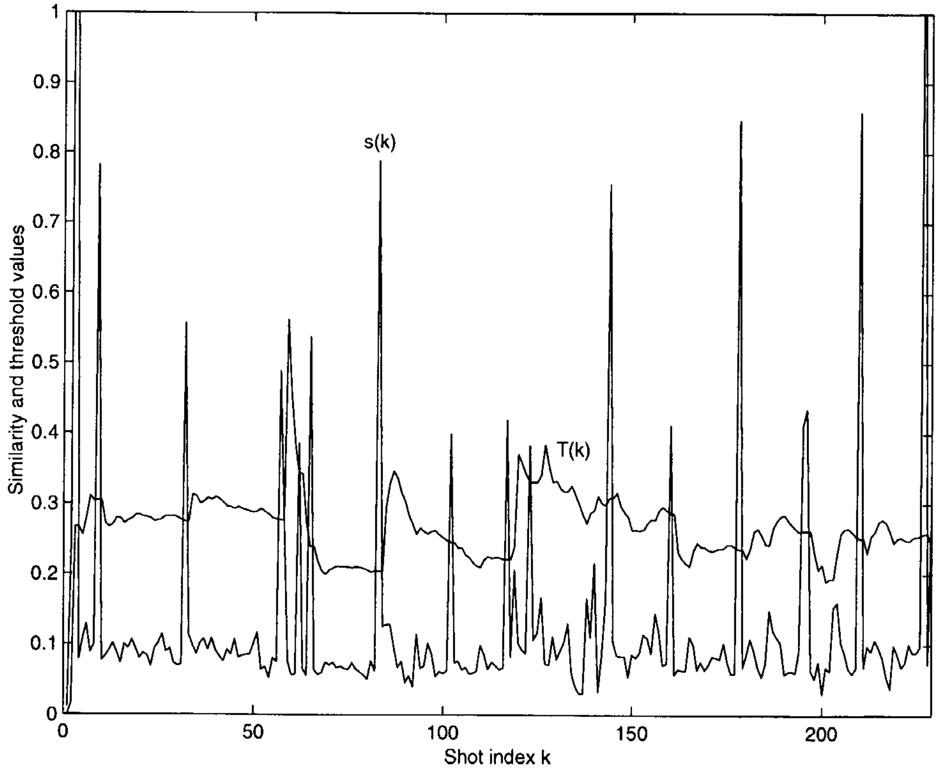


Figure 4.12: *Detection diagram for Sequence 2 and α =3.1*

All 17 anchorperson shots from *Sequence 2* were distributed as follows: 13 of them show the first (7 shots) or the second (6 shots) news reader on the right side of the screen with a news icon on the opposite side, 2 of them show the first news reader in the middle of the screen and no news icon, and 2 of them show both news readers together from two different camera positions. Two anchorperson shots were highly dynamic and characterized by a strong zoom from the studio to a news reader. As the template, one of the shots showing the first news reader on the right side of the screen was chosen.

Reliability of the detection process can be evaluated by analyzing the heights of the detection peaks in *s(k)* curves. One such curve, corresponding to the second sequence, is shown in Figure 4.12 together with the adaptive threshold *T(k)*.

4.5 Discussion

As already mentioned in the introduction to this chapter, the need for tools capable of automatically managing large amounts of information will steadily become larger with increasing volumes of video contents stored in emerging video archives. A high level of sophistication is required by such tools, since video material needs to be analyzed at the semantic level. The examples described in Section 4.2, as well as the methods for high-level analysis of movies and news in Sections 4.3 and 4.4, respectively, have shown a high potential of the low-level feature space in recovering the semantic information. This potential needs to be further exploited in the future.

In Section 4.3 an approach was presented for automatically segmenting movies into units which closely approximate actual movie episodes. The segmentation is based on an investigation of the visual content of a movie sequence and its temporal variations, as well as on the assumption that the visual content within a movie episode is temporally consistent. Consequently, an LSU is defined on the basis of overlapping links, which connect shots with similar visual content. We determine whether a link between two shots exists or not by applying an adaptive threshold function to shot dissimilarities. Based on the assumptions and definitions made in Section 4.3, the number of missed episode boundaries for a particular movie primarily depends on the degree with which an episode boundary corresponds to a large discontinuity in the global visual content flow.

Similarly, the number of falsely detected boundaries is directly related to the global temporal consistency of the visual content within an episode.

Regarding the results in Table 4.1, it can be seen that, although the percentage of the detected LSU boundaries is relatively low, the large majority of all detected boundaries indeed provide meaningful entry points into a movie. This is because the percentage of non-meaningful entries (falsely detected boundaries) is low. Since this corresponds to the objective of the approach, the results obtained for the optimal parameter combination can be considered good. A strong improvement of the performance, in terms of increasing the percentage of properly detected boundaries, is not possible by using only the visual information. We expect that involving of the audio-track analysis into the proposed procedure will be helpful. Also, the results of applying the algorithm to two movies belonging to quite different movie categories did not differ much, indicating that the detection performance, and therefore also the defined LSU model, are sufficiently consistent for different types of movies. And, finally, as the proposed technique computes the detection threshold recursively, and only looks ahead at a limited number of shots, the entire process, including the shot-change detection, key-frame extraction, and LSU boundary detection, can be carried out in a single pass through a sequence.

Reports in a news program can be considered equivalent to episodes in a movie, since they can also be retrieved separately from the news program due to their rounded context. In this sense, a report boundary is the same type of a meaningful entry point into a news program as the episode boundary is for a movie. However, while episode boundaries can approximately be determined by investigating only the visual content of a movie, this cannot be said for the report boundaries. This is due to the fact that a news report is composed out of "lossy" shots, describing the report topic from different aspects and having generally a totally different visual content. Besides this, also no visual content can be related to a certain topic. An example for this is a report about a soccer match consisting of 4 higher-level segments: an anchorperson shot characterized by a news reader and a studio background, a series of shots from the soccer field, another anchorperson shot and the series of shots showing the press conference.

The furthest we can get by analyzing only the visual content of a news program is detecting the anchorperson shots. This is because anchorperson shots are characterized by a relatively constant visual content along the entire news program. A technique developed for the detection of anchorperson shots was demonstrated in Section 4.4. As shown by experimental results, the detection can be performed with acceptable reliability under the given assumptions. The most important assumption is that no shot of a news sequence other than anchorperson shot can be used to find *P* good matches along the entire sequence. And indeed, a definite probability for failure of this condition can be the major reason for lowering the algorithm's robustness in a general case, which can be observed on a lower relative distance for the second sequence in Table 4.2. We believe that this problem can be solved by further improving the inter-shot dissimilarity metric so that different types of anchorperson shots are distinguished better from the rest of the sequence, while at the same time it allows for variation among these types.

Chapter 5

Trends in Image Coding: The "Fourth Criterion"

5.1 Introduction

For a long time, a considerable scientific and technical effort has been invested in the development and improvement of high-quality image CODECs[*] with respect to three important classical criteria

- minimization of bit rate
- minimization of image distortion
- reaching balanced/low computational costs on both the encoder and decoder side.

However, many recent developments and technical achievements in the field of multimedia signal processing have created applications for which the existing and widely used image CODECs may no longer be optimal. For instance, during an image classification or a query procedure, a large number of image comparisons needs to be performed, based on any of the criteria specified by user. As discussed in previous chapters of this thesis, the user's cognitive criteria, on the basis of which the image classification or query is to be performed, need to be transferred to the domain of spatial image features. This may require various image analysis operations for feature extraction (e.g. texture extraction, edge detection, shape-information extraction by image segmentation, image or image-region histogram computation). For many of these operations access to the full-

---

[*] CODEC is a common abbreviation for a joint COder-DECoder system.

resolution spatial image content is needed, which is not a problem if the images in a database are available in their uncompressed (e.g. RGB or YUV) format. However, due to large information volumes, databases are likely to contain *compressed* images, where in most cases currently available image encoders based on image transformations such as DCT (JPEG), subband or wavelet are used.

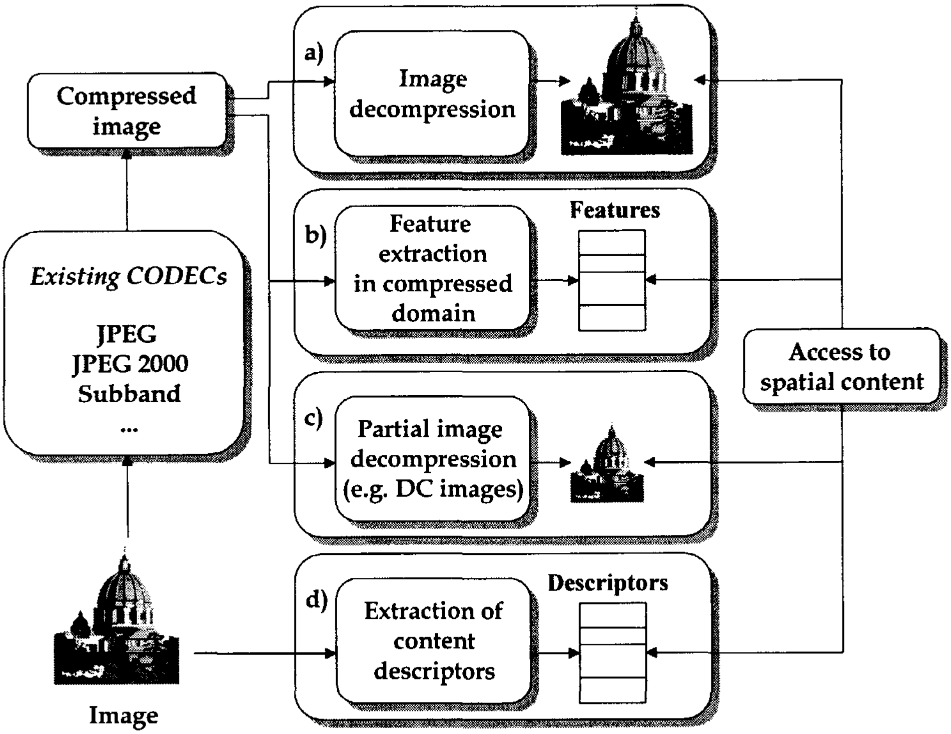
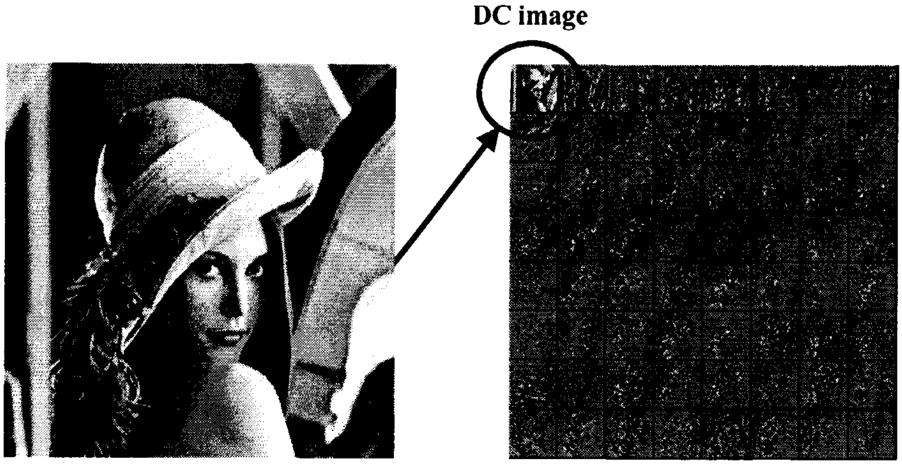

Compressed image
a)
Image decompression
b)
Feature extraction in compressed domain
Features
Existing CODECs
JPEG
JPEG 2000
Subband
...
Access to spatial content
c)
Partial image decompression (e.g. DC images)
d)
Extraction of content descriptors
Descriptors
Image

Figure 5.1: *Current possibilities to reach the spatial content in compressed images: a) full image decompression, b) feature extraction in compressed domain, c) partial image decompression, d) content access using content descriptors (MPEG-7)*

In principle, there are several possibilities to reach spatial image content in images, which are compressed using these encoders. These possibilities are given in Figure 5.1. The first possibility is to decompress an image and then to perform any feature-extraction operation. The disadvantage of this option is that the complexity of the decompression procedure, when combined with a large number of images in a database, can negatively influence the efficiency of the interaction with a database to a large degree. We can illustrate the complexity of the decompression procedure with the example of the JPEG CODEC: decompressing a JPEG image involves a series of operations such as Huffman decoding, run-length decoding, Zig-Zag decoding, dequantization, DPCM decoding of DC coefficients and the block-wise inverse DCT.

