UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Department of Signal Theory and Communications

Ph.D. Dissertation

On the Synergy between Indexing and Compression Representations for Video Sequences

Author: Javier Ruiz Hidalgo
Advisor: Prof. Philippe Salembier Clairon

Barcelona, September 2006

Abstract

This work discusses the usefulness of exploiting the synergy between compression and indexing representations of video sequences. The study is separated into two main tasks. In the first task, compression representations are investigated to produce more optimize indexing representations. In the second task, indexing representations are exploited to provide more efficient compression representations. In this thesis, the term *compression representation* denotes all data, formally called bitstream, that describes the digital content in a compact manner, using the fewer bits possible, and that allows the functionality of visualizing the content. On the other hand, the term *indexing representation* refers to all data structures, descriptors or extra information that are extracted from the content in order to provide functionalities of indexing, browsing, summarizing, searching or retrieval.

For the last years, both compression and indexing representations have been studied independently of each other and therefore both representations have been obtained or created using different schemes and algorithms. Nevertheless, both representations describe the same content and therefore it is natural to think that each of them can improved by exploiting the other one.

In the first part of this thesis, compression representations of video sequences are analyzed in order to create richer, more robust video shot representations. In particular, mosaics are jointly used by both representations to create more efficient representations. The mosaic coding scheme of the MPEG-4 standard is investigated in order to be able to use mosaic, not only in coding, but to create indexing representation to summarize and browse video shots. The proposed indexing representation is content based as the content of the video scene is analyzed and separated into background and foreground objects. A mosaic is employed to represent the background information and other representations, called key-regions, are used to represent the foreground objects.

In this work, the MPEG-4 algorithm to build the mosaic is improved using connected operators to better represent the background information without penalty on their coding performance in the mosaic coding scheme. Mosaics are also employed in the key-region extraction algorithm hence foreground objects of the scene can be segmented and represented by several key-regions. The proposed indexing representation is composed of the above men-

tioned mosaic and key-regions and constitutes a summarization of the described video shot. Moreover, it may also contain motion information hence users can have a first approximation of how foreground regions move through the scene.

In the second part, the coding efficiency of compression representations, in particular the H.264 standard, is improved using indexing representations. Four techniques are investigated to prove that indexing representations, even if extracted for other functionality, can be exploited to increase the coding efficiency of current video codecs.

In the first proposed scheme, video transitions are encoded using information extracted from a transition descriptor. The second proposed scheme reformulates part of the motion estimation step of current hybrid codecs into a search and retrieval problem. This second scheme improves the long term selection of reference frames by using low level indexing representations such as color descriptors. The third presented scheme uses a high level descriptor to create a new shuffled sequence where video codecs can better exploit its temporal redundancy. Finally, the fourth proposed scheme consists of a frame type selection based on indexing representations. In this scheme, motion information extracted from the indexing representation is introduced in the video codec hence it can select an optimum GoP structure for the video to be encoded.

Resumen

Este trabajo estudia la utilidad de explotar la sinergia entre las representaciones de compresión e indexación de secuencias de vídeo. El estudio se ha separado en dos tareas principales. En la primera tarea, las representaciones de compresión se han analizado para producir representaciones de indexación más optimizadas. En la segunda tarea, las representaciones de indexación se han explotado para generar representaciones de compresión más eficientes. En esta tesis, la expresión *representación de compresión* se refiere a todos aquellos datos, llamados normalmente bitstream, que describen el contenido digital de una manera compacta, usando el menor número de bits posible y permitiendo a su vez la funcionalidad de visualización del contenido. Por otro lado, el término *representación de indexación* se refiere a todas aquellas estructuras de datos, descriptores o información extra que ha sido extraída del contenido digital para proporcionar funcionalidades de indexación, resumen, búsqueda o adquisición.

Durante los últimos años, ambas representaciones has sido estudiadas independientemente y, por ello, las representaciones de compresión e indexación se han obtenido o generado usando diferentes esquemas y algoritmos. Sin embargo, ambas representaciones describen el mismo contenido digital y, por lo tanto, es normal pensar que cada una de ellas se puede beneficiar de la otra.

En la primera tarea de la tesis, las representaciones de compresión son analizadas para crear representaciones de planos de vídeo más robustas y optimizadas. En particular, ambas representaciones investigan los mosaicos para crear representaciones más eficientes. El esquema de codificación por mosaicos del estándar MPEG-4 se amplía para poder usar el mosaico, no sólo en codificación, sino para resumir y describir planos de vídeo. La representación de indexación que se propone en este trabajo se basa en el contenido de la escena ya que ésta es analizada para separar los objetos de primer y segundo plano. El mosaico se utiliza para representar los objetos del segundo plano mientras que otras representaciones, llamadas *regiones clave*, son creadas para representar los objetos del primer plano.

Además, en este trabajo los mosaicos se mejoran mediante operadores conexos para que puedan representar mejor el segundo plano de la escena sin que ello suponga ninguna pérdida en su eficiencia de codificación. Los mosaicos también se utilizan en el algoritmo de extracción

de regiones clave, permitiendo la segmentación y descripción de objetos de primer plano. La representación de indexación que propone en esta tesis supone un resumen compacto del plano de vídeo analizado. Además, la representación de indexación puede enriquecerse añadiendo información de movimiento con lo que se puede tener una primera idea de cómo los objetos del primer plano se mueven en la escena del plano de vídeo.

En la segunda tarea, se estudia la mejora de la eficiencia de las representaciones de compresión, en particular del estándar H.264, mediante el uso de representaciones de indexación. Se investigan cuatro técnicas distintas para probar que las representaciones de indexación, incluso si son extraídas para cubrir otras funcionalidades, pueden ser aprovechadas para incrementar la eficiencia de los codificadores de vídeo actuales.

En la primera técnica propuesta, las distintas transiciones de vídeo se codifican empleando la información extraída de un descriptor de transiciones. La segunda técnica reformula parte de la estimación de movimiento de un codificador híbrido como un clásico problema de búsqueda y adquisición. Esta segunda técnica mejora la selección de tramas de referencia a largo plazo mediante representaciones de indexación de bajo nivel como por ejemplo descriptores de color. La tercera técnica estudiada emplea un descriptor de alto nivel para crear un nuevo orden de codificación en la secuencia de vídeo donde los codificadores pueden aprovechar mejor la redundancia temporal de la misma. Finalmente, la última técnica propuesta consiste en la selección del tipo de trama basándose, de nuevo, en representaciones de indexación. En concreto, varios descriptores de movimiento se utilizan para seleccionar estructuras de GoP óptimas para codificar las secuencias de vídeo.

Agradecimientos

Este documento es el fruto de la investigación y del trabajo realizado durante estos últimos seis años. Es el final de una etapa en mi vida que me ha servido para formarme tanto profesionalmente como personalmente. Por ello, me gustaría aprovechar estas líneas para agradecer el apoyo recibido por todas las personas que me han rodeado durante todo este tiempo.

En primer lugar me gustaría dar las gracias a mi director Philippe. Ha sido todo un placer trabajar con él. Es una persona excelente de la cual tengo el orgullo de decir que, más que un director o un compañero de trabajo, es un amigo.

También dar las gracias a todo el grupo de Procesado de Imagen del departamento, por ofrecerme una mano cuando la necesitaba, por sus útiles comentarios y no pocas ideas. Gracias también a Eliseo, sin él estaría todavía *lanzando y esperando procesos*.

Por último dar las gracias a mi familia, a los amigos y especialmente a Ruth, por su apoyo y, sobretodo, por su *infinita paciencia* durante todos estos años.

A todos, de nuevo, gracias.

Contents

Notation

Through the thesis, matrices and images are written using uppercase boldface, e.g. $\mathbf{I}$. Vectors are written in lowercase boldface, e.g. $\mathbf{m}$, and represent column vectors by default. Scalar variables are written in normal typeface, e.g. α. Other notation has been introduced as follows:

$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	Inverse of matrix $\mathbf{A}$
$\mathbf{I}(x, y)$	The $(x, y)^{th}$ element of matrix $\mathbf{I}$ starting from element $(0, 0)$
$\mathbf{m}(k)$	The k^{th} element of vector $\mathbf{m}$ starting from element $k = 0$
$\mathbf{p} = (x, y, 1)^T$	Homogeneous coordinates of pixel (x, y)
$\mathcal{N}_k$	k^{th} node of a hierarchical structure or tree
$\lvert a \rvert$	Absolute value of a
$\lceil a \rceil$	Smallest integer value not less than a
$\log_a(\cdot)$	Logarithm in base a
$min(A)$	The minimum of all elements of matrix or vector A
$min(a, b)$	The minimum of two elements a and b
$max(A)$	The maximum of all elements of matrix or vector A
$max(a, b)$	The maximum of two elements a and b
$\simeq$	Approximately equal
$\vee$	Maximum operator
$\psi(f)$	Operator applied on function f
$\psi^*(f)$	Dual operator of $\psi(f)$
X, Y	Binary set
$\cup$	Union of binary images
$\cap$	Intersection of binary images
$SAD(\mathbf{A}, \mathbf{B})$	Sum of absolute differences between matrices $\mathbf{A}$ and $\mathbf{B}$
$\epsilon_s\{\cdot\}$	Binary erosion with a structuring element s
$\mathcal{G}(\cdot)$	Magnitude of a spatial gradient operator
$\mathbf{R}(\mathbf{mv})$	Predicted block from reference image $\mathbf{R}$ using motion vector $\mathbf{mv}$

Acronyms

4VD	4-vertices displacement motion estimation algorithm
APCA	Adaptive principal component analysis
AVC	Advanced video codec
B Frame	Compressed frame with previous and future image references
BiM	Binary format for MPEG-7
BPT	Binary partition tree
bps	Bits per second
CABAC	Context-adaptive binary arithmetic coding
CD	Compact disk
CIF resolution	Common intermediate format (352×288)
D	Descriptor
Ds	Descriptors
DC	DCT coefficent for zero frequency
DCT	Discrete cosine transform
DDL	Description definition language
DFT	Discrete fourier transform
DS	Description scheme
DSs	Description schema
DVD	Digital versatile disk
DTV	Digital television
DWT	Discrete wavelet transform
fps	Frames per second
GoF	Group of frames
GoP	Group of pictures
HMMD	Hue max min difference color space
HSV	Hue saturation value color space
I Frame	Compressed frame with no external image references
ISO	International Standard Organization
ITU-T	International Telecommunications Union
KLV	Key length value
kbps	Kilobits per second

MPEG	Moving pictures expert group
MV	Motion vector
P Frame	Compressed frame with previous image references
PCM	Pulse code modulation
PSNR	Peak signal to noise ratio
QCIF	Quarter common intermediate format (176×144)
RD	Rate distortion
RGB	Red green blue color space
SAD	Sum of absolute differences
SMPTE	Society of motion picture and television engineers
SSD	Sum of square differences
TV resolution	Television resolution (720×576)
VS	Video segment
XML	Extended markup language
YCbCr	Luminance chrominance color space

Chapter 1

Introduction

1.1 Motivation

During the last years, digital video representations have been an active research topic for the image processing community. This is motivated by the large amount of digital audiovisual material that is available for consumption. This amount of material is expected to grow, due to the increasing usage of the World Wide Web and the increasing acceptance of digital media to record and store audiovisual information. Historically, one of the basic functionalities of video representation has been compression. *Compression representations* aim at depicting the content in a compact manner using the minimum number of bits possible and allowing its visualization. The functionality of compression allows storing the digital content in an efficient way hence the number of video sequences that can be stored on a given space is maximized. It also permits the distribution of digital content over limited bandwidth networks.

As the amount of digital audiovisual content increases, applications that manipulate this content must not only provide the functionality of compressing it efficiently, but also of browsing, indexing, summarizing, providing functions of search and retrieval, etc. For instance, multimedia applications need to manage very large databases of audiovisual content and, therefore, there is a need to index and catalog it efficiently. Also, content providers need to show their catalog to consumers in an efficient way hence consumers are able to search for the audiovisual content they want, browse it and, finally, retrieve it. In this thesis, the term *indexing representation*, also known as *metadata* in the literature, refers to all data structures, descriptors or extra information that are extracted from the content in order to provide functionalities of indexing, browsing, search and retrieval. These indexing representations describe the audiovisual content but they cannot be used to visualize the content. For instance, an image histogram is an indexing representation of an image that has lost the spatio-temporal characteristics but that is very useful for indexing images or for searching based on, for instance, color similarity.

Classically, both compression and indexing representations have been considered independently of each other and therefore both representations have been defined and obtained using different schemes and algorithms. Even though compression and indexing representations have different goals and functionalities, both representations describe the same content and, hence, it may be possible to enrich each of them with the other. Ideally, the perfect representation would be one that is able to provide all these functionalities efficiently. By efficiently it is understood that the representation is capable of providing all the above mentioned functionalities in the most appropriate way so, for instance, the content is compressed using the fewer possible bits, the content is indexed and therefore search and retrieval queries find it easily and quickly, the content is also summarized, etc. This ideal representation, however, is virtually impossible to achieve. For that reason, and as a first step towards the support of multiple functionalities, it is more realistic to exploit the synergy between these compression and indexing representations by using some information of one representation to improve the other or even sharing the same data in both representations to provide different functionalities.

In this thesis, the synergy between indexing and compression representations is studied. Firstly, compression representations are explored to create more effective, richer indexing representations for summarizing and browsing video shots. Secondly, indexing representations are investigated in order to select those that may have some potential to improve the coding efficiency of standard compression representations. It is important to remark that, in the framework of this thesis, only standardized indexing and compression representations have been evaluated. This means that, for instance in the case of improving compression representations, not any kind of extra information is used, but only information included in standard indexing representations which, even if extracted with other functionality in mind, may help increasing the coding efficiency of compression representations.

The fact that standard indexing representations are already useful for other purposes, rather than coding, and that they might be already available for both encoder and decoder, makes these indexing representations a very nice candidate to exploit. Likewise, the massive usage of compression representations to store and transmit video content incites to take advantage of these representations in order to provide indexing or browsing functionalities.

1.2 General objectives

The main objective of this thesis is to investigate the synergy between compression and indexing representations. In order to achieve this objective, two initial steps must be performed:

- Study of standard representations. Current compression standards such as MPEG-4 and H.264/MPEG-4 AVC [1] and indexing standards such as MPEG-7 and SMPTE are

[1] Referred as H.264 through this thesis

reviewed. Specific parts of the standards that may be employed to exploit the synergy between representations are identified during this study.

- Analyses of the different scenarios where the synergy between both representations can be exploited are considered. In particular, scenarios, where the availability of indexing representations at the encoder or decoder end can be assumed, are examined. Applications included in these scenarios are investigated.

The study of the synergy itself is also performed in two steps:

- Exploitation of compression representations to improve the indexing of video sequences. Mosaics are chosen as a candidate to be shared between compression and indexing representations. Techniques are investigated to first, incorporate mosaics into a compression representations (exploiting the mosaic coding scheme proposed by the MPEG-4 standard) and second, to create indexing representation from mosaics to summarize and browse video shots. The resulting indexing representation separates the background and foreground objects of the video shot. It uses a mosaic to represent the background information and other representations, called key-regions, to represent the foreground.

- Exploitation of indexing representations to improve the video coding performance. In this step, standard indexing representations, selected from MPEG-7 and SMPTE, are studied in order to increase the coding efficiency of current compression representations. Four techniques are investigated to prove that indexing representations, even if extracted for other functionality, can be exploited to increase the coding efficiency of current video codecs.

1.3 Thesis organization

This thesis is organized in three parts. Part I presents the framework in which this work is enclosed. In this part, Chapter 2 reviews several compression and indexing standards that have appeared during the last years. Chapter 3 is devoted to review the state of the art of techniques that exploit the synergy between compression and indexing representations. First, the scenarios where this synergy can be exploited are detailed in Section 3.1. Then, Section 3.2 overviews techniques that have appeared in the literature and make use of compression representation to create indexing ones, and in particular, generate indexing representations to describe video shots for browsing and summarization. At the end of this part, Section 3.3 overviews techniques that have appeared in the literature and exploit standard indexing representations to improve the coding efficiency of current video codecs.