In order to avoid full image decompression, one can attempt to develop algorithms for extracting spatial features from compressed images directly. Results are reported in [Smi94] on using the properties of an image signal in the transform domain (DCT, Subband transform, Wavelet transform) to approximate texture features. Transformed signal properties used are, for instance, energies or first-order moments in each subband. In general, considerable research has been done in compressed-domain image processing and manipulation. Some characteristic approaches developed for this purpose can be found in [Smi93], [Cha95a] and [Cha95b]. The proposed algorithms are reported to perform efficiently, which was to be expected with respect to some specialized applications. However, the compressed-domain approaches show some inherent disadvantages. First, algorithmic solutions proposed are not universal, meaning that for each of the current and newly introduced compressed formats different algorithms are to be developed and applied. Second, since currently available and widely used image encoders are not designed with the objective to ease the access to spatial information in the compressed domain, there are limits in developing the algorithms mentioned above for reaching all possible features which may be required for image-database operations. Even if such algorithms can be developed, their complexity is likely to be greater than the one involving image decompression and performing the feature extraction on "raw" images.

Another alternative to full image decompression is *partial* decompression. For instance, a JPEG encoder allows an easier access to a low-resolution (subsampled) image version, as shown in Figure 5.2 on the example of a "Lena" image. The subsampled version obtained from a JPEG-compressed image consists of collected DC coefficients of all DCT image blocks, and can be obtained after performing the JPEG-decompression steps preceding the inverse DCT. In this way, some global aspects of the spatial image content can be obtained by avoiding the complex procedure related to the inverse DCT. Although the image content is perceptually recognizable when one looks at the subsampled version of an image, the possibilities for performing image analysis and processing operations on the subsampled image are limited. This is mainly due to missing high-frequency components and small dimensions of subsampled image versions (e.g. eight times smaller height and width in case of the JPEG-DC image).



"Lena" image

Grouped corresponding DCT coefficients

Figure 5.2: *Subsampled versions of the original image, which are obtained after partial image decompression*

Another possibility for getting access to the spatial content on compressed images more easily is to extend current compression standards by "content descriptors", which is, for instance, done in MPEG-7 [ISO97]. The descriptors can be of various types, and are available as a "side information" to compressed image streams; they are meant to represent certain aspects of the spatial image content. Then, any query or classification operation on images from a database can be performed using the weakly coded descriptors, without any need for decompressing the images themselves. This alternative has the disadvantage that the descriptors reveal only certain aspects of the spatial image content and cannot take into account all possible image features required for an arbitrary query or classification scenario, applied to a database. Thus, while it is highly practical for specific applications, this alternative is not sufficiently general to ensure unconstrained interaction with an image database.

As discussed above, there are clear limitations where it concerns efficiently accessing the spatial content in images compressed using currently available and widely used CODECs. For this reason we see the development of new image CODECs having an additional optimization criterion, namely taking into account the spatial-content accessibility, as the most appealing long-term solution for maximizing the efficiency of interaction with an image database. The proposal of a novel image CODEC which complies with this solution is the main objective of this chapter.

In Section 5.2, we first describe the basic principles on which the proposed CODEC is based. Then, all the components of the CODEC scheme are defined in Section 5.3, which is followed by performance evaluation in Section 5.4 and a discussion to this chapter in Section 5.5.

5.2 A concept of an alternative image CODEC

The need for considering an additional *fourth criterion* in a CODEC-development procedure has already been pointed out several years ago [Pic95a], [Pic95b]. New image CODECs envisioned there need to be developed, where not the bit rate, distortion and computational costs must be taken into account, but also the minimization of the *content access work* [Pic95b]. However, still no concrete proposal has been made with respect to a practical way of considering the fourth criterion during a CODEC development.

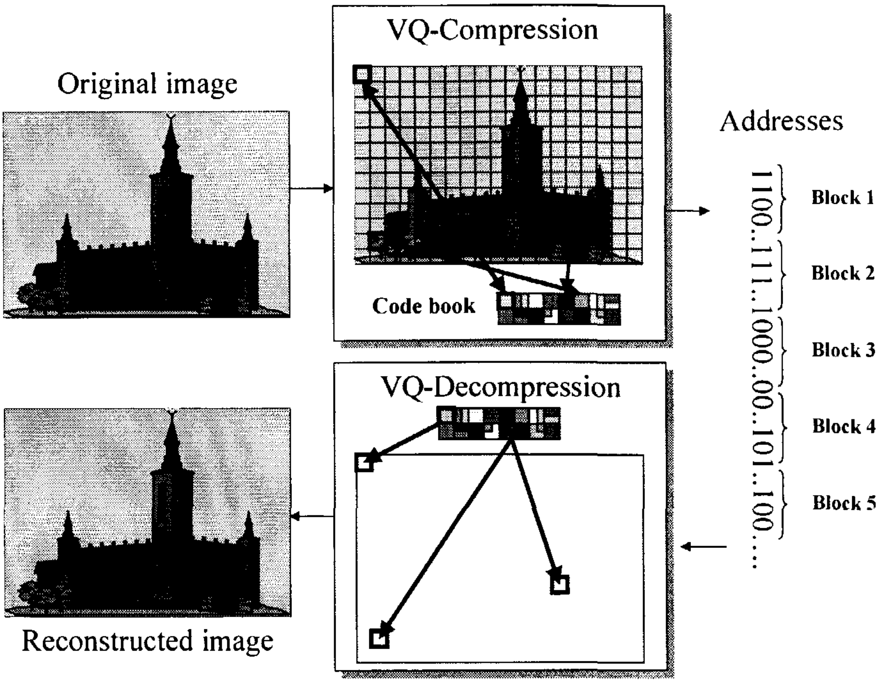


Figure 5.3: *Scheme of an image CODEC based on vector quantization*

We approach this development by considering the following issues. First, the investigation of transform-based CODECs in the previous section has shown that the image transformation applied there (DCT, subband, wavelet) is one of the major obstacles in reducing the complexity of accessing the spatial image content. Therefore, here we concentrate on *spatial-domain* image compression techniques, examples of which are Vector Quantization (VQ) [Ger92] and Fractal Image Compression [Jac92]. Second, it is not realistic to expect that a full-resolution spatial image content is available in the compressed domain if a good compression ratio needs to be obtained. This is simply because of the fact that the compression is based on reducing the redundancy and the irrelevancy of this content, so that only non-redundant and relevant content components are available in the compressed format. However, in order to obtain an increase in the efficiency of image-database operations we require that non-redundant and relevant image information contained in the compressed format is already usable for performing some of the image-database operations. We also require that the full-resolution spatial image content should quickly be reconstructable from its non-redundant and relevant elements, or in other words, that the complexity of the image-decompression procedure is considerably reduced if compared to transform-based decoders. In the following, we first briefly describe the fractal image compression and vector quantization and subsequently choose the most suitable of the two techniques as the base for developing our CODEC.

In case of fractal compression, an image is first partitioned at two different levels: in range blocks of size *NxN* at the first level, and in domain blocks of size *2Nx2N* at the second level. A transformed domain block is searched for each range blocks such that the mean square error between the two blocks is minimal. Hereby the following transformations are performed on the domain blocks: they are first subsampled by factor two to get the same dimensions as the range blocks. Then, eight isometries of subsampled domain blocks are found, including the rotated original block and its mirrored versions (mirroring over 0, 90, 180 and 270 degrees). Finally, an adjustment of the scale factor and the luminance offset is performed. Consequently, a fractal compressed image is defined by a set of relations for each range block, the index number and the orientation of the best fitting domain block, the luminance scaling and the luminance offset. Using this description, the decoder can reconstruct the compressed image by taking any initial random image and by calculating the content of each range block from its associated domain block. This reconstruction is repeated iteratively by taking the resulting image as a new initial image until the desired quality of the reconstructed image is reached.

As illustrated in Figure 5.3, compressing an image using VQ is the process of taking an image block of *NxN* pixels and finding its corresponding (most similar) block in a *code book.* A code book is a collection of representative blocks, constructed on the basis of a number of training images. Each image block is then represented by the code-book address, where the corresponding block is found. Consequently, a VQ-compressed image is simply a concatenation of addresses, collected for all image blocks. If the same code book is available at the receiver side, a VQ-compressed image can easily be decoded by filling in the blocks from a code book in the proper positions in the image, according to the addresses received by the decoder.

Because of the above descriptions of Fractal and VQ image CODECs, we find the CODEC based on Vector Quantization more suitable for our needs. First, it realizes image decompression as a fast "look-up-and-fill" procedure and involves no iterations. Second, VQ-compressed images can be compared and classified based on their block correspondences. This is because these correspondences directly depict the spatial image content with respect to the code book used. Compared to this, a list of geometric and luminance transformations of domain blocks describing the Fractal-compressed images do not provide a clear impression about the spatial-image content and, therefore, cannot be used as efficiently for image-database operations as the block correspondences and code book of the VQ. However, not all the characteristics of the basic VQ scheme are suitable for direct usage for the CODEC development in this chapter. Therefore, we adapt the basic VQ scheme in order to better suit the applications addressed here.

The adaptation is related mainly to the highly complex and time-consuming process of code-book generation. This process basically includes a *partitional clustering* of the visual material collected from a set of *training images*. Its high complexity is due to a large amount of data to be clustered and due to the iterative nature of the clustering process. Consequently, the code book is made only once and used to compress and decompress all images in a database. It is also optimized to provide the maximal quality of all reconstructed images from that database. This optimization is performed such that, first, the training images are selected as the most representative for all the images contained in a database. Second, the clustering process is designed to take into account all linear and non-linear dependencies among blocks to be found in training images. Each cluster is then represented by one most representative image block, which then becomes an element of a code book. The described process of code-book generation by the basic VQ implies that the code book can be used effectively only for compressing images that belong to the same categories as those from the training set. This is, however, unpractical for applications in general image databases because of the following reasons. First, images can be very diverse, so that one single code book might not be sufficient for coding all of them with an acceptable quality. Second, if a database is extended by new images belonging to a different class, a time-consuming update of a code book is required. Third, image exchange among different databases (users) is difficult if different code books are used.

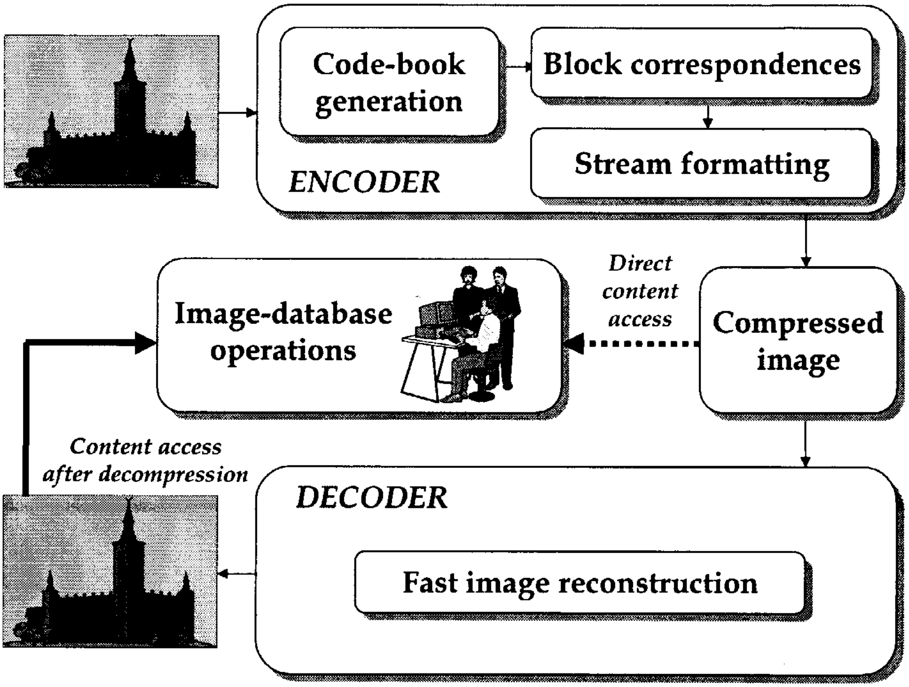


Figure 5.4: *Image CODEC enabling easy access to spatial content in compressed images*

To provide an effective solution to these problems, we apply in our CODEC a strongly simplified procedure for code-book generation, which - due to a reduced complexity - allows for generating a code book for each individual image. Using image-specific code books not only makes it unnecessary to perform highly complex code-book generation/update and to have one code book for the entire database; it has several other important advantages as well. First, the quality of reconstructed images can only improve since an image is abstracted and later reconstructed using the same blocks. Second, in contrast to the basic VQ, here the code book needs to be included in the compressed image format. As will be shown in Section 5.4, this makes it even easier to perform various image-database operations without the need for image decompression.

5.3 Image CODEC based on simplified VQ

Figure 5.4 illustrates all components of our new image CODEC in form of a block diagram. The first two steps of the encoder are the processes of making a code book and of finding the correspondences of image blocks with those belonging to a code book. Subsequently, a compressed image stream is formatted, where we only use as many bytes as necessary to encode all the addresses in order to minimize the resulting bit rate. Apart from the fact that the code book used is image specific and therefore included into the compressed stream, the decompression process fully complies with the one of the basic VQ.

As indicated in the scheme by the full arrow, a low computational complexity on the decoder side provides one possibility to reach the spatial image content easily. The other content-access possibility indicated by the dashed arrow is related mainly to the direct usage of the image-specific code book and block correspondences for image-database operations. The issues regarding the content accessibility will be discussed in detail in Section 5.4. In the following we proceed by defining all major components of the CODEC scheme in Figure 5.4.

5.3.1 Code-book generation

We first define an efficient methodology for generating an image code book. For this purpose, an image is first divided into non-overlapping square pixel blocks b_i with dimensions NxN, and each block is represented by the average color ($L^*u^*v^*$ color space) of all block pixels. We choose to work with relatively small blocks, i.e. with $N=2$ as the code book will have to be included into the compressed image format. The experiments have shown that if a similar quality of reconstructed images is to be achieved, and larger blocks, e.g. $N=4$, are used, the code-book size becomes unacceptably large (up to 20% of an image).

As shown in Figure 5.5, the code-book generation starts by including the first image block b_1 into the code book. Each further block along the arrow is compared with all blocks already in the code book. For this purpose, the Euclidean distance is computed for the three components of the average colors of blocks, that is

$$d(b_k,b_n)=\sqrt{\left(L^*(b_k)-L^*(b_n)\right)^2+\left(u^*(b_k)-u^*(b_n)\right)^2+\left(v^*(b_k)-v^*(b_n)\right)^2} \qquad (5.3.1)$$

A block b_i joins the code book if it cannot be matched well with any of already selected blocks, i.e. when the distance (5.3.1) between the block b_i and each block from the code book exceeds the threshold T.

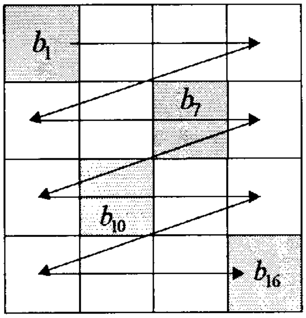
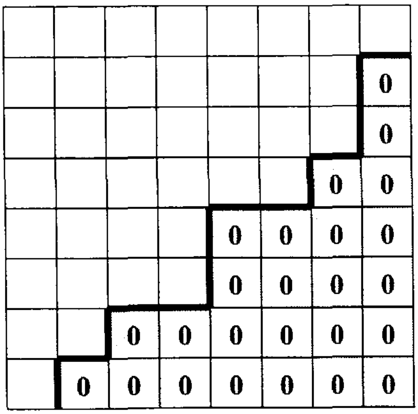


Figure 5.5: *Illustration of the simplified procedure for making a VQ code book. Grey blocks are included in the code book.*

While the code book of the basic VQ is obtained by a sophisticated procedure which optimally represents the visual material of training images, the major objective of the sequential procedure from Figure 5.5 is to *quickly* generate a code book. In order to achieve a code-book quality similar to that of the basic VQ, the described fast procedure for code-book generation requires some *fine tuning*. For this purpose, we make the threshold T locally adaptive, based on the following analysis. Since coding artifacts are particularly visible in smooth image regions (e.g. artifacts like *false contours*), the threshold function needs to be chosen such that these regions are represented by a sufficient number of code-book vectors. On the other hand, the number of blocks extracted from textured regions can be kept low since the coding artifacts are less visible there. This implies that it is convenient to make the threshold value at each block b_i dependent on the amount of texture present in its surroundings, that is

$$T=T(b_i)=f(texture\ around\ b_i) \qquad (5.3.2)$$

By properly choosing the threshold function (5.3.2), a number of code-book blocks can be extracted that is similar to using a fixed value of T. However, the extracted blocks are distributed better over an image, providing at a later stage a higher overall quality of the reconstructed image. In other words, the number of code-book vectors representing textured image regions slightly decreases in favor of those representing smooth regions.



							0
							0
						0	0
				0	0	0	0
				0	0	0	0
		0	0	0	0	0	0
	0	0	0	0	0	0	0

Figure 5.6: *Zero coefficients in a DCT block obtained by applying the quantization*

We now define the threshold function $T(b_i)$ by suitably modeling the variations of the amount of texture over an image. This is done by first dividing the gray-scale version of an image into nonoverlapping blocks B_j with dimensions 8x8 pixels, and by applying the Discrete Cosine Transform (DCT) to each of them. Then, all the elements of the DCT block are quantized according to the following procedure, which is analog to the one from JPEG:

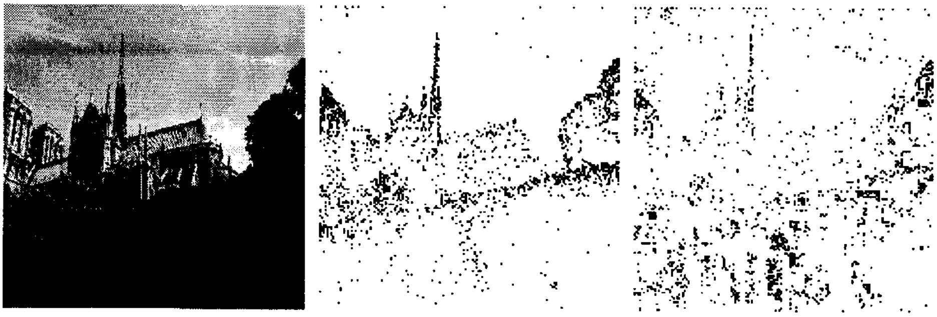
$$ROUND\left(\frac{DCT(u,v)}{Q(q)*W(u,v)}\right), \quad \text{with} \quad Q(q)=\begin{cases}\dfrac{50}{q}, & q<50\\[2ex] \dfrac{100-q}{50}, & q\geq 50\end{cases} \tag{5.3.3}$$

We call q the *quantization parameter* which can vary in the range $0<q<100$. Q is the gain factor depending on q and $W(u,v)$ is the corresponding element of the JPEG luminance quantization table.

As a consequence of the quantization, a number m_j of DCT coefficients of the block B_j will become zero, which is mainly the case with those corresponding to higher frequencies, as shown in Figure 5.6. The more texture is present in an image area, the stronger are the high-frequency components of the image signal in that area. Then, the DCT coefficients corresponding to these components are also high, and therefore will hardly ever become zero after quantization. This is not the case with smooth regions, where a large number of zero coefficients are present after the quantization. For this reason we relate the number m_j of zero-DCT coefficients to the presence or absence of a texture and formulate the threshold function as follows:

$$\text{if} \quad b_i \in B_j \quad \Rightarrow \quad T(b_i) = p_1 - p_2\left(e^{\frac{m_j - 64}{p_3}}\right) \tag{5.3.4}$$

Parameters p_1, p_2 and p_3 define the behavior of the threshold function and are to be specified experimentally. Since the parameter q directs the DCT quantization process (5.3.3), the threshold-function behavior can indirectly be adjusted by specifying a value for q. The higher the q, the lower is the gain factor Q, the smaller is the quantization step, the less DCT coefficients are zero and the threshold function (5.3.4) is shifted upwards.



a) b) c)

Figure 5.7: *Code book extraction for an image using a constant and a variable threshold: a) original image, b) image blocks included in the code book by using a constant threshold, c) image blocks included in the code book by using the threshold function (5.3.4)*

Figure 5.7a shows an image from our test set, for which we generated two code books of a similar size. The first one is obtained by using a fixed threshold. The positions from which image blocks are taken and included into the code book are indicated as black spots in Figure 5.7b. Then we used the variable threshold to generate another code book, where we adjusted the function (5.3.4) by choosing a suitable q such that a similar number of blocks is extracted, as in Figure 5.7.b. The positions from where image blocks are taken using the threshold (5.3.4) are indicated by black spots in Figure 5.7c. It can be seen that the prevailing majority of blocks in Figure 5.7b are concentrated in high-texture regions, leaving the smooth regions insufficiently represented.

5.3.2 Finding the block correspondences

The step of generating the code book is followed by the search for correspondences between image blocks b_i and blocks c_j of the code book. In this way, each block b_i is represented by an address in the code book, which is embedded into the compressed image format and determines which block c_j is used to approximate block b_i during the image reconstruction in the decoder.

We find that block c_j corresponds to the image block b_i by comparing b_i with all blocks c_j using the distance function (5.3.1) and then by minimizing (5.3.1) for all indices j. As S is the total number of blocks b_i in an image, and M the number of code-book blocks, the procedure of establishing the block correspondences can analytically be formulated as

$$\forall i \in [1,S] \quad b_i \propto c_t \quad \Leftrightarrow \quad d(b_i,c_t) = \min_{1\le j\le M} d(b_i,c_j) \tag{5.3.5}$$

5.3.3 Compressed image format specification

The format of an image, compressed using the CODEC presented in this chapter, is illustrated in Figure 5.8, and consists of the following information:

- File header
- Code book
- List of addresses for block correspondences

We will now define each of these components in detail.

File header

The file header contains 10 Bytes, which represent (in the order of appearance):

- the format identification mark (3 Bytes),
- image width and height (4 Bytes)
- the number M of blocks in a code book (3 Bytes)

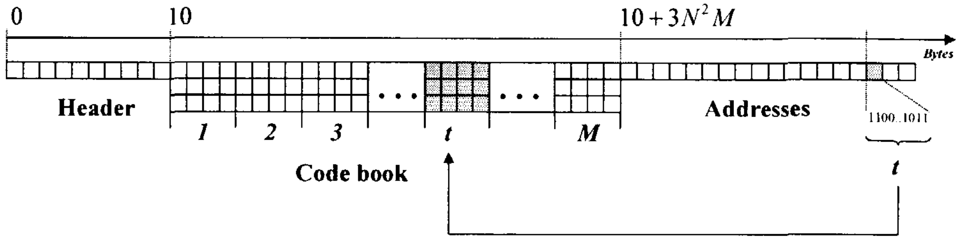


Figure 5.8: *Format of a compressed image file*

Code book

All code-book blocks $c_j,\ j \in 1..M$ are addressed in the order in which they are extracted. In the compressed bit stream, they are represented by the RGB triplets of all of their pixels, ordered in the 1-dimensional uncompressed array of $3MN^2$ Bytes, that is

$$R^1_{11}G^1_{11}B^1_{11}\ldots R^1_{1N}G^1_{1N}B^1_{1N}\ldots R^1_{NN}G^1_{NN}B^1_{NN}R^2_{11}G^2_{11}B^2_{11}\ldots R^2_{NN}G^2_{NN}B^2_{NN}\ldots R^M_{NN}G^M_{NN}B^M_{NN} \tag{5.3.6}$$

Block correspondences

The code book is followed by the list of addresses for block correspondences. For the total number of S blocks b_i in an image, there are S addresses varying in the range $1..M$. In order to reduce the size of this bit stream component, we use only so many bytes as are necessary to represent all addresses of characteristic blocks. For M blocks in a code book, the minimum required number w of bytes is computed as

$$w = \frac{1}{8}\left(\lfloor \log_2 M \rfloor + 1\right) \tag{5.3.7}$$

5.4 Performance evaluation

In this section we evaluate the performance of the developed image CODEC. We concentrate in Subsection 5.4.1 on the CODEC performance with respect to the obtained compression factor, the quality of reconstructed images and the overall computational costs. For this purpose we use a test-image set containing 54 different color images with dimensions 320x320 pixels. We experimentally found good parameter values in (5.3.4) as p_1=5.35, p_2=4, and p_3=3, and used them in our experiments. Subsequently, in Subsection 5.4.2 we evaluate the possibilities for easy access to spatial image content using CODEC we developed.