Part II is devoted to study how indexing representations can be improved based on compression ones. Chapter 4 reviews the motivations to study this particular synergy between

representations. In Chapter 5, an indexing representation to describe and summarize video shots is proposed. This indexing representation is based on a mosaic and several key-regions as mentioned previously. Chapter 6 focuses on the creation of the mosaic and proposes the use of morphological operators to improve the ability of the mosaic to represent background information. Chapter 7 details the approach to create key-regions. Key-regions are included in the proposed indexing representation and describe the foreground objects of the video shot. To finalize this part, Chapter 8 is devoted to showing some results of the proposed indexing representation to assess its ability to describe video shots. This last chapter also studies the synergy of the mosaic when used in compression representation.

Part III is devoted to studying how compression representations can be improved based on indexing ones. Chapter 9 presents the motivations of the study of this synergy between representations. In Chapter 10 the MPEG-7 and SMPTE standards are reviewed selecting the indexing representations that may have some potential to improve the coding efficiency of current video codecs. The following chapters in this part (Chapters 11 to 14) are devoted to present in detail four schemes which make use of existing indexing representations to improve the coding efficiency of the H.264 video codec.

Chapter 15 provides some conclusions and proposes future extensions to this work. Appendix A gives an overview of the MPEG-7 description definition language (DDL) employed to describe the indexing representations used in this thesis and, finally, Appendix B is devoted to show several key-frames of all test sequences used in the experimental tests of this thesis.

Part I

General Framework

Chapter 2

Overview of Standard Indexing and Compression Representations

The need of storing video in a reduced amount of bits and of allowing the possibility of distributing digital video content over networks with a limited bandwidth has motivated the study of compression representations. Two different approaches to video compression may be considered: lossless and lossy representations.

The main goal of lossless representations is to represent the content without any quality loss and using the minimum amount of bits. This is usually performed by using predictive, run-length, Huffman or arithmetic coding [59, 25, 50]. On the other hand, lossy representations describe video sequences in a rate-distortion efficient manner. In lossy representations, the quality is reduced (the original sequence cannot be recreated from the representation) but the data bitrate needed to represent the content is smaller than that of the lossless case. Lossy representations are the most common and spread case in video compression representations.

The motivation for indexing representations relies on the exponential growth of the digital multimedia content. The vast amount of digital content has introduced the need of new functionalities in order to be able to search, index, browse or retrieve the content. To fulfill that need different indexing representations have appeared in the last years. In particular, some international standards, such as MPEG-7 [32, 60], SMPTE [78] and MPEG-21 [41] focus on solving these problems.

The following sections overview some standards that provide compression and indexing representations. Section 2.1 gives an overview of standards for compression representations. Section 2.2 reviews indexing representations focusing on both MPEG-7 and SMPTE standards.

2.1 Standards for Lossy Compression Representations

Lossy compression representations have been deeply studied in the literature. The history of lossy digital video compression begins early in the 1970s with the introduction of the Discrete Cosine Transform (DCT) by Ahmed, Natarajan and Rao [3]. The DCT is a close relative of the Discrete Fourier Transform (DFT). During the DCT encoding process, a picture is broken up into smaller blocks (usually of 8×8 or 16×16 pixels) which serve as the input to the DCT transform. DCT coefficients are then quantized using weighting functions optimized for the human visual system and the resulting quantized DCT coefficients are entropy encoded. In order to decompress the image, the process is carried out in reverse. In this scheme, the quantization step is the lossy component of video compression (as video sequences were compressed as separate frames).

The first attempts to exploit the temporal redundancy of video sequences started in the 1980s with the ITU-T recommendation H.120 [34]. The temporal redundancy was used by computing the frame difference between consecutive frames. In 1991, The ITU-T recommendation H.261 [33] employed the idea of motion compensation. The frame difference was preceded by a motion estimation step to align blocks in both frames to the same temporal reference [36]. The resulting motion vectors of the estimation step were used to predict the current image and obtain the difference image (called prediction error) between current and predicted frame. Both the prediction error (encoded using a DCT scheme) and the motion vectors were encoded to form the final compression representation, known as a P frame.

Later in 1991, the standard MPEG-1 [53] offered superior quality than H.261 when operated at higher bitrates. The compression representation was optimized by using an expanded motion vector search range which allowed larger gaps between I (a frame with no external image references) and P frames. The standard also introduced B frames. These B frames could be predicted using bi-directional motion prediction from past and future references. MPEG-1 also introduced half-pel motion estimation where motion vectors have precisions up to half pixel. The MPEG-2 [30] appeared around 1994. It is similar to MPEG-1, but also provides compression representation support for interlaced video. MPEG-2 introduces and defines Transport Streams, which are designed to carry digital video and audio over unreliable media and networks. The MPEG-2 representation is now widely used on the commercial market on the DVD and DTV standards.

Later in 1995, a new compression standard called H.263 [35] was developed by the ITU-T. H.263 managed to be superior, in rate distortion sense, to H.261 at all bitrates. The compression performance was obtained by exploiting some technical features such as the use of median motion vector prediction, 3D variable length coding of DCT coefficients, arithmetic entropy coding and variable block sizes. The H.263 standard was later improved (early in 1998) by the H.263+ recommendation. H.263+ provided some new functionalities such as error resilience, custom and flexible video formats, higher video frame rate, etc. The most

important techniques introduced in H.263+ that improved video coding efficiency were an advance intra coding scheme, a de-blocking filter in the coding loop and the use of multiple reference frames [104]. This later technique was based on the idea that the best reference for a block being coded is not always in the closest frame in time and may be situated several frames in the past. Therefore, in the multiple reference frames technique (also known as long term prediction), all encoded blocks in a single frame do not share the same reference frame. For each block, information about the frame that has been used as reference has to be added. This implies an increment of the bitrate. However, results show that the gain in prediction error is greater than the bitrate needed to encode the reference and thus, the final rate-distortion efficiency increases.

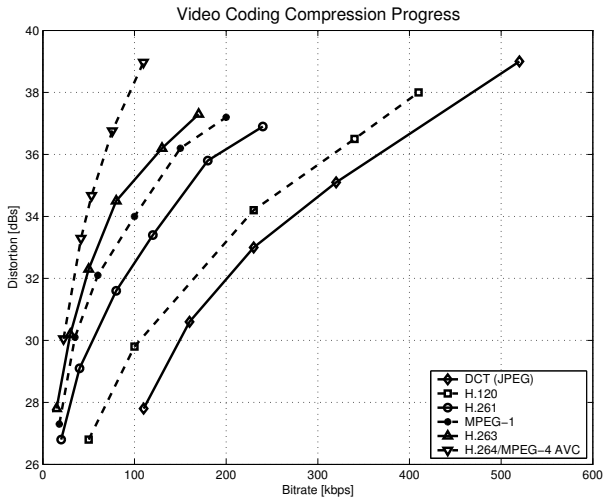


Figure 2.1: Rate-distortion curves for various coding standards. The same visual quality, in terms of PSNR, was accomplished with less bitrate as the compression representations evolved[1].

The MPEG-4 standard [75] appeared around the year 2000 (version 2). It was developed not only as an evolution to previous standards but also to provide functionalities not already present in previous compression standards. MPEG-4 introduced new techniques for video compression, such as a direct mode for B frames, quarter-pel motion estimation, global motion estimation, visual texture coding and sprite coding (or mosaic coding). Using the mosaic coding, a mosaic image of the background information is used as reference for future frames. MPEG-4 also introduced the notion of video object planes, thus video sequences can be represented as a series of video objects which can have semantic meaning. This coding technique does not focus on the compression efficiency but provides additional functionalities to the video bitstream such as, for instance, removal of video objects, allowing different visual

[1]Figure extracted from Bernd Girod: EE398B Image Communication II, Video Coding Standards: MPEG-1,2,4 no.23

qualities across video objects, etc.

Currently, the standard H.264 [1] is being developed jointly by the MPEG and ITU-T organizations. The H.264 standard does not use a DCT transform but an integer transform. Multiple reference frames (long term temporal prediction) is extensively used, together with motion estimation of 1/16 pixel precision.

Finally, newer techniques in video compression representation are focusing very heavily on wavelet transforms. Unlike DCT and integer transforms, which operate on small blocks of pixels, the Discrete Wavelet Transform (DWT) operates on the entire image, thereby eliminating blocky artifacts common in the DCT technology. Wavelet codecs also naturally provide functionality on spatio-temporal and PSNR scalability.

Figure 2.1 shows an evolution of the efficiency, in terms of rate-distortion, of compression representations. The increase of efficiency in the representations has been accomplished by an increase in the complexity of the representation itself. Fortunately, some of this complexity has been easy to overcome. For instance, the increase of computational complexity has not been critical as Moore's law (computer speed doubles every year) and hardware manufactures have helped in creating news encoders and decoders which are able to cope with new standard representations. It should be noted that, even though compression representations have also provided some other functionalities, such as error resilience, spatial/temporal/PSNR scalability or video object coding, their main goal has always been video compression.

2.2 Standards for Indexing Representations

In recent years, multimedia applications require indexing the content, browsing it, searching and retrieving it, inserting author and copyright information, etc. Therefore, new representations are needed to provide these functionalities. It is important to remark again that, in this work, the term *indexing representation* or *metadata* refer to all data structures, descriptors or extra information that describe the content in order to provide functionalities of indexing, browsing or search and retrieval. In general, compression representations might not be very well suited to index or browse the content. Take for instance any MPEG video compression representation. Browsing these representations is hard as, in order to visualize video images, they have to be decoded. Moving within the MPEG video representation is also not efficient, for instance, if the target frame is a B frame, several frames must be decoded before being able to decode the specific B frame.

Several indexing standards have recently appeared. They try to cover other functionalities rather than compression. In the case of functionalities of retrieval and consumption of content, the MPEG-21 [41] standard aims at defining a normative open framework for multimedia delivery and consumption. This framework will provide the technology needed to support users to exchange, access, trade and manipulate digital material in an efficient and

transparent way. The ISO/IEC 15938 MPEG-7 [32, 60] standard developed by MPEG and the SMPTE Metadata Dictionary [78] are two of the most well-known standards that have focused on indexing functionalities. Next sections will review in detail both MPEG-7 and SMPTE standards.

2.2.1 MPEG-7

The main objective of MPEG-7, formally called *Multimedia Content Description Interface*, is to provide standardized descriptions of audiovisual information to enable it to be quickly and efficiently searched or browsed. The MPEG-7 standard allows inter-operable searching, indexing, filtering, browsing and accessing of audiovisual content by enabling inter-operability among devices that deal with audiovisual content description. The audiovisual content described by MPEG-7 may include: still pictures, graphics, 3D models, audio, speech, video, etc.

MPEG-7 allows different granularity in its descriptions, offering the possibility to have different levels of detail. Because the descriptive features must be meaningful in the context of the application, they are different for different user domains and different applications. For example, low level description such as shape, size and trajectories can be combined with semantic information. The elements that MPEG-7 standardizes provide support to a broad range of applications (for example, multimedia digital libraries, broadcast media selection, multimedia editing, home entertainment devices, etc.).

The way MPEG-7 descriptions are used to answer user queries or filtering operations is outside the scope of the standard. The types of the content and the query do not have to be the same; for example, visual material may be queried and filtered using visual content, music, speech, etc. It is the responsibility of the search engine and filter agent to match the query data to the MPEG-7 indexing representations.

The standard provides three types of normative elements: Description Tools, a Description Definition Language (DDL), and System Tools.

Description Tools

The MPEG-7 Description Tools are composed of Descriptors and Description Schemes:

- **Descriptors:** The MPEG-7 Descriptors are designed for describing individual features (such as color, texture, motion, audio energy, etc), features of semantic objects, abstract concepts and content management processes information about the storage media.
- **Description Schemes:** The MPEG-7 DSs expand on the MPEG-7 Descriptors by combining individual Descriptors as well as other DSs within more complex structures and by defining the relationships among the Descriptors and DSs. In

MPEG-7, the DSs are categorized as pertaining specifically to the audio or visual domain, or pertaining generically to the description of multimedia.

Description Definition Language

The Descriptors and DSs are defined using the MPEG-7 DDL which is an extension of the XML Schema language [8]. An MPEG-7 description is produced for a particular piece of multimedia content by instantiating the MPEG-7 Descriptors and DSs and it is suitable for editing, searching and filtering. Refer to appendix A for an overview of the MPEG-7 DDL.

Systems Tools

The MPEG-7 Systems Tools are designed for compressing the MPEG-7 textual XML descriptions into a binary form (BiM) in order to satisfy application requirements for compression efficiency, error resilience, random access, streaming, etc. The binary form is suitable for storage, transmission and delivery of the MPEG-7 indexing representations.

Therefore, MPEG-7 Description Tools allow creating descriptions (i.e., a set of instantiated Description Schemes and their corresponding Descriptors at the users will) to incorporate application specific extensions using the DDL and to deploy the descriptions using System Tools. Figure 2.2 provides an overview of the organization of MPEG-7 Multimedia DSs into the following areas: Basic Elements, Content Description, Content Management, Content Organization, Navigation and Access, and User Interaction.

Basic Elements

The MPEG-7 Multimedia DS specification defines a number of basic elements that are used repeatedly as fundamental constructs throughout the definition of the MPEG-7 DSs. Basic elements include Schema tools, basic data-types, linking, identification and localization tools, as well as basic description elements.

The specification defines a base type hierarchy that organizes the Descriptors and DSs. The type hierarchy defines the base set of tools, including DSs, Descriptors, and Header, from which all specific MPEG-7 elements are derived. The audio and visual Descriptors and DSs extend from the abstract *DType* (Descriptors) and *DSType* (DSs), respectively. The specification also defines the *Mpeg7* root and the top-level elements, which are used to form MPEG-7 valid descriptions.

The Schema tools specification defines a Package tool that is used to describe an organization or packaging of specific Descriptors and DSs for an application. A Package can be used to organize and label the tools for ease of use and navigation. The Schema tools specification

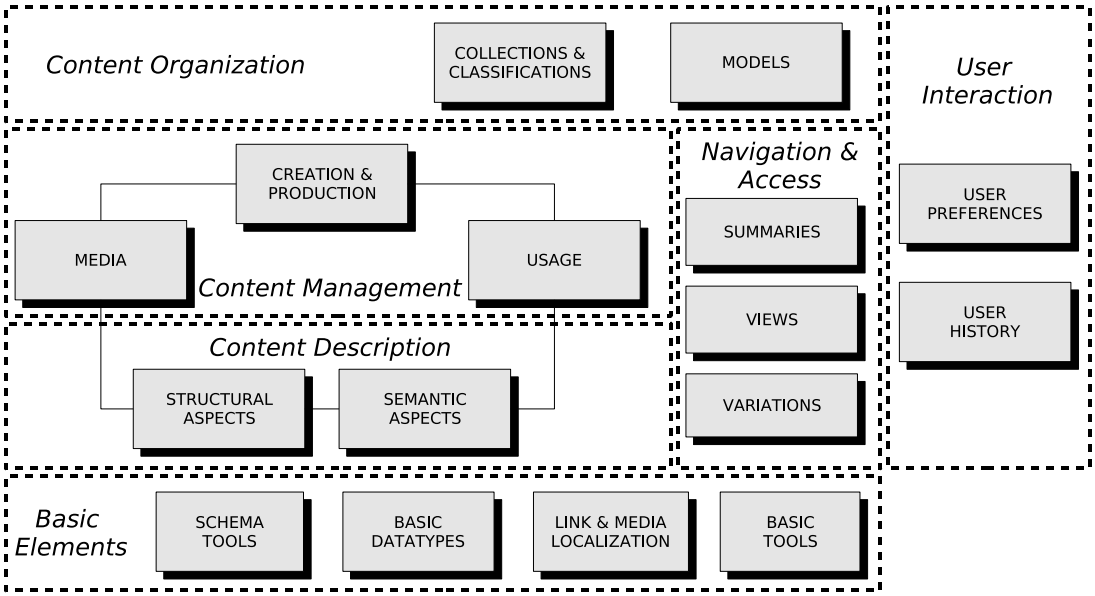


Figure 2.2: Overview of the MPEG-7 Multimedia Description Schema.

defines a *Description Metadata* DS that is used to describe indexing representations about the description itself. The *Description Metadata* DS can be embedded into any description element to describe indexing representations concerning that description. Such indexing representations include information about identifying the description (privately or publicly), the creation of the description, the version of the description, and the rights associated with the description.