5.4.1 CODEC performance regarding classical criteria

We first investigate a typical range of the compression factor, which is to be obtained using our CODEC. For this purpose we took one image from our test set and compressed it for values of the quantization parameter q in (5.3.3) varying between 5 and 95. The compression factor as a function of the parameter q is displayed in Figure 5.9a. The obtained range for this factor is [4.25, 7.93] for the test image used.

Then, we took the same test image and measured the PSNR (Peak-to-Peak Signal-to-Noise Ratio) for R, G and B color component over the entire range of the compression factor in order to see how the quality of reconstructed images depends on compression efficiency. We averaged the PSNR values of the three color channels at each measurement point and displayed them over corresponding compression factors in Figure 5.9b. A range of the average PSNR values was obtained as [30.74, 34.57]. In order to get a better impression of the above results, we also compressed the same test image using JPEG. We let the JPEG quality factor vary in its entire effective range from 5 to 95 and obtained the compression factors between 9.02 and 94.3 and PSNR between 24.62 and 39.97. Especially for the range of the compression factor obtained for our CODEC, the PSNR varied in the case of JPEG compression between 39.97 and 43.6, as also shown in Figure 5.9b. A comparison of the curves in Figure 5.9b indicates that JPEG performs better in terms of compression efficiency and resulting image distortion.

Besides compression efficiency and image distortion, the classical criteria also include the overall computational complexity. Compared to the basic VQ, the JPEG CODEC is characterized by well-balanced computational costs on the encoder and decoder side, which makes it more practical.

Thus, in order to make the VQ competeable with JPEG regarding the cost balance, a strong reduction of the encoder complexity would be required. This was one of the objectives when developing the methodology for a simplified code-book generation in Subsection 5.3.1. Although we managed to considerably reduce the encoding complexity in our CODEC compared to the basic VQ, this complexity is still relatively large. This is mainly due to small block dimensions, which, as explained before, were chosen such to increase the compression efficiency. For instance, only 4% of image information, contained in the code book of an image with dimensions 320x320 pixels, corresponds to 1024 blocks with dimensions 2x2 pixels. Consequently, for each of the 25600 blocks of that image, 1024 computations of difference values (5.3.1), threshold (5.3.4) and their mutual comparisons are required.

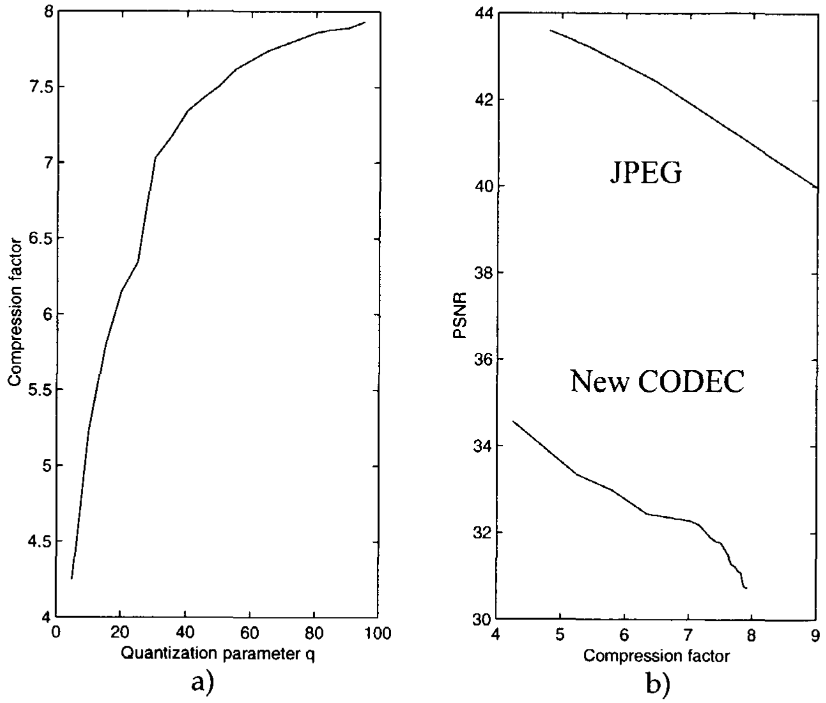


Figure 5.9: *Measurements for one test image: a) Compression factor as function of q, b) average PSNR of the R, G and B color component as a function of the compression factor*

The above comparison of our CODEC and JPEG regarding the classical optimization criteria has shown that JPEG has a better performance. We, however, took this into account with a reference to the fact that a CODEC that performs well regarding the three classical criteria is not necessarily optimal when it comes to the fourth criterion: providing easy access to spatial image content [Pic95b].

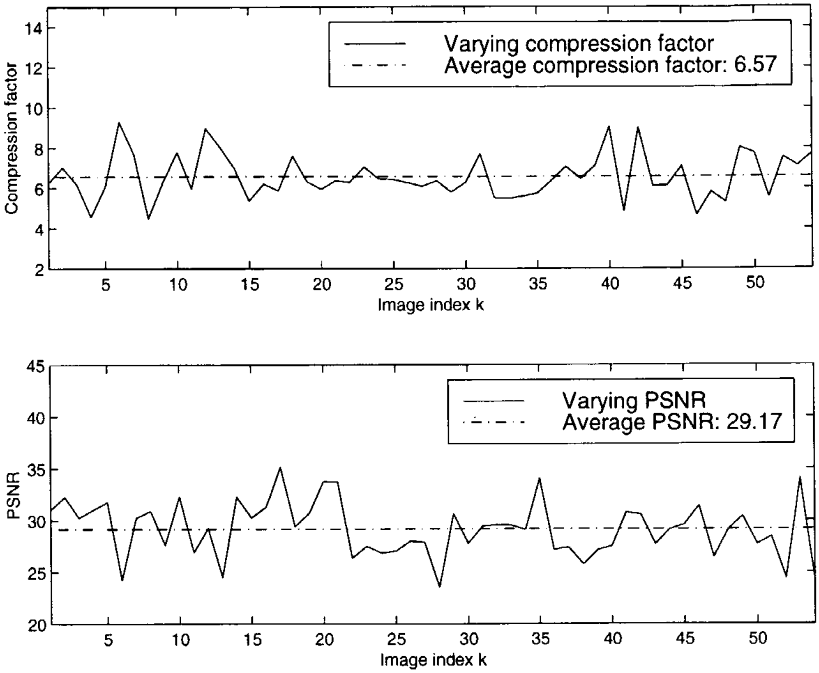


Figure 5.10: *Variations of the compression factor and PSNR for all test images and q=30*

To complete the evaluation of the CODEC performance regarding the compression efficiency and image distortion, we also investigated the consistency of the CODEC performance regarding these criteria for different images. For this purpose we fixed the quantization parameter q to the value 30 and computed the compression factor and PSNR for all images from our test set. The results are displayed in Figure 5.10. The variations of computed values can be explained by the variability of the code-book size, which depends on a spatial image content and is the only variable segment of the compressed image stream. In our measurements, the relative code-book size varied between 1.4% and 9.78%, with an average of 4.6%. Perceptual quality of the reconstructed images for q=30 can be evaluated by comparing the originals and decompressed images in Figures 5.11a-c.

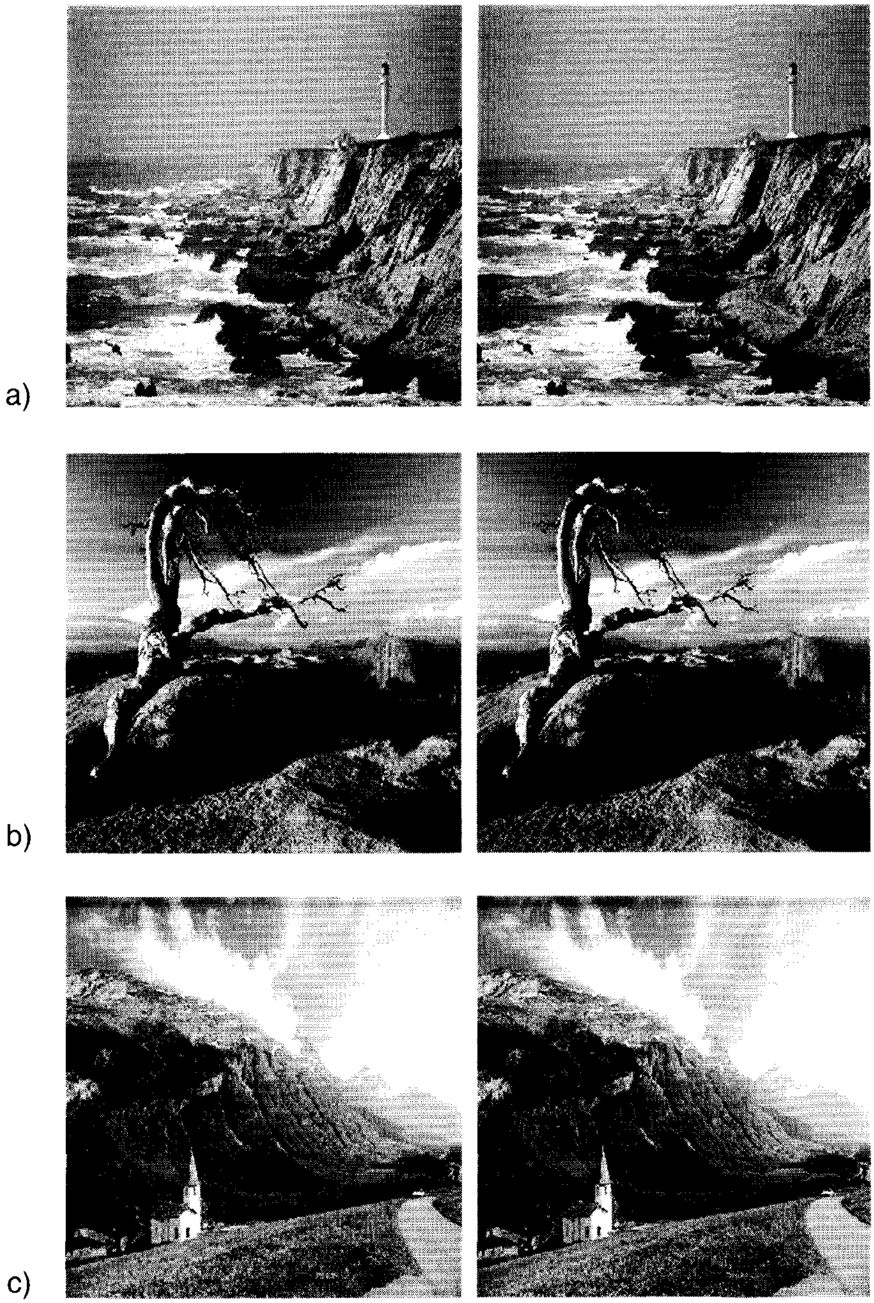


Figure 5.11: *Reconstructed images followed by the originals. The following quantitative data were obtained for q=30: a) code-book size 2%, compression factor 8, b) code-book size 5.3 %, compression factor 6, c) code-book size 4.1%, compression factor 6.4*

5.4.2 Reduction of the content access work

We address in this section some of the possibilities for easy access to spatial content in images which are compressed using the CODEC developed in this chapter. These access possibilities are:

- Easy image or image-region decompression
- Easy access to some spatial image features directly in the compressed domain (dominant and less important colors, image and image-region histograms)

An advantage of having the VQ as the underlying principle of the CODEC presented here is that decompressing an image is a simple "look-up-and-fill" procedure. Namely, a VQ-compressed image can easily be decoded by filling it with blocks from a code book, according to addresses received by the decoder. This is a clear advantage regarding the computational complexity of the decoder, if compared i.e. with the JPEG decompression procedure, containing amongst others entropy decoding, dequantization and inverse DCT. Further, since the list of addresses in the compressed image format (Figure 5.8) preserves the information about image structure, no decompression of the entire image is required in order to fully reconstruct any of its regions. Such a reconstruction is easily performed by simply choosing the region blocks in the address list, finding their corresponding blocks in the code book and filling the image regions of interest. However, an image, or any of its regions does not need to be reconstructed in order to obtain certain spatial image features; an image-database application involving these features can be performed directly on compressed images.