Content Management

These elements describe different aspects of 1) creation and production, 2) media coding, storage and file formats and 3) content usage. Many of the components of the DSs are optional. The instantiation of the optional components is often decided in view of the specific multimedia application. The functionalities of these DSs are specified as follows:

1. *Creation Information* describes the creation and production of the multimedia content. The creation information provides a title and information such as creators, creation locations and dates. It also describes how the multimedia material is classified into categories such as genre, subject, purpose, language, and so forth. Furthermore, it provides review and guidance information such as age classification, subjective review, and parental guidance. Finally, it describes whether there exists other multimedia material that is related to the content being described.

2. *Usage Information* describes information related to the usage rights, usage record, and financial information. The rights information is not explicitly included in the MPEG-7 description. Instead, links are provided to the rights holders and to other information related to rights management and protection. The underlying strategy is to enable MPEG-7 descriptions to provide access to current rights owner information without dealing with information and negotiation directly. The *Usage Record* DS provides information related to the use of the content such as broadcasting, on demand delivery, CD sales, and so forth. Finally, the *Financial* DS provides information related to the cost of production and the income resulting from content use.

3. *Media Description* describes the storage media including the compression, coding and storage format of the multimedia content. It identifies the master media, which is the original source from which different instances of the multimedia content are produced.

Navigation and Access

MPEG-7 provides DSs for facilitating browsing and retrieval by defining summaries, views, and variations of the multimedia content.

1. *Summary DSs* provide compact highlights of the multimedia content to enable discovery, browsing, navigation and visualization of multimedia content. The *Summary* DS involves two types of navigation modes: hierarchical and sequential. In the hierarchical mode, the information is organized into successive levels, each describing the multimedia content at a different level of detail. In general, the levels closer to the root of the hierarchy provide coarser summaries, and levels further from the root provide more detailed summaries. The sequential summary provides a sequence of images or video frames, possibly synchronized with audio, which may compose a slide show or multimedia skim.

2. *View DSs* are based on partitions and decompositions, which describe different decompositions of the multimedia signals in space, time and frequency. The partitions and decompositions can be used to represent different views of the multimedia content.

3. *Variation DSs* provide information about different variations of multimedia programs, such as summaries and abstracts; scaled, compressed and low-resolution versions; and versions with different languages and modalities - audio, video, image, text, and so forth.

Content Organization

MPEG-7 provides DSs for organizing and modeling collections of multimedia content. The *Collection* DS organizes collections of multimedia content, segments, events, objects or even

content descriptions. This allows each collection to be described as a whole based on the common properties.

User Interaction

The *User Interaction* DS describes user preferences and usage history pertaining to the consumption of the multimedia material. This allows, for example, matching between user preferences and MPEG-7 content descriptions.

Content Description

These elements describe the structure (regions, video frames, and audio segments) and semantics (objects, events, abstract notions).

1. *Structural Aspects DSs* describe the multimedia content from the viewpoint of its structure. The description is built around the notion of *Segment* DS that represents a spatial, temporal or spatio-temporal portion of the multimedia content. The *Segment* DS can describe connected and non-connected segments. Figure 2.3 represents a *Segment* DS which describes one temporal video segment composed of three sub-segments.

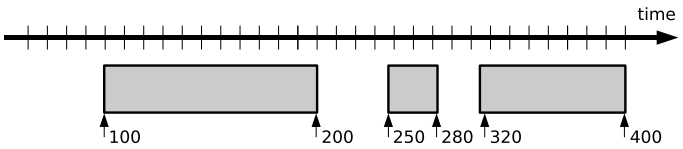


Figure 2.3: Example of segments. The temporal video segment is composed of three separated connected components. The numbers at the bottom represent the start and end time instants for each of the sub-segments.

The *Segment* DS can be organized into a hierarchical structure to produce a Table of Content for accessing or an Index for searching the multimedia content. The decomposition is described by the *SegmentDecomposition* DS which can represent any arbitrary decomposition of a segment and it can include gaps and/or overlaps. For example, in Figure 2.4, a temporal video segment is divided into three sub-segments which may also be further decomposed.

Segments can also include semantic information using textual annotations. The segments can be further described using MPEG-7 descriptors for color, texture, shape, motion, audio features, etc. As part of the work in this thesis is closely related to these descriptors, indexing representations dealing with visual information are reviewed in more detail. In particular, visual descriptors are classified in six major categories: color, texture, shape, motion, localization and others.

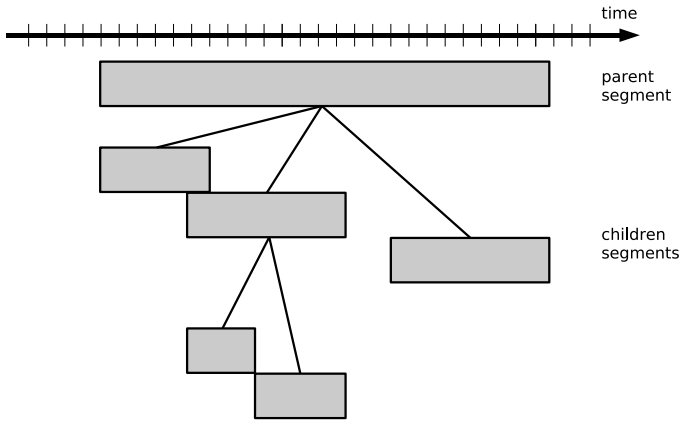


Figure 2.4: Example of a segment decomposition. The temporal video segment is decomposed into several sub-segments with gaps and overlaps.

Color Descriptors: There are seven Color Descriptors. *Color Space*, *Color Quantization*, *Dominant Colors*, *Scalable Color*, *Color Layout*, *Color Structure*, and *GoF/GoP Color*.

- *Color Space* denotes the space of color that is to be used in other color based descriptions. RGB, YCbCr, HSV, HMMD, Linear transformation matrix with RGB reference and Monochrome color spaces are supported.
- *Color Quantization* defines the uniform quantization of a color space. The number of bins which the quantizer produces is configurable. For a meaningful application in the context of MPEG-7, this descriptor has to be combined with dominant color descriptors.
- *Dominant Color* is suitable for representing local (regions) or global (entire image) features where a small number of colors are enough to characterize the color information in the region of interest.
- *Scalable Color* descriptor is a color histogram in HSV Color Space, which is encoded by a Haar transform. The *Scalable Color* Descriptor is useful for image-to-image matching and retrieval based on color feature.
- *Color Layout* represents the spatial distribution of color of visual signals in a very compact form. This compactness allows visual signal matching functionality with high retrieval efficiency at very small computational costs. It provides image-to-image matching as well as ultra high-speed sequence-to-sequence matching. One advantage of this descriptor is that there are no dependency on image or video format, resolutions and bit-depths.
- *Color Structure* descriptor is a color feature descriptor that captures both color content (similar to a color histogram) and information about the structure of this content. Its main functionality is image-to-image matching and its

intended use is for still-image retrieval, where an image may consist of either a single rectangular frame or arbitrarily shaped, possibly disconnected, regions.

- *GoF/GoP Color* descriptor extends the *Scalable Color* descriptor for video segments or collections of still images. It is more robust to round-off errors and the presence of outliers in image intensity values compared to the average histogram.

Texture Descriptors: There are three texture Descriptors: *Homogeneous Texture*, *Edge Histogram*, and *Texture Browsing*.

- *Homogeneous Texture* descriptors have emerged as an important visual primitive for searching and browsing through large collections of similar looking patterns. An image can be considered as a mosaic of homogeneous textures. These texture features, associated with the regions, can be used to index the image data. The computation of this descriptor is based on filtering using scale and orientation selective kernels.
- *Texture Browsing* descriptor is useful for representing homogeneous texture for browsing type applications. It provides a perceptual characterization of texture, similar to a human characterization, in terms of regularity, coarseness and directionality.
- *Edge Histogram* descriptor represents the spatial distribution of five types of edges, namely four directional edges and one non-directional edge. It primarily targets image-to-image matching (by example or by sketch), especially for natural images with non-uniform edge distribution. The image retrieval performance can be significantly improved if the *Edge Histogram* descriptor is combined with other descriptors such as the *Color Histogram* descriptor.

Shape Descriptors: There are three shape descriptors: *Region Shape*, *Contour Shape*, and *Shape 3D*.

- *Region Shape* descriptor expresses pixel distribution within a region. It can represent any shape and it is robust to minor deformations along the boundaries of the object.
- *Contour Shape* descriptor captures characteristic shape features of an object or region based on its contour. It uses a curvature scale-space representation, which captures perceptually meaningful features of the shape.
- *Shape 3D* descriptor provides an intrinsic shape description of 3D mesh models. It exploits some local attributes of the 3D surface.

Motion Descriptors: There are four motion Descriptors: *Camera Motion*, *Motion Trajectory*, *Parametric Motion*, and *Motion Activity*.

- *Camera Motion* descriptor characterizes 3D camera motion parameters. It can be automatically extracted or generated by capture devices. The camera

motion descriptor supports basic camera operations such as fixed, panning, tracking, tilting, zooming, etc.

- *Motion Trajectory* descriptor describes the localization, in time and space, of one representative point of a region. This descriptor is useful for content-based retrieval in object-oriented visual databases.
- *Parametric Motion* descriptor describes the motion of regions as a 2D parametric model. Specifically, affine models include translations, rotations, scaling and combinations of them. Planar perspective models make possible to take into account global deformations associated with perspective projections and quadratic models allow to describe more complex movements.
- *Motion Activity* descriptor enables to accurately express the activity of a given video sequence. The activity descriptor is useful for applications such as video re-purposing, surveillance, fast browsing, dynamic video summarization, content-based querying etc.

Localization Descriptors: There are two descriptors for localization: *Region locator* and *Spatio-temporal Locator*.

- *Region Locator* descriptor localizes regions within images or frames by specifying them with a brief and scalable representation of a box or a polygon.
- *Spatio Temporal Locator* describes spatio-temporal regions in a video sequence, such as moving object regions, and provides localization functionality. The main application of it is hypermedia, which displays the related information when the designated point is inside the object.

Others: This category includes the *Face Recognition* descriptor. It can be used to retrieve face images which match a query face image. The descriptor represents the projection of a face vector onto a set of basis vectors which span the space of possible face vectors.

2. *Conceptual Aspects* describe the multimedia content from the viewpoint of real-world semantics and conceptual notions. The *Semantic* DS involves entities such as objects, events, abstract concepts and relationships. The *Segment* DS and *Semantic* DS are related by a set of links, which allows the multimedia content to be described on the basis of both content structure and semantics together.

Note that, most of the MPEG-7 DSs are linked together, and in practice, the DSs are included within each other in MPEG-7 descriptions. For example, Usage information, Creation and Production, and Media information can be attached to individual *Segment* DSs identified in the MPEG-7 description of multimedia content structure. Depending on the application, some aspects of the multimedia content description can be emphasized, while others can be minimized or ignored.

2.2.2 SMPTE Metadata Dictionary

The SMPTE Metadata Dictionary Structure defined in [78] covers the definition of indexing representations for all types of content (video, audio and data). The Metadata Dictionary Contents RP210 defines a registered set of indexing representations for association with content or other representations.

Metadata Dictionary Structure

The Metadata Dictionary Structure provides flexibility in capturing indexing representations and exchanging them among applications through a standardized hierarchy of universal labels. Indexing representation classes are collections of indexing representations with common characteristics or attributes. Additional classes are provided for user-defined elements.

The Metadata Dictionary Contents Recommended Practice references an individual item or indexing representation using a two part 16 byte universal label that is numerical (and hence language independent) and unique. The first eight bytes label the second ones as a *tag* in a specific version of a designated Metadata Dictionary (*tags* are defined in the SMPTE document "Data Encoding Protocol Using Key Length Value (KLV)").

This *tag* is used to index the meaning or definition of the indexing representation. The actual indexing representation is stored in the *value* field which is stored in the second eight bytes of the 16 byte universal label. The Dictionary also contains information on the required format of *values* and the allowable range of *values* (if applicable) either as a list or as a bounded range.

The Metadata Dictionary is organized into nodes and leaves. To aid the management of the dictionary, these nodes and sub-nodes are assigned *tags* to which no *value* is assigned, thus giving clear "breaks" in the structure. Entries within a sub class form leaves, which are the data elements themselves. Other levels of the dictionary structure can be derived from the *tag* structure in the Metadata Dictionary Contents RP210 and are detailed in the SMPTE document "Node Structure for the metadata Dictionary".

Individual Indexing Representation Classes

Within the Metadata Dictionary, indexing representations are organized into a hierarchical structure, where each indexing representation is assigned to a class as shown in the overview of Figure 2.5. The initial set of indexing representation classes in this standard consists of:

- *Class 1:* Identification and Location
- *Class 2:* Administrative
- *Class 3:* Interpretive

- *Class 4:* Parametric
- *Class 5:* Process
- *Class 6:* Relational
- *Class 7:* Spatio-Temporal
- *Class 14:* Organizationally Registered for Public Use
- *Class 15:* Organizationally Registered as Private
- *Class 16:* Experimental

The number of indexing representation classes can be extended in the future to a maximum of 127. Although dictionary classes can be populated with any indexing representation (such as that associated with still images, audio, graphics, etc.), additional new classes may be created up to that limit to deal with specific indexing representation characteristics or attributes.

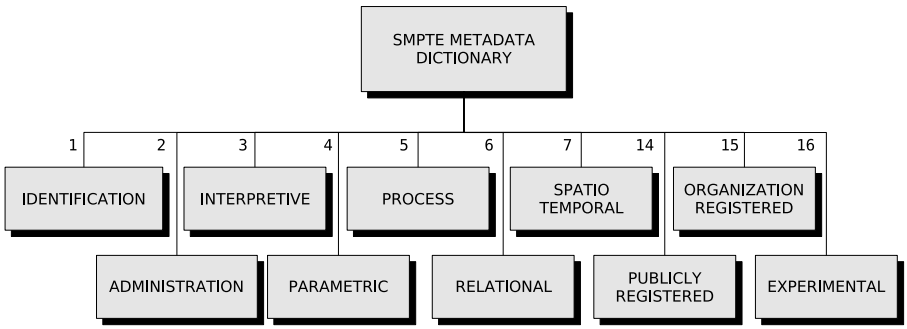


Figure 2.5: SMPTE Indexing Representation Class Structure.

Class 1: Identification and Location

Indexing representations in this class consist of identification information that describes the overall content. A critical part of *Class 1* indexing representations is unambiguous identification of the content using a single, recognized number or label such as the SMPTE Unique Material Identifier. Information in this class includes global and local identifiers as well as identifying information about the indexing representations themselves (so-called "meta-metadata"). Indexing representations in this class are similar to the Content Management MPEG-7 descriptors defined in Section 2.2.1. Examples of sub-class titles in this class are Globally Unique Identifiers, ISO Identifiers, Object identifiers, Device Identifiers, Unique Identifiers, Local Locators and Titles.

Class 2: Administration

Indexing representations in this class consist of administrative or business data that describe information about the content or indexing representations that are relevant to its application. Information on authorized use, restrictions on use and encryption are in this class. Cost information and information needed to protect intellectual property or to protect ownership shall also be contained in *Class 2*. This class is similar to the MPEG-7 Content Management. Examples of sub-class titles in this class are Supplier Rights, Financial Information, Security, Publication Outlet, Participating Parties, Broadcast and Repeat Statistics.

Class 3: Interpretive

Indexing representations in this class consist of descriptive information that is normally considered either a subjective, a human-generated description of the content or a computational result from machine examination of the content. Interpretive information consists of, but is not limited to, textual terms (e.g. keywords, narrative summary, titles, genre categories, scripts, etc.), or computational metrics (e.g. color histograms, texture maps, object shapes, facial features, etc.). Information in *Class 3* shall be principally used for indexing, cataloging, administering, searching, and retrieving the content. The low-level descriptors included in this class are quite similar to the MPEG-7 low-level visual descriptors. Examples of sub-class titles in this class are Fundamental (such as ISO Language Code, length and time systems), Descriptive (Human-Assigned), Categorization, Assessments and Descriptors (Machine-Assigned or Computed).