As a first example, the image-specific code book itself, which is directly accessible in the compressed image format (Figure 5.8), can effectively be used for performing global classification of images or some more general image queries. In our approach, a block is selected in a code book if the average color of its pixels is sufficiently representative for that image. This implies that if average colors are computed for all code-book blocks c_i, a general idea can be obtained about the colors present in the image. In this way, images containing drastically different color content can be separated from each other, or – if different from the query image regarding their colors - not given by the system as query results.

Image classifications and queries based on image-specific code books can be made even more specific if also the information related to the usage of code-book blocks for image reconstruction is taken into account. This information can easily be retrieved from the list of addresses for block correspondences. Then, by counting the numbers of times a_i, that a code-book block c_i is present in the address list, the image and an arbitrary image region can be represented as

$$image \Leftrightarrow [a_1c_1,..a_Mc_M] \tag{5.4.1a}$$

$$image\ region \Leftrightarrow [a_mc_m,..a_nc_n] \tag{5.4.1b}$$

After the average color of each block c_i has been computed, the expressions (5.4.1a-b) can provide a global idea about the color composition of the image (region), both in terms of which general colors are present there and in which amount. The higher the number a_i, the more important role plays the block c_i in the image (region), and thus also the average color of its pixels, that is

$$\max_i a_i = a_{\text{dominant}} \Rightarrow c_{\text{dominant}} \Rightarrow \text{dominant average color} \tag{5.4.2}$$

As an example, we now estimate the computational complexity of obtaining the information on a general color composition of an image based on (5.4.1a) and on computing the average color of each block c_i. We estimate the complexity by determining the number of reading (O_{read}), adding (O_{add}), multiplying ($O_{multiply}$) and comparing ($O_{compare}$) operations, which are to be performed.

With S addresses to be found in the last segment of the compressed image format in Figure 5.8, the number of operations required to obtain all the coefficients a_i can be estimated as follows:

$$C_{a_i} = S\left(O_{read} + O_{add}\right) \tag{5.4.3}$$

Then, the average color is computed for each block of the code book with the following amount of operations:

$$C_{c_i} = 3\left(N^2 O_{add} + O_{multiply}\right) \tag{5.4.4}$$

The entire complexity of obtaining the information on a general image color composition complying with (5.4.1a) is now given as

$$C_{\text{color composition}} = C_a + MC_{c_i} + MO_{multiply} \tag{5.4.5}$$

As another example, histograms for image (regions) can easily be computed by collecting pixels of blocks c_i and taking into account the values a_i. Here, we compute the bins h of an 1-dimensional color histogram $H(h)$, where h can be the value of any pixel-color component K ($K=R$, G or B) and where only characteristic blocks c_i, used for reconstruction of an image region (5.4.1b), are considered:

$$H_{R,G,B}(h) = \sum_{i=m}^{n} a_i v_h(i)$$

with (5.4.6)

$$v_h(i) = \begin{cases} l_{i,h}, & pixel(K=h) \in c_i \\ 0, & otherwise \end{cases}$$

The function $v_h(i)$ indicates whether a pixel with its K color component corresponding to h is present in the block c_i, and in which amount $l_{i,h}$. Also by counting the number of operations required for the expression (5.4.6), we estimate the complexity of obtaining an image region histogram directly from the compressed image in Figure 5.8 as

$$C_{hist} = C_{a_i} + (n-m)\left(N^2\left(O_{read} + O_{compare}\right) + \left(l_{i,h} + 1\right)O_{add} + O_{multiply}\right) \tag{5.4.7}$$

We now like to compare the C_{hist} from (5.4.7) with the complexity of computing the same histogram on the decoded image. This last complexity is given as:

$$C_{hist,decompressed} = C_{decompressed} + XY\left(O_{read} + O_{compare} + O_{add}\right) \tag{5.4.8}$$

Since the size of the code book M is only a small fraction (average of 4.6% in our tests) of the total number of pixels in an image obtained by multiplying both image dimensions X and Y, and since $0 \le l_{i,h} \le N^2$ with $N=2$, the second summand in (5.4.8) can be considered considerably larger than the second summand in (5.4.7). Further, if, for instance, JPEG CODEC is used as alternative, the first summand $C_{decompressed}$ in (5.4.8) includes the number of operations required for Huffman decoding, run-length decoding, Zig-Zag decoding, dequantization, DPCM decoding of DC coefficients and the block-wise inverse DCT. As such, this summand can realistically be assumed far larger than the value C_{a_i} in (5.4.7). Therefore, it can be said that the histogram computation using (5.4.6) is computationally considerably less expensive than if performed after e.g. a JPEG-compressed image is decoded.

5.5 Discussion

The image CODEC presented in this chapter was developed to suit emerging applications on large-scale image databases, where a fast and easy access to spatial image content can considerably improve the efficiency of interacting with an image database. While the currently available CODECs are optimized with respect to the classical criteria (bit rate, image distortion and overall computational complexity), introducing an additional fourth criterion related to providing easy access to spatial image content has the effect that existing CODECs are no longer optimal and, as discussed in Section 5.1, that the development of new CODECs is needed. It would be best if we could retain an excellent compression efficiency, low distortion of reconstructed images and nicely balanced and low computational complexity of JPEG in newly developed CODECs and still be able to easily perform any operation on spatial image content. Such a perfect balance among the four optimization criteria needs indeed to be the guiding objective of research in this area. The development of the CODEC in this chapter can be understood as a first step in the process of reaching this objective. We deliberately left the powerful concept of transform-based CODECs in order to remain in the spatial domain and so to provide means for accessing the spatial image content more easily. In this way we expected a priori a lowering of the compression efficiency and the quality of the reconstructed images, compared to JPEG. Also we took into account a possible misbalance and an increase of computational complexity at the encoder side. However, as a compensation, we are able to decompress an image much more quickly and to reach some of the characteristic image features directly in the compressed domain. Although we can say that in some way we found an acceptable trade-off between four optimization criteria, we are also aware of the fact that the developed CODEC is far from optimal. Nevertheless, we hope with our CODEC to provide a solid base for further research in this area.

Chapter 6

Directions for Future Research

6.1 Where are we now?

The amount of information that we are exposed to due to advances in the digital transmission and storage technology is steadily rising. In principle, this is not a problem at all, as there are many benefits that we can draw from the "information avalanche". The problem emerges only when this avalanche reaches the size where it is difficult to handle it. Then, having available systems for information retrieval is of the utmost importance. The algorithms for visual-content analysis presented in this thesis were developed with the objective of making a further contribution to a successful practical realization of automated information retrieval systems. Conclusions about the effectiveness of proposed approaches can be drawn from previous chapters of this thesis. The questions that we like to address in this chapter are: technologically, how far are we at this point, what is the rate of technological growth regarding the automation of information management processes, and what are the major directions for pursuing further research in the area of automated information management and retrieval systems.

On the one hand, we can claim that we have come far enough in terms of the obtained technological know-how and regarding the time elapsed since research in the area of information retrieval was initiated. What once started as a modest effort at some industrial and academic research labs (IBM, MIT, ISS*) has grown into a research area involving an enormous

---

* Institute for Systems Science, National University Singapore

number of people and showing its first concrete products (QBIC [Nib93], VIRAGE [Jai97], INFORMEDIA [Chr94], Alta Vista Photo and Media Finder [ALT]).

Compared to the situation of only several years ago, where the biggest issue regarding video-content analysis was to provide reliable ways for automated shot-boundary detection, we currently have available a wide selection of algorithms for high-level video analysis, some of them presented in Chapter 4 of this thesis. Similar advances are obvious also in the image analysis area: not many years have passed between realizing the need for efficient image classification and retrieval systems and the moment when advanced algorithms for these purposes based on high-level criteria [Vai98] became available.

But on the other hand, we must realize that we are still at the very beginning of developing robust solutions for information retrieval. It can be even said that, in spite of all the achievements in the last several years, the same conclusion can be drawn at this moment as in [Pic95]: "the theory and tools that facilitate browsing, querying, retrieval and manipulation of imagery are still in their infancy". This is understandable if we are really aware of the difficulty of transfering cognitive aspects of information analysis and processing to the system level. Thus, the challenge for the research community regarding the development of information management and retrieval systems remains high and we anticipate that this will be the case for many years to come. There are countless issues which need to be addressed in order to meet this challenge successfully. We feel, however, that most of these issues can be grouped into three major research directions that we formulate and discuss in the next section.

6.2 Where to go?

As already discussed in Chapter 1, the user-specified request for information search or its organization is based on his perception of the information content. Such a request needs to be processed in the feature domain, which implies that finding suitable features and related algorithms for providing similar retrieval results as in the case of manual search is crucial for successfully transferring information retrieval to the system level. The methodologies presented in this thesis show some possibilities for reaching this objective. However, a further intensive research is required in order to increase the reliability of existing feature-based image/video analysis methods and to provide solutions for new applications. We need to investigate the suitability of different features, to model various cognitive aspects of video/image content in the feature domain, to search for perceptual feature-comparison metrics, etc. Here we also need to take into account the fact that no universal solutions are possible. Moreover, finding optimal ways of using features for operations on an information database needs to be pursued depending on the target application. Not only is this the only feasible option, but in this way we also have a chance to really optimize the system performance regarding specific applications.

Further, we have to be aware of the fact that a perfect realization of the cognition-based information analysis and processing in the feature domain is not realistic. Therefore, the development of information retrieval systems which are capable of learning through user interaction becomes very important. Having such systems can compensate for an imperfection of features and metrics used, and help improving the reliability and the response time for future operations on a database. For developing such systems, we can use the know-how from the areas of machine learning and artificial intelligence.

Finally, we shown in Chapter 5 the importance of developing information retrieval systems which are able to deal with compressed information directly without reducing their efficiency due to compression. For this reason, further research should be pursued for finding efficient methodologies for extracting relevant features from image/video/audio information compressed using current compression standards. Although a considerable number of approaches related to this has been reported in recent literature, such feature extraction is generally difficult to perform because the optimization of the available compression standards did not consider providing a fast content accessibility. Due to the last statement, we strongly recommend pursuing intensive research toward new compression methods, where the content-access criterion plays an important role. There, either the content should be easily accessible directly in the compressed domain, or the computational complexity of the decompression procedure should be minimized. However, each of these objectives should be combined with the classical criteria, that is, the minimization of the bit rate, of the distortion and of the overall computational complexity. New compression solutions based on the optimum synergy among the four criteria, as discussed in Chapter 5, are required for compressing the information when building the digital libraries of the next generation.

References

[Aha96] Ahanger G., Little T.D.C.: ***A Survey of Technologies for Parsing and Indexing Digital Video,*** Jornal of Visual Communication and Image Representation, Vol. 7, No.1, pp.28-43, March 1996.