Class 4: Parametric

Indexing representations in this class consist of information that describes the technical characteristics of the camera, sensor, or system that originates the content. Information about the technical characteristics of the content is also provided, including but not limited to its creation parameters and the configuration of the originating system. This set of descriptors describes how the content has been encoded and is closely related to the MPEG-7 Media Description. Examples of sub-class titles in this class are Video Content Encoding Characteristics, Audio Content Encoding Characteristics, Data Content Encoding Characteristics, Metadata Encoding Characteristics, Audio test parameters, Film Pulldown characteristics, Fundamental sequencing and scanning, MPEG Coding Characteristics and Timecode Characteristics.

Class 5: Process

Indexing representations in this class consist of information that describes how the content was processed or otherwise changed or enhanced after its creation. This class shall include many of the parameters in an edit decision list. Additional information in *Class 5* shall be

an audit trail (heritage) of all changes to the original content over time. Also a record of compression/decompression steps and any changes in storage media or format is included. It describes a certain number of events that may have affected the content. Examples of sub-class titles in this class are Process indicators, Manipulation, Downstream Processing History, Enhancement or Modification, Audio processor settings (Device-specific) and Editing Information.

Class 6: Relational

Indexing representations in this class consist of information that describe the relationships between objects in the content or between any combination of content, objects and other indexing representations. Examples of sub-class titles in this Class are Generic Relationships, Content to Content Relationship, Related production material, Numerical sequence, Relationship structures, etc.

Class 7: Spatio-temporal

Indexing representations in this class consist of information about aspects of the content or the originating camera, sensor, or system relating to time, place or space. Geo-spatial information in *Class 7* shall be any information that defines the positions or places (either absolute or relative) of objects, scenes, individuals, or any other component of the content. Temporal elements such as dates, time codes, synchronization marks, temporal keywords, and motion vector parameters shall also be part of *Class 7* indexing representations. Examples of sub-class titles in this class are Position and Space Vectors, Absolute Position, Image Positional information, Rate and Direction of Positional Change, Abstract Locations, Angular Specifications, Distance Measurements, Operational Date and Time Information, Absolute Date and Time, Relative Durations, Rights Date and Time, Setting Date and Time (Characterized Time Period), Delay and Latency.

Class 14: Organizationally Registered for Public Use

Indexing representations in this class consist of individual elements of indexing representations that have been registered by a specific user organization or individual and are therefore reserved and managed separately from the other classes (1-7). Information about Publicly Registered indexing representations appears in the appropriate sections of the published Metadata Dictionary Contents. *Class 14* is managed by the SMPTE Registration Authority.

Class 15: Organizationally Registered as Private

Indexing representations in this class consist of individual elements or groups of elements of indexing representations that have been registered by a specific user organization or indi-

vidual. They are reserved and managed separately from the other classes (1-7). Information about Organizationally Registered indexing representations is not made public but the indexing representations *tags* are publicly identified in the Metadata Dictionary Contents and are reserved for use by the registered organization. *Class 15* is managed by the SMPTE Registration Authority.

Class 16: Experimental

Indexing representations in this class consist of information whose definition and use does not need to conform to the definitions in the Metadata Dictionary. *Class 16* is intended for use in multimedia research or other limited access, experimental environments where experimentation with new indexing representations and applications does not depend on strict conformance to approved standards.

Chapter 10 studies these MPEG-7 and SMPTE descriptors focussing on the subsets that more clearly present a potential for video coding and justifies why some specific descriptors, in particular MPEG-7 descriptors, are chosen.

Chapter 3

Synergy between Compression and Indexing Representations

Even though compression and indexing representations provide different functionalities, both representations can take advantage of each other. Both representations describe the same content, therefore, it is natural to think that both representations may benefit from one another. The previous chapter has reviewed some standards for both indexing and compression representations. This chapter reviews some techniques that have appeared in the literature where this synergy between representations is exploited. Section 3.2 reviews techniques that exploit compression representations to create indexing ones. Section 3.3 reviews techniques in the literature that enhance compression representations using indexing ones.

One of the best examples of the synergy between these representations is the use of motion vectors. Motion vectors have been employed extensively in video compression in order to improve the coding efficiency. Using motion vectors, the temporal redundancy of the video scene is greatly reduced. Instead of compressing video frames separately and, thus, only exploiting intra-frame redundancy, a motion estimation is performed hence the current frame can be predicted from past frames. Only the motion vectors and the prediction error image are coded.

But motion vectors can also be used to provide indexing functionalities. For example, queries for videos with a characteristic motion, such as zooms, pans, etc., can be constructed and searched using motion vectors that have been extracted with the goal of compression functionality. For instance, inspecting the motion vectors of Figure 3.1, it is very easy to say that the video scene corresponds to a zoom in.

In the case of motion vectors, compression and indexing representation share the same data type. However, it is obvious that motion vectors extracted for compression may not be the best information to index a video. They are extracted with a compression goal in mind and might not correspond to the real motion of the video scene. Therefore, an extra step to

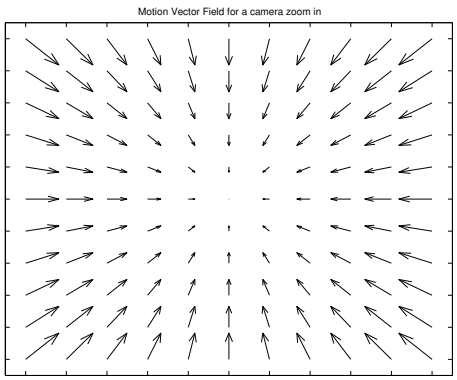


Figure 3.1: Resulting motion vectors (block matching) for a camera zoom in motion.

process motion vectors, or to compute new motion information from the original content may be needed in order to provide other representations more suitable for indexing purposes.

This thesis aims at studying the synergy between compression and indexing representations. In Part II of the thesis, compression representations are investigated in order to improve indexing ones. In particular, mosaics are used to create compression representations that can also be used to construct useful indexing representations to describe video shots. In Part III of the thesis, compression representations are improved with the help of indexing ones. In particular, standard MPEG-7 indexing representations are investigated to improve the coding efficiency of standard H.264 compression representations.

3.1 Application Scenarios

Before going into explaining the details of the synergy between representations, it is important to define the different scenarios where this synergy can be studied. When creating compression representations which are aware of indexing ones, several scenarios can be foreseen. These scenarios can be divided into different categories depending on the availability of the indexing representations at the encoder, or both at the encoder and the decoder. Some applications will imply the availability of the indexing representations while others will not assume that the indexing representations are accessible for both encoder and decoder. In this later situation, video codecs that are aware of indexing representations will be forced to extract their own indexing representations. If the indexing representations are also needed for the decoding process, they will have to be sent together with the content if not available, or cannot be re-created on the decoder.

The different scenarios can be grouped into the following three cases: indexing representations are only used in the encoder, indexing representations are needed by encoder and decoder but are available for both and, finally, indexing representations are needed by encoder

and decoder but, as they are not available and cannot be re-created at the decoder side, they must be streamed together with the content.

3.1.1 Indexing Representations Only Used in the Encoder

In some situations the encoder will make use of the indexing representations to simply optimize its encoding strategy. The resulting bitstream (coded using the available indexing representations) is still compatible with indexing representation unaware decoders. Thus, standard codecs are able to decompress the bitstream without accessing the representation. In this scenario, the encoder is only able to use indexing representations to efficiently select its encoding strategy by, for instance, deciding the best encoding parameters for one particular content.

This scenario is similar to one used by various commercial MPEG-2 encoders [9, 61, 94]. In particular, some commercial encoders try different encoding strategies for every possible quality factor Q and GoP structure configuration. However, in this case, the search for the optimum encoding parameters is very complex as the number of possible combinations is extremely high. In these situations, available indexing representations may help reducing the initial search phase or simply removing the need of a search completely as the already existing indexing representations may point to the best possible encoding parameters. Chapter 14 shows how MPEG-7 indexing representations may help in the GoP structure selection of current hybrid encoders. Normally, the GoP structure is usually fixed over the entire sequence as it is very difficult to efficiently select the type of frame (as current decisions affect future ones). However, existing indexing representations may help in the decision of the best GoP structure to choose in terms of overall bitrates savings for the same quality.

3.1.2 Indexing Representations Available for both Encoder and Decoder

In some other situations, in order to fully exploit the existing indexing representations, the encoder will have to severely modify its encoding strategy and the bitstream compatibility with classical decoders will be broken. In these situations, the decoder will also need the indexing representations to correctly extract the video content from the bitstream. Fortunately, there are applications where the indexing representations will be available at both the encoder and the decoder sides at no cost.

Consider for instance a scenario where a user is browsing a database of movies from a video content provider. The user is able to browse the archive, search and query for different movies, etc. During this initial search phase, the user gets a local copy of the indexing representation thus he/she is able to interact with the database. Once the user has selected a movie, and before the downloading starts, both encoder (at the server side) and decoder (at the user side) have the common knowledge of the indexing representations used in the

browsing, query and searching steps. This information can be used by both the encoder and the decoder to improve the coding efficiency and no further transmission of the indexing representations needs to be done.

Applications can make use of this scenario to improve the encoding strategy. In this case, the use of indexing representations can go further than in the previous scenario as the encoder may modify its encoding strategy even if that implies to modify the decoder. In this scenario the transmission of video content is preceded by a searching step that facilitates the access to the indexing representations by the decoder. This scenario is used, for instance, in the mosaic coding scheme employed in Section 8.2 of this thesis. In general, applications that exploit indexing representations available at both encoder and decoder may fit into two different categories depending on how the indexing representations are employed:

- If the content provider has no severe restriction on the amount of storage space, the provider can have the video content encoded in various bitstreams. These bitstreams encode the same video content but using different indexing representations combinations. For instance, content may be encoded using three bitstream combinations; without indexing representations, using a sub-set of the indexing representations available and using all possible existing indexing representations. When the user requests the transmission of the content, the provider application is able to choose the bitstream to send (between the three possibilities) depending on whether the user has obtained the indexing representations needed for the different bitstreams.

- Other applications may choose to re-encode on the fly the video content when the transmission is requested. The provider encoder stores the indexing representations used during the browsing and searching steps and uses only those available in the decoder to increase the coding efficiency. Therefore, video content is encoded on the fly after the user has performed the search. This application can open new possibilities for the encoding as for instance indexing representations on the user side can inform to the encoder of possible similar content (such as previous episodes of the same series or previous news programs from the same channel). With that information the encoder may choose to reference the new content with content already stored in the user end. For instance, the starting headers of the news program can be omitted and used from the already stored content of the user. The drawback is the need of the real time encoding which may not fulfill the operational request of the final user.

3.1.3 Indexing Representations Streamed with Video Content

In some situations, the indexing representations will need to be sent together with the content. This scenario occurs when the video content provider uses a particular indexing representation to encode the content but this indexing representation is not available to the decoder. In

that case, the encoder needs to send the used indexing representation together with the content thus the decoder application is able to access the decoded content. Fortunately, some standards such as MPEG-4 or H.264 include parts of the bitstream where indexing representations can be easily attached. An example of this scenario may be the use of MPEG-7 color descriptors as the available indexing representations. These descriptors can be used to select reference frames in video sequences. Unfortunately, the descriptors are also needed in the decoder (to select the same reference frames) and, therefore, they must be streamed together with the content if not available. Chapter 12 in Part III of this thesis studies this technique assuming both scenarios where indexing representations are or are not available to the decoder.

In general, when indexing representation aware video codecs are used, the scenario where indexing representations are needed by the decoder must be carefully studied. The penalty of streaming the indexing representations together with the content may drop the coding efficiency of indexing representations aware codecs. In this scenario, the bitrate needed to send the indexing representations must be compared with the bitrate savings of the indexing representations aware codec. Obviously, if the indexing representations bitrate exceeds the bitrate savings then it must be reconsidered if the complexity of using indexing representations aware video codecs is useful, even if these indexing representations support extra functionality.

3.2 Indexing Representations using Compression Representations

Making indexing representations work together with compressed content has been seen as a useful an interesting approach by the image and video community. Historically, compression representations have been developed prior to indexing ones. In that sense, most of the multimedia content is already present in compressed form. Therefore, it is useful to create indexing and browsing applications that already work with compressed content.

For instance, in video databases with a huge amount of video content stored in compressed form, analysis tools can compute indexing representations by using directly the compressed content, such as extraction of key-frames for browsing, etc. In the same scenario, the query tool can also create and search new indexing representations on the fly. For instance, the user may be able to select a video, stored locally in the user machine or remotely in the database, and search for similar videos in the database. In this case, compression aware indexing tools can create indexes on the fly (from compressed content), and match between a set of compressed videos allowing queries on demand.

3.2.1 General Indexing Representations

Extracting indexing representations for fast and accurate search & retrieval functionalities from compressed content have been extensively developed and investigated by the community. The common idea for these indexing techniques is to select some basic set of features of the video which are able to identify and discriminate between the other videos in the database. In general, the set of features that can be selected for video indexing are based on the content of the video frames such as color, texture, shape, motion, spatial relationships, etc. An extensive review of general indexing techniques can be found in [4, 26].

One example of indexing representations that may benefit from compression representations is, for instance, some of the MPEG-7 descriptors reviewed on Section 2.2.1 [37]. The *Motion Activity* descriptor may employ motion vectors from compressed sequences to compute a simple descriptor which includes attributes of intensity, direction and spatial localization of a video sequence activity [14, 74, 92]. These attributes can index the video sequence into high (such as action movies or sports) to low (such as interviews) motion sequences. The MPEG-7 *Color Layout* descriptor has also a very close relation with compression representations. This descriptor uses the DCT coefficients of color images to specify a spatial distribution of colors for high-speed retrieval and browsing [46]. These DCT coefficients may be directly extracted from the compressed content if the same cosine transformation is used.

However, indexing representations cannot only take advantage of compression representations by being extracted from them. Another interesting approach to exploit the synergy between both representations is to share the same data for compression and indexing, such as for instance, motion vectors or mosaics.

3.2.2 Video Shot Indexing Representations

Part of the work in this thesis exploits compressed content to create indexing representations that describe video shots. For video indexing applications, representing video scenes usually includes an initial phase which structures the original content. A classical structuring approach consists in detecting shots in a video sequence and to group them into scenes [73, 83, 106, 107]. This grouping allows representing the scene into layers or hierarchical representations which allows representing each shot of the video scene independently of each other.

Key-frame Representations

The description of shots is often based on key-frames. In that case, several key-frames are used to represent and browse a set of shots [110]. In the same framework of this thesis, key-frame extraction algorithm can also benefit from a synergy between compressed representations. Some key-frame extraction techniques examine the DCT coefficients in the compressed domain. For

instance in [44], key-frames are extracted using an accumulated histogram of DC images extracted from the DCT coefficients of each frame of the MPEG sequence. In [20], dominant color components are computed directly from DCT coefficients of MPEG bitstreams. The dominant color components are used to select meaningful key-frames for each shot.

Other key-frame extraction techniques are based on the motion information in the compressed domain. For instance in [13], a relation is proposed between the motion activity of a shot and the number of key-frames needed to represent it. The spatial activity of motion can also be considered, for instance in [69], key-frames are constructed only when the motion activity of motion vectors in the center of the frame is relevant. The temporal distribution of macroblocks types in MPEG frames is used in [7] to compute dominant frames which are used as key-frames. Dominant frames are composed of I or P frames used as prediction reference for most macroblocks from a subsequent B frame.

Other key-frame extraction techniques propose to combine both motion and DCT information of compressed content. In [108], key-frames are extracted based on detecting changes by global motion compensating DC coefficients of MPEG frames. Other techniques propose different methods such as stochastic approaches where key-frames are computed by minimizing a cross-correlation criterion among compressed video frames [15].

Content based Representations

Other methods for shot representation involve a deeper analysis of the video content than only extracting key-frames. In that case, the content of video frames can be studied and more complex content based representations can be created. Content based representations have been deeply studied in literature [12, 63, 111]. These representations provide an abstraction layer that represents the content of the video scene. For instance, in [18], image partition sequences are used to create a simple graph [64] representing the video sequence. This graph can be tracked over the video sequence in order to identify regions that become occluded and reappear through the sequence. A more sophisticated approach for video representation involves the analysis of the spatio-temporal content of the video scene. For instance in [102], the representation of a shot is composed of a set of layers. Each layer contains a density map (and the corresponding mask) with the additive values of each pixel. The set of layers is ordered in depth and velocity maps are used to define how the layers are warped over time. In [86], a simultaneous multiple motion estimation is performed to create a layered representation of the video shot content.

Content based indexing representations for video shots can also exploit the synergy between compression representations. In this framework, mosaics are often used to represent the background information of shots. Mosaics are panoramic views of the background components that are visible during the video shot. Mosaics are used in compression representations to improve the coding efficiency of current video encoders [29, 96]. Even if some work related

to the use of mosaics in different coding standards have appeared in the literature (such as in MPEG-2 [28], H.264 [58] or in wavelet codecs [10]), the principal use of mosaics in video coding is presented with the mosaic coding scheme of the MPEG-4 standard [75].

Mosaic coding techniques involve a generation of a mosaic which is used as reference for later frames of the video. The mosaic can be generated on-line (while the video is encoded) or off-line (prior to the encoding process) [68]. For the results of this thesis, it will be assumed that the mosaic is generated off-line and, therefore, it can be exploited for other functionalities. For the mosaic generation, the global motion of the shot is estimated. For such reason, pixels that follow the global motion are assumed to be in the background. After the estimation, frames are warped and blended to form the mosaic. During the last years, different techniques have been studied to generate the mosaic. For instance, Lee *et al.* proposed a layered coding system using the affine motion model [52]. In [89], a recursive closed-loop long term prediction scheme was used to build the mosaic while in [16], a hierarchical global motion estimation technique is employed. Lu *et al.* proposed an arbitrary-shape spatial prediction and a reliability-based blending scheme to construct the mosaic [57].

Some mosaic techniques focus on improving the background/foreground classification of pixels in the mosaic. In [38], a simple threshold is applied to the motion estimated difference image (between the mosaic and the original). In [56], the previous difference is segmented using an *opening* operator. In [89], a reliability mask is defined to indicate the classification between background and foreground regions on the shot. However, segmenting foreground and background regions using pure motion-based algorithms fails when video scenes present rapidly changing backgrounds, when foreground objects present little motion with respect to the camera or when foreground objects have a low contrast with respect to the background. This thesis presents a new technique (detailed in Chapter 6) to remove foreground regions using morphological connected operators [21, 22] combining both motion and spatial information.

The main challenge of these shot representation techniques based on mosaics is, however, how to represent the foreground regions of the scene. Several techniques, such as the one proposed in [86], employ multiple motion estimations to create several mosaics for each dominant motion of the scene. However, using mosaics is not always enough when foreground regions are not rigid and their shapes vary through the shot. To solve this problem, a semi-automatic mesh-based mosaic representation is proposed in [99] which is able to cope with some deformations and occlusions of foreground objects.

In [11], foreground regions are detected and placed in the mosaic on a single time instant of their trajectory. In [27], Irani *et al.* introduced the notion of *synopsis* mosaics where mobile foreground objects are superimposed to the mosaic representation in order to indicate their relative trajectories to the global background motion. In synopsis mosaics, however, only the trajectory of foreground objects is displayed hence the information of the foreground

shape is lost. Other techniques, such as the one proposed in [88], improved the synopsis mosaics by adding several time instants of foreground regions on the mosaic. Fusiello *et al.* employed a scheme for representing shots in [17] using a mosaic and foreground object segmentation. Foreground regions are segmented following a singular value decomposition and are represented on a layered graph.

In this thesis, a new indexing representation for video shot is presented in Part II. It exploits the synergy of compression representation with the use of mosaics. Foreground regions are extracted from the video shots by combining motion and spatial information. In particular, morphological tools [85] such as connected operators and watersheds [65, 100] are used to segment foreground regions from the video frames. Instead of adding extracted foreground regions back into the mosaic, a new structure called *key-region* is built to represent each detected foreground region. As foreground regions may not be rigid during the entire shot duration, several key-regions can be used to represent a unique foreground object. The final shot representation is based on the mosaic to represent the background and several key-regions to represent the foreground regions of the scene.

3.3 Compression Representations using Indexing Representations

Exploiting indexing representations to improve compression has not been investigated by the image and video community as deeply as the synergy of compression representations to improve indexing. Indexing representation appeared historically later and, therefore, it was more obvious to integrate indexing representation with compression than the other way around. In order to exploit the synergy between indexing representations and compression, questions such as "is this specific index representation useful for coding?" must be answered. In particular, one can wonder if, for instance, using the histogram of an image can be useful to improve the coding of that image. Or if using motion information of objects can be used to further refine the reference of that object.

It is important to remark that this thesis is focused on improving compression techniques based on already existing indexing representations and not any other extra-information. The use of extra-information to improve coding is employed on most common coding strategies (such as the use of motion vectors). However, the work on this thesis only focuses on existing indexing representations and, in particular, the MPEG-7 and SMPTE standards. Even though indexing representations are extracted with other functionality in mind, results in this thesis will show that indexing representations may still be useful for video coding.

In general, this line of research is rather new and few contributions have been reported in the literature. An initial contribution can be found in [90] and [91] where the MPEG-7 *Parametric Motion* descriptor is used to improve the motion estimation and compensation

step in advanced prediction schemes (Global Motion Compensation) for the standard H.264. A rate-distortion optimization method is used to identify macroblocks that are only affected by the global motion of the scene. For such macroblocks, no prediction error is transmitted and macroblocks are reconstructed using only the MPEG-7 *Parametric Motion* descriptor on the decoder end.

In [70], MPEG-7 texture descriptors are used to signal the presence of detailed texture within the image. A texture analyzer identifies the texture regions of the image with no important subjective details. These texture blocks are skipped in the encoder and separately synthesized in the decoder using a synthetic texture generator. Both texture analyzer and synthesizer are based on MPEG-7 texture descriptors.

Other techniques have proposed the use of indexing representations to code human faces. The eigen-faces provided by the indexing representation generate an eigen-space that is used to project the face to be coded. The projection, together with a possible error image, is sent to the decoder which uses the same MPEG-7 eigen-space to recreate the original face. However, results in [76] show that the direct use of the MPEG-7 *Face Recognition* descriptors is not useful for coding as MPEG-7 face descriptors cannot provide a useful representation of an arbitrary face image. However, the authors propose to also encode the reconstruction error of representing the face with the MPEG-7 descriptors using an adaptive principal component analysis (APCA). This second scheme is able to outperform H.264 when encoding faces.

In this thesis, four new techniques that exploit indexing representations to improve coding are proposed [23, 24]. Part III of this thesis is devoted to present and study these indexing representation aware video codecs.

Part II

Using Compression Representations to Improve Indexing Representations

Chapter 4

Motivation

4.1 Introduction

In this thesis, a new technique to represent and summarize video shots is proposed. The technique is based on mosaics that can be shared between compression and indexing representations (Section 5.1 gives a more complete overview of the exploitation of mosaics for both compression and indexing). The proposed representation uses the mosaic to segment foreground regions and to group them into key-regions. These key-regions describe foreground objects that do not follow the background motion of the scene.

The motivation of this particular representation of video scenes comes from the MPEG-7 proposals p185 and p186 [79, 80] sent to the MPEG committee in October, 2000. The p186 proposal focuses on describing a MPEG-7 Description Scheme for still images while p185 describes entire video sequences. Both proposals were well accepted by the MPEG committee and the image/video community, and were the grounds for the current image and video sequence descriptors in MPEG-7. In particular, the *StillRegion*, *Mosaic* and *MovingRegion* description schema in the MPEG-7 standard share similar concepts with the proposals. The following section makes a short review of both proposals.

4.2 MPEG-7 p185 & p186 Proposals

The *Still Image* and the *Video* DSs described in the proposals are inspired by the classical way of describing written documents such as books. The two basic tools used to search and retrieve information in a book are the Table of Contents and the Index.

The Table of Contents is generally a hierarchical representation that splits the document into elementary pieces (chapters, sections, subsections, etc.). The order in which the items are presented follows the linear structure of the book itself. As a result, the Table of Contents is a representation of the linear structure of the book. However, the goal of the Index is

not to define the linear structure, but to define a set of potentially interesting items and to provide references to the book sections where these items are discussed. The items are selected because of their semantic values. Obviously, a given item may appear in several sections of the book. In many cases, the Index is also presented in a hierarchical fashion to allow fast access to the item of interest to the user.

4.2.1 MPEG-7 Still Image Description Scheme Proposal

The proposal p186 was designed to describe the visual content of a still image. The goal of the *Still Image* DS is to structure the descriptors related to the visual information contained in the image. It describes the visual appearance, its structure and its semantic content. To describe images, an approach similar to the one used to describe books may be used. It involves two hierarchical structures represented by trees, the Region and Object Trees.

The Region Tree plays the role of the Table of Contents. Its main purpose is to describe the spatial organization of the image. The nodes of the tree represent connected components of the space called *regions* [1]. The structure of the Region Tree defines the inclusion relationship between elementary items as in a Table of Contents of a book.

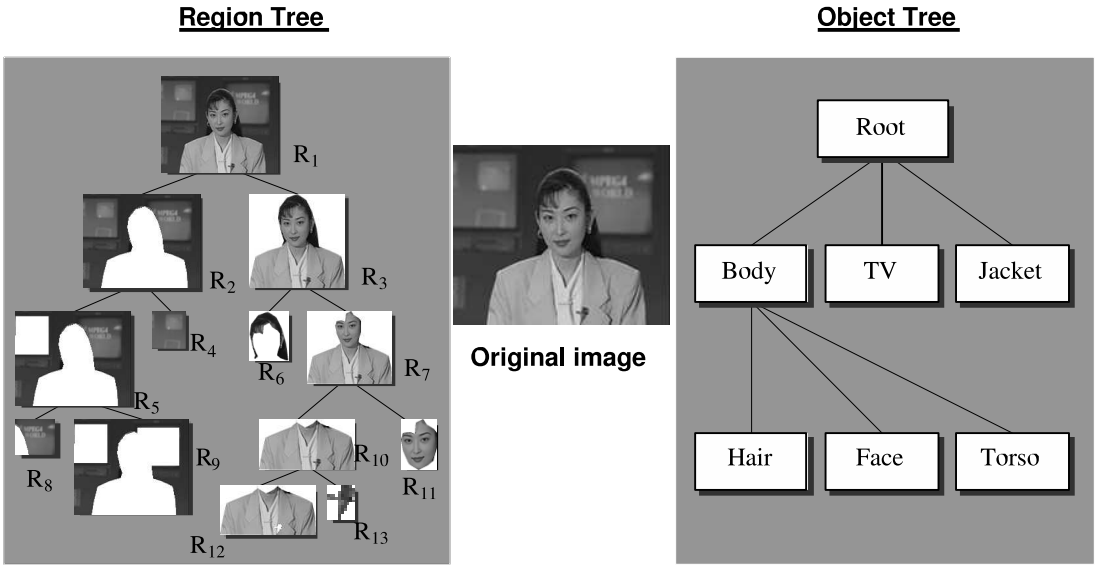


Figure 4.1: Simple example of Region and Object Trees describing a still image.

A very simple example is illustrated in the left side of Figure 4.1. The entire original *Akiyo* image can be seen as the root of the tree. The tree defines how this global region can be subdivided into more elementary regions. Descriptors are attached to each node in the

[1] The term *region*, and not *object*, is used because regions themselves need not to have a clear semantic meaning as for example region R_9 in Figure 4.1

Region Tree and describe visual properties of the corresponding regions, in particular color, spatial and geometrical characteristics. The Region Tree could be an arbitrary tree, which is a tree where each node can have an arbitrary number of children. In this proposal an inclusion relationship between two regions (the fact that region A is included or not in region B) is used. As a result, the tree describes how a region can be decomposed into two components, known as a *Binary Partition Tree* [81, 82].

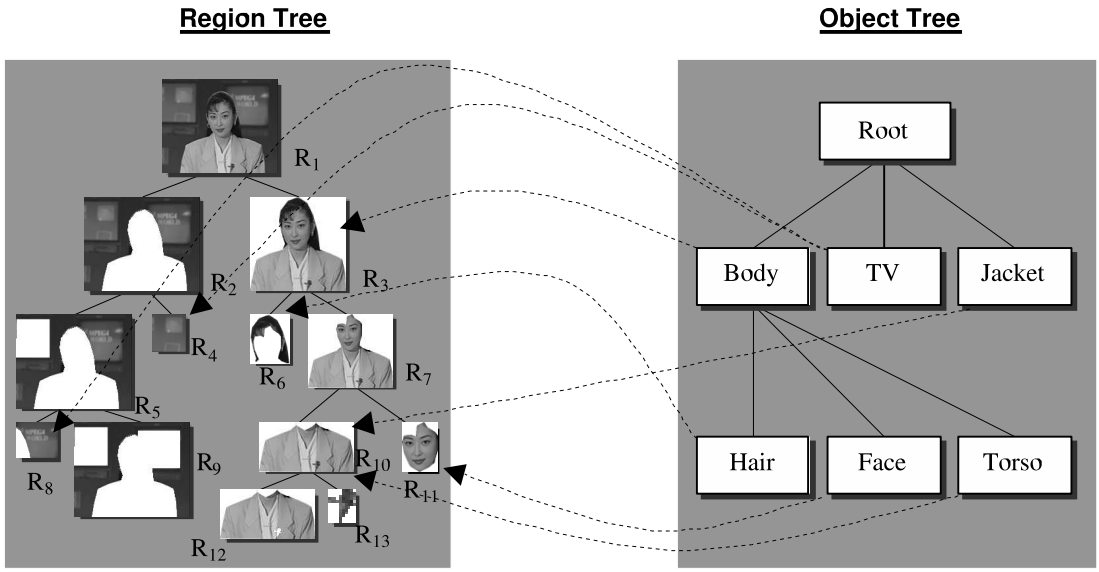


Figure 4.2: Example of references from the Object Tree to the Region Tree.

The Object Tree is the image Index. It is composed of a list of objects with a semantic meaning. As an Index, the most important functionality of this tree is to relate its objects to regions of the Region Tree. An example of an Object Tree is shown in the right side of Figures 4.1 and 4.2. The relationship between nodes in the Object Tree is of the type *is-made-of*. For example, the object *Body* is made of sub-entities called *Hair*, *Face* and *Torso*. Each object in the Object Tree has to refer to one or several regions in the Region Tree. For example, in Figure 4.2, the object *Body* refers to region R_3 and TV to regions R_4 and R_8. Conversely, one region may be referred to by several objects. This is the case for regions involving various semantic meanings. For example in Figure 4.2, region R_{10} is referred at the same time by the *Jacket* and by the *Torso* objects.

4.2.2 MPEG-7 Video Description Scheme Proposal

The *Video* DS proposal is more related to the shot representation developed in the framework of this PhD thesis. The proposed DS describes the structure of the sequence as well as its semantic content in terms of events. Features related to sequence properties such as motion

and camera activity are included in this DS. The *Video* DS also relies on two hierarchical structures represented by trees. The first one is devoted to the description of the video structure and is called the Sequence Tree, whereas the second one describes what is happening and is termed the Event Tree.

The Sequence Tree is the Table of Contents of the video sequence. Its major functionality is to define the structure of the sequence and to describe its visual properties. Each node of the tree represents a connected component in time (such as a video sequence or a shot). The Sequence Tree describes how each sequence can be divided in shorter sub-sequences. Therefore, the leaves of the tree are assumed to be shots (or at least part of shots).

The Event Tree is the Index of the video sequence. It defines, in a hierarchical fashion, a set of events and sub-events, characterizes them and relates them with sequences in the Sequence Tree. Each element of this tree points to the occurrences of the corresponding shots or sequences in the Sequence Tree. The tree concentrates all the descriptors related to the semantic of what is happening in the video.