[Aku92] Akutsu A. et al.: ***Video indexing using motion vectors,*** Proceedings of VCIP'92, Boston 1992

[ALT] Alta Vista Photo and Media Finder
http://image.altavista.com/cgi-bin/avncgi/

[Ari96] Ariki Y., Saito Y.: ***Extraction of TV News Articles based on Scene Cut Detection using DCT Clustering,*** Proceedings of ICIP '96, Vol. 3, pp. 847-850, Lausanne CH, 1996

[Arm93a] Arman F., Hsu A., Chiu M.: ***Feature Management for Large Video Databases,*** Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases, February 1993.

[Avr98] Avrithis Y.S., Doulamis N.D., Doulamis A.D., Kollias S.D.: ***Efficient Content Representation in MPEG Video Databases,*** Proceedings of IEEE Workshop on Content-Based Access of Image and Video Databases, Santa Barbara CA, 1998

[Bac96] Bach J.R. et al.: ***The Virage Image Search Engine: An open framework for image management,*** Proceedings of IS&T/SPIE Storage and Retrieval for Still Image and Video Databases IV, Vol. 2670, 1996

[Bor93] Bordwell D., Thompson K.: ***Film Art: An Introduction***, McGraw-Hill, New York 1993

[Bor96] Boreczky J.S., Rowe L.: ***Comparison of video shot boundary detection techniques***, Proceedings of IS&T/SPIE Storage and Retrieval for Still Image and Video Databases IV, Vol. 2670, February 1996.

[Boy99] Boykin S., Merlino A.: ***Improving Broadcast News Segmentation Processing***, Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Florence 1999

[Cha95a] Chang S.-F.: ***Compressed-Domain Techniques for Image/Video Indexing and Manipulation***, Proceedings of ICIP'95, Washington DC, October 1995

[Cha95b] Chang S.-F.: ***New Algorithms for Processing Images in the Transform-Compressed Domain***, Proceedings of IS&T/SPIE, Vol. 2501, February 1995

[Che97] Chen L., Faudemay P.: ***Multi-Criteria Video Segmentation for TV News***, IEEE First Workshop on Multimedia Signal Processing, Princeton NJ, 1997

[Chr94] Christel M., Stevens S., Wactlar H.: ***Informedia Digital Video Library***, Proceedings of ACM 2nd International Conference on Multimedia, Video Program, New York, October 1994

[Col76] Coll D.C., Choma G.K.: ***Image Activity Characteristics in Broadcast Television***, IEEE Transactions on Communications, pp. 1201-1206, October 1976

[Cur65] Curran T.F., Ross M.: ***Optimum Detection Thresholds in Optical Communications***, Proceedings of IEEE, pp. 1770-1771, November 1965

[Dav79] Davies D.L., Bouldin D.W.: ***A Cluster Separation Measure***, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No.2, pp. 224-227, April 1979

[DeM98] DeMenthon D., Kobla V., Doermann D.: ***Video Summarization by Curve Simplification***, Proceedings of CVPR '98, Santa Barbara CA, 1998

[ETS94] ETS 300 421: ***Digital broadcasting systems for television, sound and data services; framing structure, channel coding and modulation for 11/12 GHz staellite services***, EBU/ETSI JTC, December 1994

[ETS97] ETS 300 401 ed.2: ***Radio broadcasting systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers***, 1997

[Fis95] Fischer S., Lienhart R., Effelsberg W.: ***Automatic Recognition of Film Genres***, Proceedings of ACM Multimedia '95, San Francisco 1995

[Fri67] Friedman H.P., Rubin J.: ***On some Invariant Criteria for Grouping Data***, Journal Amer. Stat. Assoc., Vol.62, pp.1159-1178, 1967

[Fur95] Furht B., Smoliar S.W., Zhang H.: ***Video and Image Processing in Multimedia Systems***, Kluwer Academic Publishers 1995

[Ger92] Gersho A., Gray R.M.:***Vector Quantization and Signal Compression***, Kluwer Academic Publishers, Boston 1992

[Gre97] Gresle P., Huang T.S.: ***Video Sequence Segmentation and Key Frames Selection Using Temporal Averaging and Relative Activity Measure***, Proceedings of VISUAL '97, San Diego 1997

[Gon95] Gong Y. et al.: ***Automatic Parsing of TV Soccer Programs***, Proceedings of the International Conference on Multimedia Computing and Systems, May 1995

[Gun98] Gunsel B., Tekalp. A.M.: ***Content-Based Video Abstraction***, Proceedings of ICIP '98, Chicago USA, 1998

[Ham94] Hampapur A., Jain R., Weymouth T.: ***Digital Video Segmentation***, Proceedings of ACM Multimedia'94, 1994

[Han97a] Hanjalic A., Ceccarelli M., Lagendijk R.L., Biemond J.: ***Automation of systems enabling search on stored video data***, Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases V, Vol. 3022, February 1997.

[Han97b] Hanjalic A., Lagendijk R.L., Biemond J.: ***A novel video parsing method with improved thresholding***, Proceedings of the Third Annual Conference of the Advanced School for Computing and Imaging, Heijen, The Netherlands, 1997

[Han97c] Hanjalic A., Lagendijk R.L., Biemond J.: ***A New Method for Key Frame based Video Content Representation***, in *Image Databases and Multi Media Search*, Eds. A.W.M. Smeulders, R. Jain, World Scientific, Singapore, 1997

[Han99a] Hanjalic A., Lagendijk R.L., Biemond J.: ***Semi-automatic News Analysis, Indexing and Classification System based on Topics Preselection***, Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases VII, Vol. 3656, January 1999

[Han99b] Hanjalic A., Lagendijk R.L., Biemond J.: ***Automated High-Level Movie Segmentation for Advanced Video Retrieval Systems***, IEEE Transactions on Circuits and Systems for Video Technology, June 1999

[Han99c] Hanjalic A., Lagendijk R.L., Biemond J.: ***An Efficient Image CODEC enabling Content-based Operations in Compressed Domain***, Proceedings of ICIP '99, Kobe 1999

[Han99d] Hanjalic A., Zhang H.: ***Optimal Shot Boundary Detection based on Robust Statistical Models***, Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Florence 1999

[Han00] Hanjalic A., Zhang H.: ***An Integrated Scheme for Automated Video Abstraction based on Unsupervised Cluster-Validity Analysis***, IEEE Transactions on Circuits and Systems for Video Technology, 2000

[Hua99a] Huang S.: ***Digital television: a new way to deliver information***,Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases VII, Vol. 3656, January 1999

[Hua99b] Huang Q., Liu Z., Rosenberg A.: ***Automated Semantic Structure Reconstruction and Representation Generation for Broadcast News***, Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases VII, Vol. 3656, January 1999

[ISO95] ISO/IEC 13818: ***Information Technology – Generic coding of moving pictures and associated audio information***, ISO/IEC JTC 1/SC 29, April 1995

[ISO97] ISO/IEC JTC1/SC29/WG11, ***MPEG-7: Context and Objectives (v.4)***, July 1997.

[Jac92] Jacquin A.E.: ***Image Coding based on a Fractal Theory of Iterated Contractive Image Transformations***, IEEE Transactions on Image Processing, Vol.2, No.1, pp.18-30, January 1992

[Jai82] Jain A.K., Chandrasekaran B.: ***Dimensionality and sample size considerations in pattern recognition practice***, in Handbook of Statistics, Vol. 2 (P. Krishnaiah and L.N. Kanal eds.), North-Holland Publishing Company, Amsterdam 1982

[Jai88] Jain A.K., Dubes R.C.: ***Algorithms for Clustering Data***, Prentice Hall Advance Reference Series, 1988

[Ken98] Kender J.R., Yeo B.: ***Video Scene Segmentation Via Continuous Video Coherence***, Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Santa Barbara, June 1998

[Kik92] Kikukawa T., Kawafuchi S.: ***Development of an automatic summary editing system for the audio visual resources***, Transactions of the Institute of Electronics, Information and Communication Engineers, Vol J75-A, No.2, 1992

[Lie99] Lienhart R.: ***Comparison of automatic shot boundary detection algorithms***, Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases VII, Vol. 3656, January 1999

[Liu98] Liu Z., Huang Q.: ***Classification of Audio Events in Broadcast News***, Proceedings of IEEE Workshop in Multimedia Signal Processing, December 1998

[Mai93] Mai K., Miller J., Zabih R.: ***A robust method for detecting cuts and dissolves in video sequences***, Proceedings of ACM Multimedia '95, San Francisco 1995

[Max60] Max J.: ***Quantization for Minimum Distortion***, IRE Transactions on Information Theory, pp. 7-12, 1960

[McG99] McGee T., Dimitrova N.: ***Parsing TV programs for identification and removal of nonstory segments***, Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases VII, Vol. 3656, January 1999

[Men95] Meng J., Juan Y., Chang S.: ***Scene Change Detection in a MPEG CompressedVideo Sequence***, Proceedings of IS&T/SPIE, Vol. 2419, February 1995

[Nag92] Nagasaka A., Tanaka Y.: ***Automatic video indexing and full-video search for object appearances***, in *Visual Database Systems II*, Eds. Knuth E. and Wegner L.M., volume A-7 of IFIP Transactions A: Computer Science and Technology, pages 113-127, North-Holland, Amsterdam 1992

[Nib93] Niblack W. et al.: ***The QBIC Project: Querying images by content using color, texture and shape***, Proceedings of IS&T/SPIE, Storage and Retrieval for Image and Video Databases, San Jose, February 1993

[Oka93] Okamoto H. et al.: *A Consumer Digital VCR for Advanced Television,* IEEE Transactions on Consumer Electronics, Vol. 39, No.3, pp. 199-204, August 1993.

[Ots93] Otsuji K., Tonomura Y.: ***Projection Detecting Filter for Video Cut Detection,*** Proceedings of ACM Multimedia '93, 1993

[Ots91] Otsuji K., Tonomura Y., Ohba Y.: ***Video browsing using brightness data,*** Proceedings of SPIE/IS&T VCIP'91, Vol.1606, 1991

[Paa97] van Paasen R.: ***Subjective Representation of Video: An Exploratory Study,*** MSc. Thesis, Eindhoven University of Technology (Nl), 1997

[Pap84] Papoulis A.: ***Probability, Random Variables and Stochastic Processes,*** McGraw-Hill, International Editions, 1984

[Pen93] Pennebaker W.B., Mitchell J.L.: ***The JPEG Still Image Data Compression Standard,*** Van Nostrand Reinhold, New York 1993

[Pen94a] Pentland A., Picard R., Davenport G., Haase K.: ***Video and Image Semantics: Advances Tools for Tellecommunications,*** IEEE MultiMedia, Summer 1994

[Pfe96] Pfeiffer S., Lienhart R., Fischer S., Effelsberg W.: ***Abstracting Digital Movies Automatically,*** Journal of Visual Communication and Image Representation, Vol. 7, No. 4, pp. 345-353, December 1996.