A concrete example of how the Sequence and Event Trees might look like is the one shown in Figure 4.3. The sequence corresponds to a reduced version of *Jornal da noite*, a Portuguese news program included in the MPEG-7 content set (several key-frames of the sequence can be seen of Figure B.18 in the appendix B). The Sequence Tree describes the news program decomposing it into sequences and shots. In the example, the Sequence Tree is composed of 9 sequences, 4 sub-sequences and 34 shots where each shot is represented in the Figure 4.3 by a single key-frame. The Event Tree lists the semantic items of the sequence. For each item, some links to the Sequence Tree are added to the events. For instance, the semantic item *News Presenter* is linked to several shots of the Sequence Tree.

Shots in the Sequence Tree can be further described with several descriptions such as a time reference, a classification of the type of shot transitions used, a characterization of the camera activity, a description of the motion of the shot, etc. The visual information can be characterized, as a first example, by a set of key-frames. Each key-frame is a still image extracted from the shot. If the visual content remains stable during the shot, one or a few key-frames are sufficient to characterize the entire shot. In the presence of significant changes due to camera motion, an alternative and richer representation may be needed. This representation may consist of a mosaic plus a set of foreground regions or *key-regions*. In practice, key-regions are identified as not belonging to the background because they have a different motion or are not in the same depth plane. This information is gathered in the *Video* DS, which is decomposed into one *Background Mosaic* DS and several *Foreground Object* DSs.

Together with the *Background Mosaic* DS visual information, a set of warping parameters can be included in the representation. These parameters are computed in order to create the mosaic (the warping parameters are the parameters defining the geometrical transformation applied to each image in order to represent it within a common reference system). The

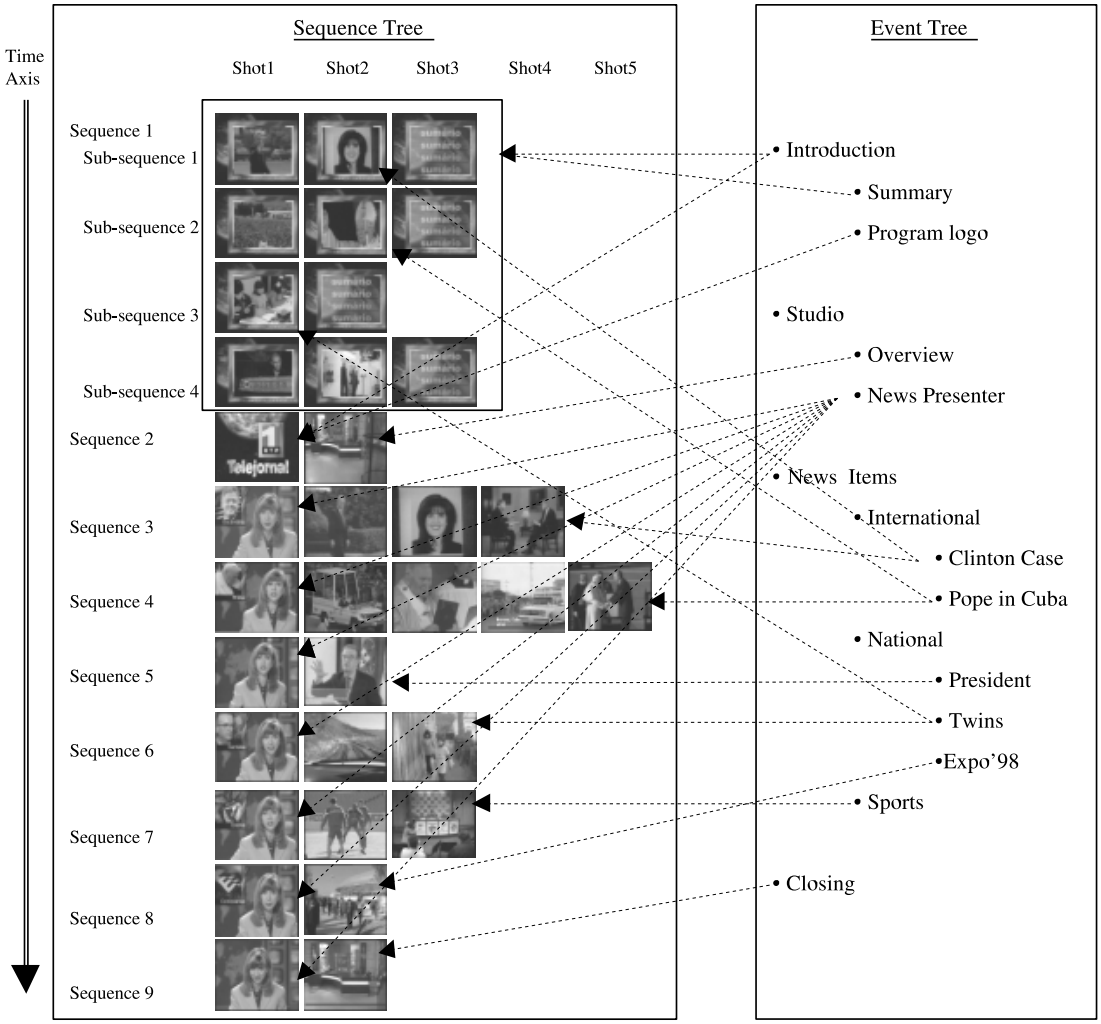


Figure 4.3: Example of Sequence and Event Trees (nodes of the tree are described with one key-frame).

Foreground Object DS is decomposed into several *Key-region* DSs. This decomposition is included here in order to be able to represent various visual instances of the same foreground object within the shot. Typical examples include key-regions representing a face or a human body. Note that both the mosaic and the key-regions may be described by the *Still Image* DS proposed in p186.

An illustration of the visual information description can be seen in Figure 4.4 for the outdoor sequence *ETRI_od_C* included in the MPEG-7 test set. Figure B.1 in the appendix B shows several key-frames of the sequence. The key-frame representation is not content based and may not be very accurate to represent very long shots or high motion sequences. However, the proposed indexing representation (in Figure 4.4) separates background from foreground information. It involves a mosaic plus three foreground objects. The first foreground object

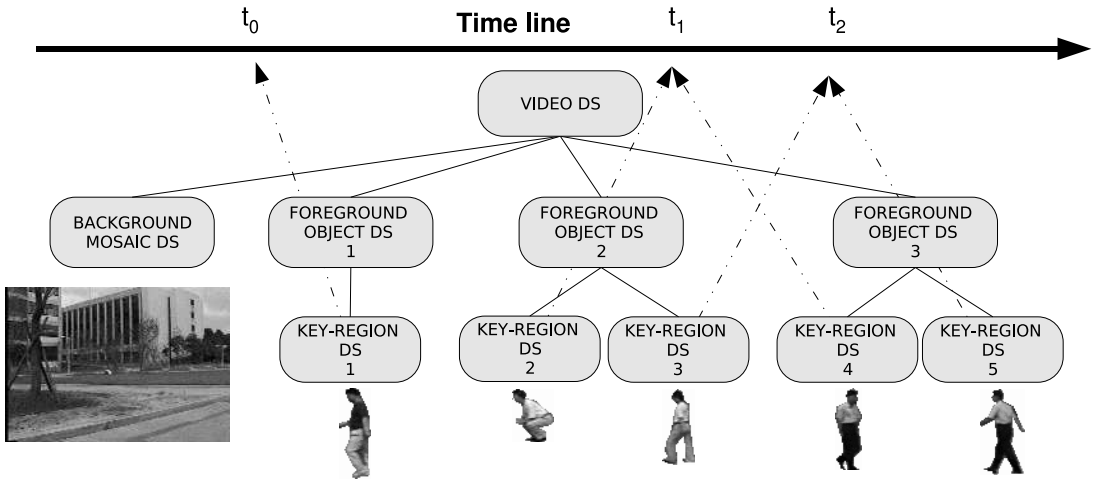


Figure 4.4: Example of a *Video* DS for the *ETRI_od_C* sequence.

is represented by key-region $k = 1$ and corresponds to a man walking. During the analysis and indexing process, it has been judged that one key-region is enough to characterize its appearance. The second foreground object is represented by key-regions $k = 2$ and $k = 3$ because its visual appearance exhibits significant variation during the shot. The man represented by key-region $k = 2$ is crouching at time t_1 and walking at time t_2. Finally, the last foreground object is represented by key-regions $k = 4$ and $k = 5$ and describes a man walking towards the camera at time t_1 and then walking away at time t_2.

4.2.3 Related MPEG-7 Description Schema

The previous proposals were the base to some MPEG-7 descriptions and description schema currently in the standard. In particular, the MPEG-7 *StillRegion* DS, the *Mosaic* DS and the *MovingRegion* DS share similarities with the proposed descriptions.

MPEG-7 StillRegion Description Scheme

The *StillRegion* DS describes an image or a 2D spatial region of an image or a video frame. In the case of digital images, a still region can correspond to a single pixel, an arbitrary group of pixels, or the full image. The still region does not need to be connected in space. The *SpatialMask* D optionally describes a 2D spatial mask that describes a set of non-connected sub-regions that form the still region. The *StillRegion* DS can also incorporate information of the time point of the still region in the video sequence.

The following DDL syntax defines the *StillRegion* DS (the Appendix A gives an overview of the MPEG-7 DDL):

```
<!-- ######################################## -->
```

```
<!--  Definition of StillRegion DS          -->
<!-- ####################################### -->
<complexType name="StillRegionType">
  <complexContent>
    <extension base="mpeg7:SegmentType">
      <sequence>
        <choice minOccurs="0">
          <element name="SpatialLocator" type="mpeg7:RegionLocatorType"/>
          <element name="SpatialMask" type="mpeg7:SpatialMaskType"/>
        </choice>
        <choice minOccurs="0">
          <element name="MediaTimePoint" type="mpeg7:mediaTimePointType"/>
          <element name="MediaRelTimePoint"
            type="mpeg7:MediaRelTimePointType"/>
          <element name="MediaRelIncrTimePoint"
            type="mpeg7:MediaRelIncrTimePointType"/>
        </choice>
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="VisualDescriptor" type="mpeg7:VisualDType"/>
          <element name="VisualDescriptionScheme" type="mpeg7:VisualDSType"/>
          <element name="GridLayoutDescriptors" type="mpeg7:GridLayoutType"/>
        </choice>
        <element name="MultipleView" type="mpeg7:MultipleViewType"
          minOccurs="0"/>
        <element name="SpatialDecomposition"
          type="mpeg7:StillRegionSpatialDecompositionType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The semantics of the *StillRegionType* are:

StillRegionType describes an image or a 2D spatial region of an image or a video frame. The *StillRegion* DS uses visual description tools (*VisualType* and *VisualDSType*) to describe the visual features of the still region. The spatial localization or composition of the still region is optionally described using a choice of *SpatialLocator* or *SpatialMask*.

SpatialLocator describes the spatial localization of the still region. The spatial locator describes a spatially connected sub-region that includes the spatial components of the still region. (optional)

SpatialMask describes a 2D spatial mask that defines the spatial composition of the still region. The still region is formed from the set of sub-regions described by the *SpatialMask*. If absent, the still region is composed of the single connected region defined by the *SpatialLocator*. (optional)

MediaTimePoint indicates the time point of a still region from a video sequence. (optional)

MediaRelTimePoint indicates the relative time point of a still region from a video sequence. (optional)

MediaRelIncrTimePoint indicates the relative incremental time point of a still region from a video sequence. (optional)

VisualDescriptor describes a visual feature of the still region using a visual descriptor. (optional)

VisualDescriptionScheme describes visual features of the still region using a visual DS. (optional)

GridLayoutDescriptors describes visual features of the sub-regions resulting from a grid decomposition of the still region. *GridLayoutDescriptors* descriptions only apply to rectangular still regions. (optional)

MultipleView describes visual features of a 3D object depicted in the 2D still region as seen from one or more viewing positions or angles. (optional)

SpatialDecomposition describes a spatial decomposition of the still region into one or more sub-segments. (optional)

MPEG-7 Mosaic Description Scheme

The *Mosaic* DS describes a mosaic or panoramic view of a video segment. The *Mosaic* DS extends the *StillRegion* DS and describes a panoramic view of the scene as a single image. The *Mosaic* DS also describes the motion model and the warping parameters of the mosaic and can be used to describe the mosaic used in this thesis (Chapter 6).

The following DDL syntax defines the *Mosaic* DS:

```
<!-- ################################## -->
<!--  Definition of Mosaic DS           -->
<!-- ################################## -->
<complexType name="MosaicType">
  <complexContent>
    <extension base="mpeg7:StillRegionType">
      <sequence>
        <element name="WarpingParam" minOccurs="0">
          <complexType>
            <sequence minOccurs="1" maxOccurs="unbounded">
              <choice>
                <element name="MediaTimePoint"
                  type="mpeg7:mediaTimePointType"/>
                <element name="MediaRelTimePoint"
                  type="mpeg7:MediaRelTimePointType"/>
                <element name="MediaRelIncrTimePoint"
                  type="mpeg7:MediaRelIncrTimePointType"/>
              </choice>
              <element name="MotionParam" type="mpeg7:floatVector"/>
            </sequence>
            <attribute name="modelType" use="required">
              <simpleType>
                <union>
                  <simpleType>
                    <restriction base="NMTOKEN">
                      <enumeration value="translational"/>
                      <enumeration value="rotationAndScaling"/>
                      <enumeration value="affine"/>
                      <enumeration value="perspective"/>
                      <enumeration value="quadratic"/>
                    </restriction>
                  </simpleType>
                  <simpleType>
                    <restriction base="mpeg7:termReferenceType"/>
                  </simpleType>
                </union>
              </simpleType>
            </attribute>
            <attribute name="sourceSequenceWidth" type="positiveInteger"
              use="required"/>
            <attribute name="sourceSequenceHeight" type="positiveInteger"
              use="required"/>
            <attribute name="xOffset" type="float" use="required"/>
```

```
            <attribute name="yOffset" type="float" use="required"/>
            <attribute name="xOrigin" type="float" use="required"/>
            <attribute name="yOrigin" type="float" use="required"/>
          </complexType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The semantics of the *MosaicType* are:

MosaicType describes a mosaic of a video segment.

WarpingParam describes the warping parameters and reference information used to construct the mosaic. (optional)

MediaTimePoint indicates the time point for the motion parameters.

MediaRelTimePoint indicates the relative time point for the motion parameters.

MediaRelIncrTimePoint indicates the relative incremental time point for the motion parameters.

MotionParam indicates the motion parameters as a vector. The parameters of the model are given by the *ParametricMotion* descriptor.

modelType indicates the type of the applied motion model. The types of models are translational, rotation and scaling, affine, perspective and quadratic model.

sourceSequenceWidth indicates the X dimension of the constructing video segment in pixel units.

sourceSequenceHeight indicates the Y dimension of the constructing video segment in pixel units.

xOffset indicates the X pixel coordinate of the top-left pixel of the first frame of the video segment in the mosaic.

yOffset indicates the Y pixel coordinate of the top-left pixel of the first frame of the video segment in the mosaic.

xOrigin indicates the X pixel coordinate of the origin of the spatial reference coordinate system in the first frame.

yOrigin indicates the Y pixel coordinate of the origin of the spatial reference coordinate system in the first frame.

MPEG-7 MovingRegion Description Scheme

The *MovingRegion* DS describes a video or a spatio-temporal region of a video. In the case of digital video, a moving region can correspond to an arbitrary group of pixels, or the full video. The moving region does not need to be connected in space or time. The *SpatioTemporalMask* D optionally describes a spatio-temporal mask giving a set of non-connected sub-regions in the case that the moving region is formed from non-connected sub-regions.

The DDL syntax that defines the *Mosaic* DS is:

```
<!-- ######################################### -->
<!--  Definition of MovingRegion DS (11.4.10)  -->
<!-- ######################################### -->

<!-- Definition of MovingRegion DS -->
<complexType name="MovingRegionType">
  <complexContent>
    <extension base="mpeg7:SegmentType">
      <sequence>
        <choice minOccurs="0">
          <element name="SpatioTemporalLocator"
            type="mpeg7:SpatioTemporalLocatorType"/>
          <element name="SpatioTemporalMask"
            type="mpeg7:SpatioTemporalMaskType"/>
        </choice>
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="VisualDescriptor" type="mpeg7:VisualDType"/>
          <element name="VisualDescriptionScheme" type="mpeg7:VisualDSType"/>
          <element name="VisualTimeSeriesDescriptor"
            type="mpeg7:VisualTimeSeriesType"/>
        </choice>
        <element name="MultipleView" type="mpeg7:MultipleViewType"
          minOccurs="0"/>
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="SpatialDecomposition"
            type="mpeg7:MovingRegionSpatialDecompositionType"/>
          <element name="TemporalDecomposition"
            type="mpeg7:MovingRegionTemporalDecompositionType"/>
          <element name="SpatioTemporalDecomposition"
            type="mpeg7:MovingRegionSpatioTemporalDecompositionType"/>
          <element name="MediaSourceDecomposition"
            type="mpeg7:MovingRegionMediaSourceDecompositionType"/>
        </choice>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The semantics of the *MovingRegionType* are:

MovingRegionType describes a video or a spatio-temporal region of a video. The *MovingRegion* DS uses visual description tools (*VisualDType* and *VisualDSType*) to describe the visual features of the moving region. The spatio-temporal localization or composition of the moving region is optionally described using a choice of *SpatioTemporalLocator* or *SpatioTemporalMask*.

SpatioTemporalLocator describes the spatio-temporal localization of the moving region. The spatio-temporal locator describes a spatio-temporally connected sub-region that includes the spatio-temporal components of the moving region. (optional)

SpatioTemporalMask describes a spatio-temporal mask that defines the spatio-temporal composition of the moving region. The moving region is formed from the set of sub-regions described by the *SpatioTemporalMask*. If absent, the moving region is composed of the single connected region defined by the *SpatioTemporalLocator*. (optional)

VisualDescriptor describes the visual features of the moving region using a visual descriptor. (optional)

VisualDescriptionScheme describes the visual features of the moving region using a visual description scheme.

VisualTimeSeriesDescriptor describes a temporal sequence of visual features in the moving region. The *VisualTimeSeriesDescriptor* applies only in the case of a connected moving region.

MultipleView describes visual features of a 3D moving physical object depicted in the moving region as seen from one or more viewing positions or angles (optional).

SpatialDecomposition describes the spatial decomposition of the moving region into one or more sub-segments. (optional)

TemporalDecomposition describes the temporal decomposition of the moving region into one or more sub-segments. (optional)

SpatioTemporalDecomposition describes the spatio-temporal decomposition of the moving region into one or more sub-segments. (optional)

MediaSourceDecomposition describes the media source decomposition of the moving region into one or more sub-segments. (optional)

4.3 Video Shot Representation

In this thesis, a method to automatically compute the *Video* DS of a shot has been developed. The representation is based on mosaics, hence, background information in the shot can be shared between this indexing representation and a compression one. Foreground regions are also detected and several key-regions, which represent foreground objects, are created in order to define the complete scene. Finally, warping information for the mosaic and motion information for the key-regions can be attached to the final representation including motion information in the final indexing representation.

This method of representing a video shot is very different from the classical approach of using one or more key-frames. Using key-frames to describe shots is very efficient for browsing as usually one or two key-frames are enough for the user to have a notion of what is happening in the video scene. On the other hand, using key-frames is not optimal with, for instance, high motion sequences, as a few key-frames may not be able to properly describe the entire shot. Also, sequences with very long shots are difficult to represent using only key-frames.

A more complex representation for video shots is based on the *Video* DS proposed in this thesis. One of the main advantages of this descriptor is that it is content based and, therefore, the information is structured into background and foreground regions that match objects in the video shot. The term “content based” means that the structure of the representation depends on the contents of the video shot itself. This representation is also suitable for high motion scenes as motion descriptors can be attached to both the mosaic and the foreground key-regions.

The disadvantages of this representation over key-frames are that, being a more complex representation, the creation algorithm for the mosaic and key-region is computationally more expensive. It needs to estimate the background information to create the mosaic which is

sometimes not possible as foreground regions may occlude background information for the entire shot duration. The extraction algorithm runs in a two-pass approach as the mosaic (with background information) is needed in order to extract key-regions (with foreground information). This means that the entire video shot is needed to extract the complete descriptor.

The following chapters of Part II of the thesis are devoted to the description of this sequence representation and the algorithm for extracting both the mosaic and the key-regions. The next chapter gives an overview of the elements of the descriptor and how this descriptor is obtained from a video sequence. Chapter 6 reviews the mosaic creation algorithm while Chapter 7 explains the details for the key-region extraction algorithm. Finally, Chapter 8 shows some results for a set of different video sequences and further describes the synergy between this representation and the use of mosaics for coding.

Chapter 5

Shot Representation Using a Mosaic and Key-regions

5.1 Mosaic and Key-regions Representation

The shot representation proposed in this thesis is based on a mosaic [86, 102] to represent the background information and on several key-regions [21, 22] to represent foreground objects. This indexing representation is based on the content of the video shot and exploits the synergy with compression representations by using a common mosaic to represent the background.

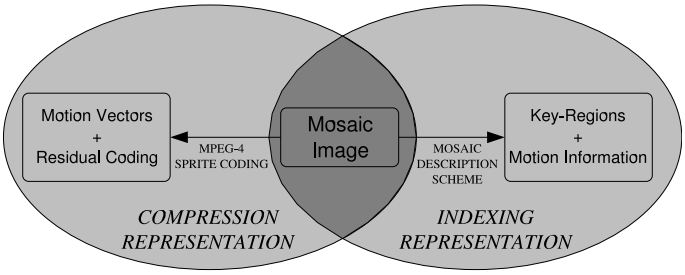


Figure 5.1: Compression and indexing representations may benefit each other by sharing a mosaic between them.

Figure 5.1 gives an overview of the scenario where mosaics are shared between compression and indexing representations. In this case, the mosaic may be used by the mosaic coding scheme (proposed by the standard MPEG-4) to create a compression representation and by the *Mosaic* DS to create a useful description for browsing. This allows applications to store efficiently both representations. It also facilitates the transmission as users that have previously browsed the content, therefore using the mosaic description, may not need to download the mosaic image again to retrieve the content. In that case, only the new data for compression, the motion information and the residual coding, is needed.

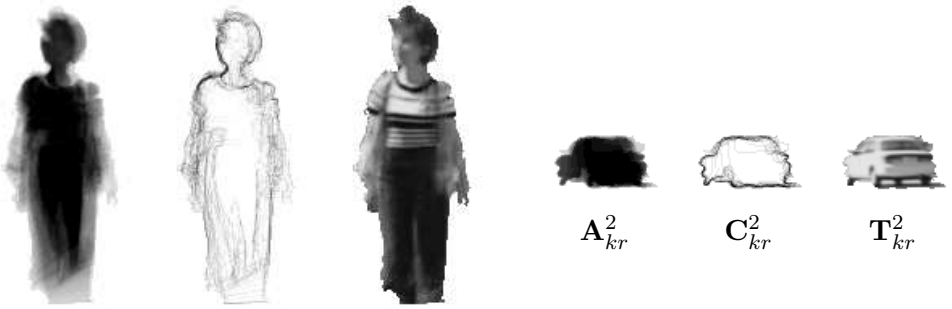
A typical example of the proposed indexing representation is shown on Figures 5.2 and 5.3. The Figures represent the shot descriptor created for 100 frames of the *NHKvideo7* sequence of the MPEG-7 test set with a resolution of 320 × 240 at 30 fps (several key-frames of the sequence are shown in Figure B.2). The mosaic shown in Figure 5.2 presents a panoramic view of the background components that are visible during the shot. This mosaic is created by merging all background information to the same spatial reference. The global motion of the scene is computed in order to select background regions that are merged into the mosaic (image regions are part of the background if they follow the global motion of the scene).

Figure 5.2: Mosaic representing the *NHKvideo7* sequence. Trajectories for both key-regions (in 5.3) are drawn as white arrows on the mosaic.

In the case of foreground regions, three images are used to represent them (as opposed to the background which is only represented by one image, the mosaic). This is needed because foreground regions may be composed of non-rigid objects and one image may not be sufficient to capture the representation of foreground regions. The key-region images are composed of an appearance image $\mathbf{A}^k_{kr}$, a contour image $\mathbf{C}^k_{kr}$ and a texture image $\mathbf{T}^k_{kr}$. The upper-index k stands for a key-region number as there might be more than one foreground region in a scene while the kr sub-index is used for clarity and it indicates that the previous images are part of the key-region k.

Figure 5.3 shows the two key-regions extracted for the *NHKvideo7* sequence. On the left the first key-region (a girl) is represented by the corresponding, from left to right, appearance, contour and texture images. On the right side, three images are also used to represent the other key-region in the shot (a car).

The appearance image $\mathbf{A}^k_{kr}$ represents the number of times a pixel has been estimated as belonging to key-region k. In Figure 5.3, darker pixels denote a high confidence that they belong to the key-region k while bright pixels denote that sometimes, but not often, they



a) Key-region 1 (girl) b) Key-region 2 (car)

Figure 5.3: Two key-regions, a) and b), for the *NHKvideo7* sequence. Each key-region is represented from left to right by an appearance image $\mathbf{A}_{kr}$, a contour image $\mathbf{C}_{kr}$ and a texture image $\mathbf{T}_{kr}$.

have been selected as part of the foreground of key-region k. This may happen, for instance, due to errors in the extraction process or because of small deformations in the foreground regions. The contour image $\mathbf{C}^k_{kr}$ is based on a spatial gradient and it stores the confidence of the key-regions contours over time. The contour image is used to modify the shape of the extracted foreground region on a frame by frame basis. This use of the contour image greatly improves the reliability of the extracted key-regions. The texture image $\mathbf{T}^k_{kr}$ represents the overall texture of the key-region in the shot.

Figure 5.4 shows another example of key-regions from the *ETRI_od_B* sequence (Figure B.3 shows several key-frames of the sequence). In this example, the key-region corresponds to a person walking in front of a static camera. The appearance, contour and texture images contain information of the activity followed by the key-region. In this case, higher body parts (body, chest) show no relative movement while lower part (legs) show a considerable amount of relative motion. The examples prove how the three images can efficiently describe a foreground region of the video scene.

In order to entirely describe the video shot, motion information can also be included in the final representation. The motion information used to merge all background information into the mosaic, the warping parameters, can be attached in the mosaic representation. Motion information for each key-region can also be attached. A simple and compact representation is to superpose the trajectories of the key-regions on the mosaic (in a similar manner to the synopsis mosaic introduced by Irani *et al.* in [27]). This facilitates the user to understand the motion performed for the various key-regions directly in the mosaic. Figure 5.2 shows the trajectories followed by the two key-region (both the girl and the car) as white arrows. The direction of the arrow represents the direction followed by the corresponding key-region over time.

$\mathbf{A}^1_{kr}$ $\mathbf{C}^1_{kr}$ $\mathbf{T}^1_{kr}$

Figure 5.4: Key-region for a person walking in the *ETRI_od_B* sequence. From left to right, appearance $\mathbf{A}^1_{kr}$, contour $\mathbf{C}^1_{kr}$ and texture $\mathbf{T}^1_{kr}$ images.

5.2 Overview of the Extraction Algorithm

The shot representation extraction algorithm involves three steps. The general scheme of the extraction algorithm is highlighted in Figure 5.5. The first step is the background mosaic computation (top blocks of Figure 5.5). The second step extracts the shape of each key-region at each time instant (middle blocks of Figure 5.5) and the last step combines the information obtained at all time instants and builds the key-region models (bottom blocks of Figure 5.5). The extraction process is done on a frame by frame basis. Since the mosaic is needed to extract key-region candidates, a two pass approach has been adopted. In the first pass the mosaic is computed (first row of Figure 5.5). In the second pass, the key-regions are extracted and modeled (second and third rows of Figure 5.5).

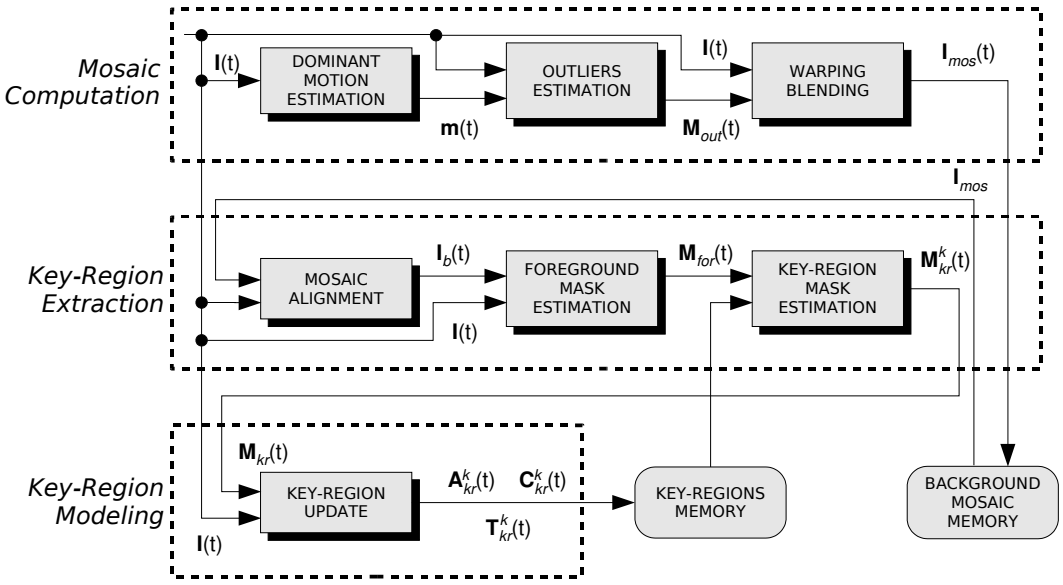


Figure 5.5: Overview of the shot representation extraction algorithm.

The mosaic creation algorithm used in this thesis is based on the MPEG-4 mosaic creation algorithm [75]. In order to improve the extraction of foreground regions in the mosaic, a method to compute mosaics by using morphological connected operators to identify outliers is employed. The background creation process itself follows a classical approach [11, 16]. The first step is to compute the dominant motion between successive input images, $\mathbf{I}(t)$ and $\mathbf{I}(t-1)$. This motion is used to warp all original frames $\mathbf{I}(t)$ in the same coordinate system. The warped images are blended to produce the mosaic, $\mathbf{I}_{mos}$. In order to be robust, the blending should only take into account pixels belonging to the background. As a result, before the warping and blending step, outliers that do not follow the dominant motion are identified. They are represented by an outliers mask $\mathbf{M}_{out}(t)$ (shown in the top row of Figure 5.5). Section 6.4 will show how morphological connected operators efficiently allow the identification of outliers.

Once the final mosaic has been computed, $\mathbf{I}_{mos}$, the second step extracts, for each key-region k at time t, a key-region mask $\mathbf{M}_{kr}^{k}(t)$. This extraction starts by the mosaic alignment. Its goal is to produce an estimation of the background information, $\mathbf{I}_b(t)$, at time t. Taking into account the dominant motion, the relevant part of the mosaic is un-warped to be compared to the current image $\mathbf{I}(t)$. A foreground mask, $\mathbf{M}_{for}(t)$, is computed by comparing the original image $\mathbf{I}(t)$ with the background estimation $\mathbf{I}_b(t)$. The foreground mask $\mathbf{M}_{for}(t)$ is an estimation of the key-regions at time t. However, this estimation is not very reliable because it is obtained on the basis of the observation of a single time instant. To improve the robustness of the analysis, the last step combines the contour information of the foreground masks extracted at each past time instant and selects the most reliable sections that have been observed to create the mask of the key-region, $\mathbf{M}_{kr}^{k}(t)$ (Section 7.3 explains the details of this approach).