[Pic95a] Picard R.W.: ***Light-years from Lena: Video and Image Libraries of the Future,*** Proceedings of IEEE ICIP '95, Vol. I

[Pic95b] Picard R.W.: ***Content Access for Image/Video Coding: The Fourth Criterion,*** Tech. Rep. 295, MIT Media Lab, Perceptual Computing, Cambridge MA, 1994 (also available as MPEG Doc. 127, Lausanne 1995)

[Sah88] Sahoo P.K., Soltani S., Wong A.K.C., Chen Y.C.: ***A survey of thresholding techniques,*** CVGIP, 41:233-260, 1988

[Sal73] Salt B.: ***Statistical style analysis of motion pictures,*** Film Quarterly, Vol 28, pp. 13-22, 1973

[Sau97] Saur D.D. et al.: ***Automated Analysis and Annotation of Basketball Video,*** Proceedings of IS&T/SPIE Vol. 3022, February 1997

[Sha95a] Shahraray B.: ***Scene change detection and content-based sampling of video sequences,*** Proceedings of IS&T/SPIE Vol. 2419, February 1995

[Sha95b] Shahraray B., Gibbon: ***Automatic generation of pictorial transcripts of video programs,*** Proceedings of IS&T/SPIE Digital Video Compression: Algorithms and Technologies, San Jose 1995

[SMA] SMASH project home page: http://www-it.et.tudelft.nl/pda/smash

[Sme97] Smeulders A.W.M., Jain R. (Eds.): ***Image Databases and Multimedia Search,*** Series on Software Engineering and Knowledge Engineering, Vol.8, World Scientific Singapore, 1997

[Smi94] Smith J.R., Chang S.-F.: ***Quad-Tree Segmentation for Texture-Based Image Query,*** Proceedings of ACM 2nd Multimedia Conference, San Francisco, October 1994

[Smi93] Smith B.C., Rowe L.A.: ***Algorithms for Manipulating Compressed Images,*** IEEE Computer Graphics & Applications, Septermber 1993

[Son98] Song S., Kwon T., Kim W.: ***Detection of gradual scene changes for parsing of video data,*** Proceedings of IS&T/SPIE Vol. 3312, 1998

[Sun97] Sun X., Kankanhalli M.S., Zhu Y., Wu J.: ***Content-Based Representative Frame Extraction for Digital Video,*** Proceedings of IEEE Multimedia Computing and Systems, Austin TX 1997

[TNO97] ***TNO Magazine,*** Vol.1, No.4, December 1997

[Ued91] Ueda H., Miyatake T., Yoshizawa S.: **IMPACT:** ***An interactive natural-motion picture dedicated multimedia authoring system,*** Proceedings of the CHI'91, 1991

[Vai98] Vailaya A., Jain A., Zhang H.: ***On Image Classification: City vs. Landscape,*** Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries, Santa Barbara, June 1998

[Vai99] Vailaya A., Jain A., Zhang H.: ***A Bayesian Framework for Semantic Classification of Outdoor Vacation Images,*** Proceedings of IS&T/SPIE Storage and Retrieval for Image and Video Databases VII, Vol. 3656, January 1999

[Vas98] Vasconcelos N., Lippman A.: ***A Bayesian Video Modeling Framework for Shot Segmentation and Content Characterization,*** Proceedings of CVPR '98, Santa Barbara CA, 1998 [dWi92] de With P.H.N.: ***Data Compression Systems for Home-Use Digital Video Recording,*** IEEE Journal on Selected Areas in Communication, Vol 10, No. 1, pp. 97-121, Januar 1992.

[dWi93] de With P.H.N., Rijckaert A.M.A., Keesen H.-W., Kaaden J., Opelt C.: ***An Experimental Digital Consumer HDTV Recorder using MC-DCT Video Compression,*** IEEE Transactions on Consumer Electronics, Vol. 39, No.4, pp. 711-722, November 1993.

[Wol96] Wolf W.: ***Key frame selection by motion analysis,*** in Proceedings of IEEE ICASSP'96, 1996

[Yan93] Yanagihara N., Siu C., Kanota K., Kubota Y.: ***A Video Coding Scheme with a High Compression Ratio for Consumer Digital VCRs,*** IEEE Transactions on Consumer Electronics, Vol. 39, No. 3, pp 192-198, August 1993.

[Yeo95a] Yeo B.-L., Liu B.: ***Rapid Scene Analysis on Compressed Video,*** IEEE Transactions on Circuits and Systems for Video Technology, Vol.5, No.6, December 1995

[Yeu95a] Yeung M.M., Liu B.: ***Efficient Matching and Clustering of Video Shots,*** Proceedings of IEEE ICIP '95, Vol. I

[Yeu97] Yeung M., Yeo B.-L.: ***Video Visualisation for Compact Presentation and Fast Browsing of Pictorial Content,*** IEEE Transactions of Circuits and Systems for Video Technology, Special Issue on Multimedia Technology, Systems and Applications, October 1997.

[Zha93] Zhang H., Kankanhalli A., Smoliar S.W., ***Automatic partitioning of full-motion video,*** Multimedia Systems, Vol.1, pp. 10-28, 1993

[Zha95a] Zhang H., Low C.Y., Smoliar S.W.: ***Video Parsing and Browsing using Compressed Data,*** *Multimedia Tools and Applications*, vol. 1, pp. 89-111, Kluwer Academic Publishers, 1995.

[Zha97a] Zhang H., Wang J.Y.A., Altunbasak Y.: ***Content-Based Retrieval and Compression: A Unified Solution,*** Proceedings of ICIP '97, Santa Barbara CA, 1997

[Zha97b] Zhang H., Wu J., Zhong D., Smoliar S.W.: ***An Integrated System for Content-Based Video Retrieval and Browsing,*** Pattern Recognition, Vol. 30, No.4, pp.643-658, 1997

[Zhu98] Zhuang Y., Rui Y., Huang T.S., Mehrotra S.: ***Key Frame Extraction Using Unsupervised Clustering,*** Proceedings of ICIP '98, Chicago 1998

Summary

In recent years, technology has reached a level where vast amounts of digital information are available at a low price. During the same time, the performance-versus-price ratio of digital storage media has steadily increased. Because it is easy and relatively inexpensive to obtain and store digital information, while the possibilities to manipulate such information are almost unlimited, the *digital libraries* in the professional and consumer environment have grown rapidly. Examples are digital museum archives, Internet archives, image/video archives available to commercial service providers and private collections of digital information at home. All of these are characterized by a quickly increasing capacity and content variety.

With steadily increasing information volumes stored in digital libraries of various types, finding efficient ways to quickly retrieve information of interest becomes crucial. Since searching manually through GBytes of unorganized stored data is tedious and time-consuming, the need grows for transferring information retrieval tasks to automated systems. Realizing this transfer in practice is not trivial, and this specially for video and images. The main problem is that typical retrieval tasks, such as "find me an image with a bird!", are formulated on a *cognitive level*, according to the human capability of understanding the information content and analyzing it in terms of objects, persons, sceneries, meaning of speech fragments or the context of a story in general. Opposed to this, the only feasible way to analyze an image or a video at the algorithmic or *system level* can be in terms of *features*, such as color, texture, shape, frequency components, audio and speech characteristics, and using the algorithms operating on these features. Such algorithms are, for instance, image segmentation, detection of moving objects, shape matching, recognition of color compositions, determination of relations among different objects or analysis of the frequency spectrum of the audio or speech stream. These algorithms can be developed using the state-of-the-art in image and audio analysis and processing, computer vision, statistical signal processing, artificial intelligence, pattern recognition and other related areas. Experience has shown, however, that the parallelism between the cognition-based and feature-based information retrieval is not viable in all cases. Therefore, the development of feature-based content-analysis algorithms has not been directed to enable queries on the highest semantic level, such as the above example with a bird, but mainly towards extracting certain semantic aspects of the information that would allow for a reduction of the overall large search space. The material presented in this thesis is meant to contribute further to research efforts in this direction and concentrates on the specific problem of video and image retrieval.

The first contribution of this thesis is a series of novel algorithms for video analysis and abstraction. These algorithms are developed to provide an overview of the video-library content and logical entry points into a video when browsing through a video library. Also a video index may be constructed based on visual features contained in the abstract, which can then be used for video queries using image retrieval techniques. On the one hand, algorithmic solutions are provided for segmenting a video into temporally homogeneous fragments called *video shots*, for condensing each of the shots into a set of characteristic frames called *key frames* and for performing a high-level analysis of a video content. This high-level analysis includes determining semantic relationships among shots in terms of their temporal characteristics and suitable features of their key frames, and identification of certain semantically interesting video shots. Examples are merging the shots of a movie into scenes or episodes or the identification of anchorperson shots in news programs. On the other hand, we develop an algorithm for automatically summarizing an arbitrary video by extracting a number of suitable key frames in such a way that the result is similar as when that video is summarized manually. One characteristic application where having such abstracts is useful is browsing through a video and searching for a scene of interest. The user only needs to check a limited amount of information contained in an abstract instead of going through the entire video in the fast-forward/rewind mode, while still having available all the characteristic information related to the video content and thus being able to understand and follow that content exclusively on the basis of the abstract.

The second contribution of this thesis is related to the fact that the prevailing amount of information stored in digital libraries will be available in *compressed* form. This is understandable in view of the existence and further improvements of many powerful compression algorithms, through which the space on digital storage media and the capacity of the transmission channels can be used more efficiently. Consequently, features required for information analysis at system level need to be easily accessible in the compressed domain. If this criterion is not taken into account when compressing images before they are included in a digital library, a difficulty in reaching (computing) image features combined with a large number of images in a large-scale library can considerably negatively influence the efficiency of the interaction with that library. Since existing image-compression standards like JPEG are optimized regarding the compression efficiency, image distortion and computational costs, but not regarding the accessibility of all image features which may be required for image-database operations, there is a need for alternative image compression methodologies by which all four optimization criteria are considered. We address this issue in this thesis by developing a novel image CODEC where an acceptable compromise among these four criteria is reached.

Samenvatting

In de afgelopen jaren heeft de technologie een ontwikkelingsniveau bereikt waarbij grote hoeveelheden digitale informatie beschikbaar zijn tegen een lage prijs. Tegelijkertijd is de prijs-prestatieverhouding van moderne opslagmedia almaar verbeterd. Het gemak en de relatief lage kosten waarmee digitale informatie kan worden verkregen en opgeslagen, naast de bijna onbeperkte mogelijkheden om deze informatie te manipuleren, zijn de belangrijkste oorzaken van de snelle groei van zogenaamde *digitale bibliotheken* in zowel de professionele sfeer als in de consumentenomgeving. Voorbeelden hiervan zijn digitale museumarchieven, Internetarchieven, beeld- en videoarchieven bij commerciële aanbieders van diensten en privé-collecties van digitale informatie thuis in de huiskamer.

Met de toenemende hoeveelheden informatie in digitale bibliotheken wordt het snel en efficiënt zoeken naar de specifieke informatie steeds belangrijker. Omdat het moeilijk en tijdrovend is om GBytes aan data handmatig te doorzoeken, neemt de noodzaak toe om zoektaken te laten verrichten door automatische systemen. De praktische realisatie hiervan is echter verre van eenvoudig, en dit in het bijzonder voor het geval van video en beelden. Het grootste probleem hierbij is dat typische zoektaken zoals "zoek een foto met een vogel!" zijn geformuleerd op het *cognitieve niveau*, op basis van het vermogen van de mens om de informatieinhoud te begrijpen en te analyseren m.b.t. objecten, personen, landschappen, de betekenis van spraakfragmenten en de context van een verhaal in het algemeen. Hier staat tegenover dat de enige haalbare manier om beeld- of videoinformatie te analyseren op algoritmisch of *systeemniveau* is door gebruik te maken van *kenmerken*, zoals kleur, textuur, vorm, frequentie- en tijdsignaaleigenschappen van audio en spraak, en door algoritmen te gebruiken die operaties uitvoeren op deze kenmerken. Hierbij kunnen verschillende algoritmen worden onderscheiden, zoals t.b.v. beeldsegmentatie, detectie van bewegende objecten, herkenning van kleurcomposities, bepaling van relaties tussen verschillende objecten en analyse van frequentie- en tijdsignaal-eigenschappen van audio en spraak. Deze algoritmen kunnen worden ontwikkeld onder gebruik making van de "state-of-the-art" op het gebied van de beeldverwerking, audio- en spraakherkenning, computer vision, statistische signaalverwerking, kunstmatige intelligentie, patroon-herkenning, etc. De ervaring heeft ons echter geleerd dat het laten overeenkomen van cognitie-gebaseerde en kenmerk-gebaseerde zoekprocessen nog verre van praktisch haalbaar is. Om deze reden is de ontwikkeling van kenmerk-gebaseerde analysetechnieken feitelijk niet gericht op het realiseren van zoekprocessen op het hoogste semantische niveau, zoals bovengenoemde voorbeeld met een vogel, maar meer op de extractie van bepaalde semantische aspecten van de informatie, die vervolgens gebruikt kunnen worden om de (grote) zoekruimte te beperken. Het materiaal zoals gepresenteerd in dit proefschrift is bedoeld om verder bij te dragen tot het onderzoek in deze richting en concentreert zich op het specifieke probleem van het zoeken naar video en beelden.

De eerste bijdrage van dit proefschrift is een serie nieuwe algoritmen voor de analyse en samenvatting van video-data. Deze algoritmen zijn ontwikkeld om een overzicht te geven van de inhoud van een videobibliotheek en logische instappunten te verschaffen in een videoprogramma gedurende het doorzoeken van zo'n bibliotheek. Ook kan een video-index geconstrueerd worden op basis van visuele kenmerken binnen de samenvatting van de video. Deze index kan vervolgens gebruikt worden om bepaalde segmenten van een video te zoeken onder gebruik making van beeld-zoekmechanismen. Zo worden in dit proefschrift aan de ene kant algoritmische oplossingen gegeven voor de segmentatie van een video in temporeel homogene fragmenten, ook wel *video shots* genoemd, voor de representatie van elk shot door een aantal karakteristieke frames, of *key frames*, en voor een "high-level" analyse van de video-inhoud. Deze high-level analyse omvat het vinden van semantische relaties tussen shots in termen van temporele karakteristieken en specifieke kenmerken van key frames, en de identificatie van video shots met een hoge semantische waarde. Voorbeelden zijn het groeperen van film shots in episodes of scènes of de identificatie van nieuwslezer shots in een nieuwsprogramma. Aan de andere kant wordt er een algoritme ontwikkeld voor de automatische samenvatting van een willekeurige video-sequentie door een aantal geschikte key frames te extraheren op zodanige wijze dat het resultaat gelijk is aan dat van een handmatige samenvatting. Eén karakteristieke toepassing waarbij zo'n collectie key frames van nut is, is het doorzoeken van de video en het zoeken naar een specifieke episode of scène. De gebruiker hoeft alleen de sterk gereduceerde hoeveelheid informatie in de samenvatting te doorzoeken in plaats van de hele sequentie in de "fast-forward/rewind" mode, terwijl hij/zij nog steeds beschikt over alle karakteristieke informatie m.b.t. de inhoud van de sequentie, en zodoende in staat is de inhoud te volgen louter op basis van de samenvatting.

De tweede bijdrage van dit proefschrift is gerelateerd aan het feit dat verreweg de grootste hoeveelheid aan opgeslagen informatie in digitale bibliotheken beschikbaar zal zijn in *gecomprimeerde* vorm. Dit is gemakkelijk te begrijpen in het licht van de beschikbaarheid van vele krachtige compressiealgoritmen met nog mogelijke verbeteringen, waardoor de opslagruimte in digitale bibliotheken en de capaciteit van transmissiekanalen nog efficiënter benut kunnen worden. Als gevolg hiervan dienen kenmerken die noodzakelijk zijn voor de informatie-analyse op systeemniveau gemakkelijk toegankelijk te zijn in het gecomprimeerde domein. Als dit criterium niet in acht wordt genomen tijdens het comprimeren van beelden en voordat zij deel uitmaken van een digitale bibliotheek, dan wordt het moeilijk om bepaalde beeldkenmerken te vinden (te berekenen), zeker als we te maken hebben met grote aantallen beelden, met als gevolg een sterk negatieve beïnvloeding van de efficiëntie van de interactie met de digitale bibliotheek. Omdat bestaande standaarden zoals JPEG geoptimaliseerd zijn m.b.t. compressie-efficiëntie, beeldvervorming en rekenkomplexiteit maar niet m.b.t. toegangelijkheid van alle kenmerken zoals nodig voor beeld-database operaties, zijn er alternatieve beeldcompressiemethoden nodig die alle genoemde optimalisatiecriteria in beschouwing nemen. Hier wordt in het proefschrift op ingegaan door een nieuw CODEC voor beelden te ontwikkelen waarbij een acceptabel compromis is bereikt m.b.t. de vier genoemde criteria.

Acknowledgments

So, after a lot of "heavy science" from previous chapters, we come to the point where I with a great pleasure wish to express my gratitude to many individuals who in the one way or another supported me during the last four years and helped me produce this thesis.

First of all, I was extremely happy to have had Inald Lagendijk and Jan Biemond as guides through the complex labyrinth of science. I thank them both for giving me a chance to do my Ph.D. at the ICT Group and for their valuable support over the years. However, I owe an extremely pleasant stay at the ICT Group not only to my direct advisors; I cannot avoid expressing my enormous appreciation to Annett for being always there for all of us and for keeping her smile and relaxness even in the busiest times. Ben and Hans, thanks for keeping our engines running, and Peter, Gerhard, Erik, André, Isabel and the rest of our lunch group for an always funny and "inspiring" atmosphere in the cantina.

From the people outside the ICT Group, I would first like to thank Mirjam Nieman for courageously fighting with my grammatical errors throughout the thesis. I also feel obliged to mention an exceptionally good cooperation among all partners within the EU-ACTS SMASH project in the time period 1995-1998. Special thanks to Marco Ceccarelli for many productive discussions and for his contribution to our several publications.

Then, of course, a lot of things would have been different if I had not visited Hewlett-Packard Labs in Palo Alto in Summer 1998. Therefore, special thanks to HongJiang for making my visit to HP possible, to Wei for his great help regarding various "every-day" issues, and to Cathy, Aditya and Xavier for great times in our SEED corner.

Last but not least, special gratitude to my family for too many things to write down...

Curriculum Vitae & Bibliography

Alan Hanjalic was born on 23 April 1971 in Sarajevo, Bosnia and Herzegovina. He completed his high school education in 1989 at Arlington High School, LaGrangeville NY, USA, and at the High Music School in Sarajevo where he majored piano and obtained an artistic diploma in 1990.

In 1991 he completed two years of studies at the Electrical Engineering Department of the University Sarajevo, when he moved to the Friedrich-Alexander University Erlangen-Nuremberg, Germany, to continue his studies. There he received the Diplom-Ingenieur (Dipl.-Ing.) degree in Electrical Engineering in 1995. From 1995 to 1999 he worked towards his Ph.D. degree at the Information and Communication Theory (ICT) Group of the Delft University of Technology, The Netherlands, in the area of visual-content analysis for advanced multimedia retrieval systems.

In parallel with doing his Ph.D. he worked from 1995 to 1998 as a researcher and software developer within the European ACTS SMASH (Storage for Multimedia Applications Systems in the Home) project. From May to September 1998 he was with Hewlett-Packard Laboratories, Palo Alto (CA, USA), where his activities were concentrated on developing efficient video segmentation and abstraction techniques.

In 1999 he joined the scientific staff of the ICT Group as Assistant Professor.

Bibliography

Books

1. A. Hanjalic, G.C. Langelaar, P.M.B. van Roosmalen, J. Biemond, R.L. Lagendijk: ***Image and Video Databases: Restoration, Watermarking and Retrieval,*** Volume 8 of the series *Advances in Image Communications,* to be published by Elsevier Science, 2000

Journal publications

1. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***Automated High-Level Movie Segmentation for Advanced Video Retrieval Systems,*** IEEE Transactions on Circuits and Systems for Video Technology, June 1999

2. A. Hanjalic, H. Zhang: ***An Integrated Scheme for Automated Video Abstraction based on Unsupervised Cluster-Validity Analysis,*** IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Object-based Video Coding and Description

Patents

1. Patent application in the area of automated video content analysis and abstraction (in cooperation with Hewlett-Packard Labs, Palo Alto)

Various presentations

1. R.L. Lagendijk, A. Hanjalic, G.C. Langelaar, J.C.A. van der Lubbe: ***Methods for Browsing and Copy Protection in Multimedia Storage Systems,*** Key note presentation at ICOMT '96: International Conference on Multimedia Technology and Digital Telecommunication Services, Budapest (H), 1996

2. R.L. Lagendijk, A. Hanjalic: ***(The SMASH project's) Perspective on Browsing of Consumer Video Archives,*** Presentation at MPEG-7 Seminar, Bristol (UK), 1997

3. A. Hanjalic: ***Image Compression for Database Retrieval Applications,*** Presentation at the Fall 1999 Meeting of the Dutch Society for Pattern Recognition and Image Processing, hosted by PHILIPS Medical Systems, Best (Nl), 1999

Technical reports

1. D. Melpignano, C Tani, J. Tasic, A. Hanjalic, G.C. Langelaar, K. Makinwa, J. Ero, M. van der Korst, F. Schalij, R. Tol: ***Report on Future Needs and Technical Possibilities of a SMASH,*** EU-ACTS Project SMASH, Deliverable #3, August 1996

2. A. Hanjalic, G.C. Langelaar, R.L. Lagendijk, M. Ceccarelli, M. Soletic: ***Report on Technical Possibilities and Methods for Security of SMASH and for Fast Visual Search on Compressed/Encrypted Data,*** EU-ACTS Project SMASH, Deliverable #5, November 1996

3. A. Hanjalic, G.C. Langelaar, R.L. Lagendijk, M. Ceccarelli, M. Soletic: ***Specification of Hardware and Software Architecture for Security and Intelligent Search Facilities,*** EU-ACTS Project SMASH, Deliverable #13, February 1998

Papers in conference proceedings

1. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***Achievements and Challenges in Visual Search of Video,*** 17th Symposium on Information Theory in the BENELUX, Enschede (NL), 1996

2. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***A New Method for Key Frame based Video Content Representation,*** In A.W.M. Smeulders and R. Jain (eds.): Image Databases and Multi-Media Search, ISBN 981-02-3327-2, World Scientific Singapore, Singapore 1998 (Proceedings of the First International Workshop on Image Databases and Multi-Media Search, Amsterdam (NL) 1996)

3. E. Steinbach, A. Hanjalic, B. Girod: ***3D Motion and Scene Structure Estimation with Motion Dependent Distortion of Measurement Windows,*** IEEE International Conference on Image Processing (ICIP'96), Lausanne (CH), 1996

4. R.L. Lagendijk, A. Hanjalic, M.P. Ceccarelli, M. Soletic, E.H. Persoon: ***Visual Search in a SMASH System,*** IEEE International Conference on Image Processing (ICIP'96), Lausanne (CH), 1996

5. M. Ceccarelli, A. Hanjalic, R.L. Lagendijk: ***A Sequence Analysis System for Video Databases*** , In V. Cappellini (eds.): Time-Varying Image Processing and Moving Object Recognition 4, ISBN 0 444 82307 7, pages 133-138, Elsevier, Amsterdam 1997 (Proceedings of the 5th International Workshop on Time-Varying Image Processing and Moving Object Recognition, Florence (I), 1996)

6. A. Hanjalic, M. Ceccarelli, R.L. Lagendijk, J. Biemond: ***Automation of Systems Enabling Search on Stored Video Data,*** SPIE/IS&T ELECTRONIC IMAGING '97, Storage and Retrieval for Image and Video Databases V, San Jose (Ca, USA), 1997

7. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***A novel video parsing method with improved thresholding*** , Third Annual Conference of the Advanced School for Computing and Imaging (ASCI '97), Heijen (NL), 1997

8. J. Zaletelj, R. Pecci, F. Spaan, A. Hanjalic, R.L. Lagendijk: ***Rate Distortion Optimal Contour Compression Using Cubic B-Splines,*** IX European Signal Processing Conference (EUSIPCO '98), Rhodos (Gr), 1998

9. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***Template-based Detection of Anchorperson Shots in News Programs ,*** IEEE International Conference on Image Processing (ICIP '98), Chicago (Il, USA), 1998

10. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***Semi-Automatic News Analysis, Classification and Indexing System based on Topics Preselection,*** SPIE/IS&T ELECTRONIC IMAGING '99, Storage and Retrieval for Image and Video Databases VII, San Jose (Ca, USA), 1999

11. A. Hanjalic, H. Zhang: ***Optimal Shot Boundary Detection based on Robust Statistical Models,*** IEEE International Conference on Multimedia Computing and Systems (ICMCS'99), Florence (I), 1999

12. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***Automatically Segmenting Movies into Logical Story Units,*** Third International Conference on Visual Information Systems (VISUAL '99), Amsterdam (Nl), 1999

13. A. Hanjalic, R.L. Lagendijk, J. Biemond: ***Efficient Image CODEC with Reduced Content Access Work,*** IEEE International Conference on Image Processing (ICIP '99), Kobe (Jp), 1999