Finally, the last step of the algorithm takes into account the key-region masks, $\mathbf{M}_{kr}^{k}(t)$, as well as the original image, $\mathbf{I}(t)$, to update the key-region models. The update step is performed by merging the key-region candidates (represented in the mask $\mathbf{M}_{kr}^{k}(t)$) into the corresponding key-region k. For non-rigid foreground objects, if the deformation of the object cannot be modeled by the extraction algorithm, they can be split into several key-regions. This last step is also responsible for grouping all extracted key-regions that correspond to the same foreground objects.

The extraction method proposed in this thesis combines both motion and texture information. Motion information, in particular the global motion of the scene, is needed to create the mosaic. The texture information (gray level) is used in the extraction of key-region mask candidates from the computed background information at time t, $\mathbf{I}_b(t)$. The next two chapters are devoted to explain in detail both the mosaic computation (Chapter 6) and the key-region extraction and modeling (Chapter 7). Finally, Chapter 8 comments some results of the final representation.

Chapter 6

Background Mosaic Creation

6.1 Definition

A mosaic is a static image based on a complete video sequence. In the mosaic, all frames of the video sequence are merged into the same spatial reference. In the process of creating a mosaic, foreground regions of the video scene are penalized, hence, the final mosaic is composed mainly of background parts of the scene.

Figure 6.1 shows an example of a mosaic for the *Stefan* sequence (some key-frames of *Stefan* are shown on Figure B.4). In the original sequence, the camera is following the tennis player who moves along the tennis court. The final mosaic is the tennis court where all moving regions (in this sequence, the tennis player) have been filtered out. However, the people on the stands and the referees have not been filtered out as they stand still through the duration of the video and, therefore, are considered as part of the background.

Figure 6.1: Mosaic of the first 240 frames of the *Stefan* sequence.

The mosaic creation algorithm is composed of three steps (denoted by the three top blocks of Figure 5.5). In the first step (detailed in Section 6.2), the dominant motion $\mathbf{m}(t)$ of the scene is computed using a motion estimation algorithm. Pixels of the image that do not follow the dominant motion are penalized. This penalty is represented by a weight map image (computed in a frame by frame basis). The weight map image, denoted by $\mathbf{W}_{im}(t)$, uses a dynamic range of $[0, 1]$, thus, pixels with values near 1 follow the dominant motion and are considered to be part of the background. Pixels with values near 0 are detected as foreground and are not used to update the mosaic.

In order to increase the robustness of the penalty estimation of foreground regions, a new processing block is employed in the second step (detailed in Section 6.4). Its goal is to estimate outliers using connected operators and to remove them (i.e. the weight map $\mathbf{W}_{im}(t)$ of these regions are set to 0), therefore, the final mosaic creation algorithm is more robust against foreground regions.

The third step is responsible for warping and blending the images into the mosaic (detailed in Section 6.5). In order to be robust against motion estimation errors when computing $\mathbf{m}(t)$ (between two consecutive images), a new motion estimation is performed (denoted as warping) between the current image and the mosaic. Finally, the blending is responsible for merging the image into the mosaic using the modified weight map image $\mathbf{W}_{im}(t)$ (with the detection of outliers) and the warping parameters.

6.2 Dominant Motion Estimation

The first step of the mosaic creation is to compute the global motion, usually refer as dominant motion, between two consecutive frames $\mathbf{I}(t)$ and $\mathbf{I}(t-1)$. It is assumed that most of the image (more than 50%) is occupied by background regions so the dominant motion estimation will be able to model the motion of the background.

Several motion models can be chosen to estimate the dominant motion between two frames. The most common are a translational motion with 2 parameters for each pixel [2], the affine motion with up to six parameters [67] or the perspective motion with a total of eight parameters [49, 95].

In general, the dominant motion of a video sequence is created by the motion of the camera recording the scene. Video cameras can create a multitude of motions such as translations, pans, tilts, etc. (Figure 6.2 shows the different motions associated to camera motions). The perspective model is able to characterize all possible camera motions and it is used in the framework of this work. The perspective motion model is suitable when the background scene can be approximated by a planar surface.

The perspective motion model is a motion model composed of eight different parameters. Assume that the vector $\mathbf{m}$ represents the perspective motion parameters ($\mathbf{m}(i)$ represents the

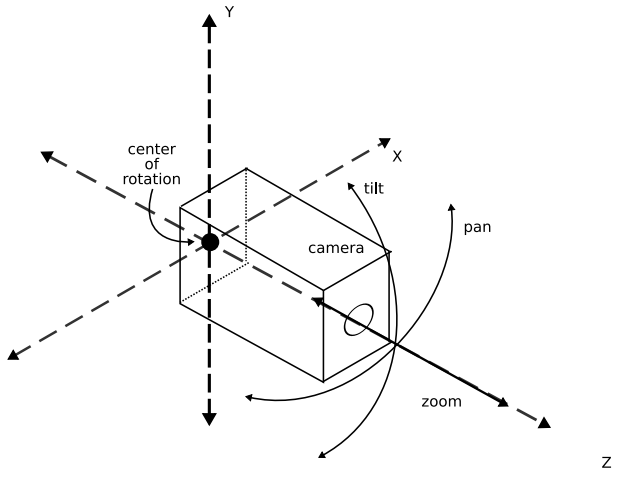


Figure 6.2: Possible camera motions.

i^{th} position starting from 0). If a pixel in the image $\mathbf{I}(t-1)$ is denoted by (x_1, y_1), the pixel (x_0, y_0) in the image $\mathbf{I}(t)$ can be computed (using the $\mathbf{m}$ perspective parameters) with the perspective equation:

$$x_0 = \frac{\mathbf{m}(0) \cdot x_1 + \mathbf{m}(1) \cdot y_1 + \mathbf{m}(2)}{\mathbf{m}(6) \cdot x_1 + \mathbf{m}(7) \cdot y_1 + 1} \qquad y_0 = \frac{\mathbf{m}(3) \cdot x_1 + \mathbf{m}(4) \cdot y_1 + \mathbf{m}(5)}{\mathbf{m}(6) \cdot x_1 + \mathbf{m}(7) \cdot y_1 + 1} \tag{6.1}$$

Sometimes it is more convenient to use the matrix form to define the perspective equation:

$$\begin{pmatrix} x'_0 \\ y'_0 \\ d'_0 \end{pmatrix} = \begin{pmatrix} \mathbf{m}(0) & \mathbf{m}(1) & \mathbf{m}(2) \\ \mathbf{m}(3) & \mathbf{m}(4) & \mathbf{m}(5) \\ \mathbf{m}(6) & \mathbf{m}(7) & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \\ d_1 \end{pmatrix} \tag{6.2}$$

Or in a shorter notation:

$$\mathbf{p}'_0 = \mathbf{M} \cdot \mathbf{p}_1 \tag{6.3}$$

where the matrix $\mathbf{M}$ characterizes the 8 perspective motion parameters, the vector $\mathbf{p}_1$ represents the homogeneous coordinates [19] of the pixel (x_1, y_1). The parameter d_1 is used to keep the matrix form and it is equal to 1. The vector $\mathbf{p}'_0$ denotes the new homogeneous coordinates of the pixel (if moved following the perspective motion). d'_0 is the denominator of (6.1), therefore, following the homogeneous coordinate system, the real coordinates of point (x_0, y_0) are $x_0 = x'_0/d'_0$ and $y_0 = y'_0/d'_0$.

In order to estimate the eight $\mathbf{m}$ perspective parameters that model the motion between two images $\mathbf{I}(t)$ and $\mathbf{I}(t-1)$ several techniques may be used. In general, most of the techniques minimize the prediction error by a differential or matching technique [2, 39, 105]. In our case, a Levenberg-Marquardt algorithm has been used to estimate the perspective motion between

two images. This method has been proposed for the dominant motion estimation in MPEG-4 [16, 49]. The Levenberg-Marquardt method is based on a multi-resolution gradient descent to minimize the prediction error. The algorithm assumes that the error function has a unique minimum and therefore a minimization of the parameters in the direction of the maximal descend of the gradient moves towards the minimum of the error function (Figure 6.3).

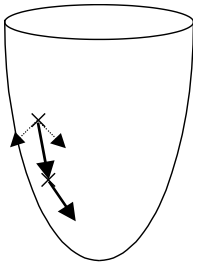


Figure 6.3: If the prediction error function (in this example represented as a parabola) only presents one minimum, the maximal descent of the gradient ensures a fast minimization of the error.

The Levenberg-Marquardt algorithm uses a multi-resolution approach. Figure 6.4 represents the pyramid of multi-resolution images with, for instance, three levels. The pyramid is always built from bottom to top. In order to build the next level the previous level image is filtered by a low pass filter and decimated by 2. A 3-tap separable filter with coefficients [1/4 1/2 1/4] has been selected as the low pass filter.

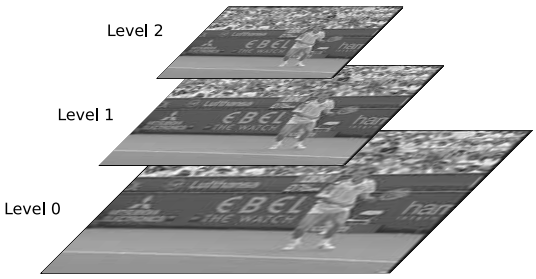


Figure 6.4: Pyramid of multi-resolution images. Images on top of the pyramid are smaller and with less resolution. In this case, the bottom image is the original image. The second level is decimated by 2 and the top level by 4.

The gradient descend is applied first at the top image of the pyramid (with less resolution) and iterates until a certain criterion of convergence is satisfied. The perspective motion parameters resulting from this first iteration are sent to the next lower level of the pyramid and the algorithm iterates again. Finally, the last parameters are projected into the final level of the pyramid (bottom) and the last iterations obtain the final motion.

The iteration at each level is performed by minimizing the prediction error in the direction of the maximal gradient descent. The error function designed to minimize the prediction error

is composed of the sum of squared differences (SSD) between the current frame $\mathbf{I}(t)$ and the motion compensated previous frame, denoted by $\hat{\mathbf{I}}(t-1)$.

$$e(t) = \sum_{\forall(x,y)} \mathbf{E}^2(x,y,t) = \sum_{\forall(x,y)} \Big(\hat{\mathbf{I}}(x,y,t-1) - \mathbf{I}(x,y,t)\Big)^2 \tag{6.4}$$

where (x, y) denotes the spatial coordinates of a pixel in the current frame and the summation is performed on all pixels of the current image $\mathbf{I}(t)$. In order to simplify the notation the dependency on time is frequently omitted and therefore the error equation results:

$$e = \sum_{\forall(x,y)} \mathbf{E}^2(x,y) = \sum_{\forall(x,y)} \Big(\hat{\mathbf{I}}(x,y) - \mathbf{I}(x,y)\Big)^2 \tag{6.5}$$

The motion parameters $\mathbf{m}$ are computed by minimizing e. Since the dependency of e on $\mathbf{m}$ is non-linear, an iterative process is used. In this iteration, the motion parameters are modified by a small amount $\mathbf{H}^{-1} \cdot \mathbf{b}$ in the direction of the maximal gradient descent. For each iteration k of the process, the perspective motion parameters are updated as follows:

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \mathbf{H}^{-1} \cdot \mathbf{b} \tag{6.6}$$

$\mathbf{H}$ is proportional to the Hessian matrix of $\mathbf{E}$ and is usually referred to as the curvature matrix [77]. $\mathbf{b}$ is an 8-element vector which corresponds to the gradient of $\mathbf{E}^2$ with respect to $\mathbf{m}$. In the computation of $\mathbf{H}$, the second order derivatives can be approximated from the first order derivatives [77], hence, each element (x, y) of the matrix $\mathbf{H}$ can be found as:

$$\mathbf{H}(x,y) = \frac{1}{2} \sum_{\forall(c,r)} \frac{\partial^2 \mathbf{E}^2(x,y)}{\partial \mathbf{m}(r) \partial \mathbf{m}(c)} \simeq \sum_{\forall(x,y)} \frac{\partial \mathbf{E}(x,y)}{\partial \mathbf{m}(r)} \frac{\partial \mathbf{E}(x,y)}{\partial \mathbf{m}(c)} \tag{6.7}$$

Each element i of the vector $\mathbf{b}$ is computed as:

$$\mathbf{b}(i) = -\frac{1}{2} \sum_{\forall(x,y)} \frac{\partial \mathbf{E}^2(x,y)}{\partial \mathbf{m}(i)} = -\sum_{\forall(x,y)} \mathbf{E}(x,y) \frac{\partial \mathbf{E}(x,y)}{\partial \mathbf{m}(i)} \tag{6.8}$$

The iteration is performed at each level of the multi-resolution pyramid and it is stopped when N_{max} iterations are performed or if the update term is small. Empirically $N_{max} = 32$ and a correct threshold for the update term is $\epsilon_1 = 0.1$ for the translational parameters (m_0 and m_1) and $\epsilon_2 = 0.001$ for the rest of the parameters [16]. The parameters found at each level are translated to the level below until the final parameters are computed. The translation onto a higher level of resolution only implies multiplying by a factor of 2 the perspective motion parameters $\mathbf{m}(0)$ and $\mathbf{m}(1)$ and dividing $\mathbf{m}(6)$ and $\mathbf{m}(7)$ by 2.

One of the main disadvantages of the Levenberg-Marquardt algorithm is its dependency on the initial conditions. If they are not accurate enough, the iteration converges easily to a

non-optimal solution. These errors can be minimized by using a good initialization. In the case of the mosaic creation, for each image $\mathbf{I}(t)$ the previous perspective motion parameters (computed for image $\mathbf{I}(t-1)$) are used as initialization. This assumes that the motion of the sequence is smooth and the motion between two pair of frames of the sequence is similar to that of the previous pair, which most of times is accomplished.

In the case of the first image, where no previous motion exists, an *N-step matching algorithm* also based on a multi-resolution pyramid of images is used [48]. For the initialization, a translation model is estimated using a three level multi-resolution pyramid. A search range of ± 4,± 2 and ± 1 pixels in the first, second and third level respectively is used at each pyramid level for the translational estimation.

Another disadvantage of the Levenberg-Marquardt minimization algorithm is the influence of the local object motion in the scene (outliers). Outliers do not follow the dominant motion and may distort the estimation. Section 6.4 will discuss a robust technique to eliminate the influence of outliers in the mosaic creation but it is also good to minimize the influence in the motion estimation step. In order to do that, a truncated quadratic [62] error function can be used instead of Eq. (6.5). This comes from considering M-estimators that minimize a function $\rho(\mathbf{E})$ of the form:

$$min \left(\sum_{\forall(x,y)} \rho\left(\mathbf{E}(x,y)\right) \right) \tag{6.9}$$

where $\rho(\mathbf{E})$ is a symmetric positive-definite function with a unique minimum at the pixels were $\mathbf{E} = 0$. In the original formulation of Eq. (6.5) $\rho(\mathbf{E})$ is a quadratic error function $\mathbf{E}^2$. However, a quadratic function gives large weights to large errors and therefore it is useful to define a new $\rho(\mathbf{E})$ function using a truncated quadratic error. For each pixel in the residual error image $\mathbf{E}$, the $\rho(\mathbf{E})$ is defined as:

$$\rho\left(\mathbf{E}\right) = \begin{cases} \mathbf{E}^2 & |\mathbf{E}| \leq h \\ 0 & |\mathbf{E}| > h \end{cases} \tag{6.10}$$

The threshold h is obtained by using the histogram of the absolute value of the error $|\mathbf{E}|$. In particular, h is chosen so as to exclude the pixels resulting in the top 10% of the histogram of $|\mathbf{E}|$. This threshold is updated at every level of the multi-resolution pyramid (only once at the beginning of the iterations at each pyramid level).

To summarize, the steps of the minimization algorithm to estimate the dominant motion of the scene between two consecutive images $\mathbf{I}(t)$ and $\mathbf{I}(t-1)$ are:

1. Create the multi-resolution pyramid with current image $\mathbf{I}(t)$.
2. Initialize the motion parameters $\mathbf{m}(t)$ with the previous motion parameters $\mathbf{m}(t-1)$ or with a translational coarse estimation if $t = 0$.

3. For each level of the pyramid (from top to bottom) do:
   (a) Translate motion parameters from upper level.
   (b) Build histogram of $|\mathbf{E}|$.
   (c) Obtain threshold h for the truncated quadratic equation (6.10) to exclude 10% of the pixels in the previous histogram.
   (d) Iterate to refine motion parameters. For each iteration do:
      i. Compute matrix $\mathbf{H}$ and gradient vector $\mathbf{b}$.
      ii. Update parameters with Eq. (6.6) by computing $\mathbf{H}^{-1}$.
      iii. Repeat until a maximum number of iterations is reached or the update term is too small.

6.3 Weight Map Computation

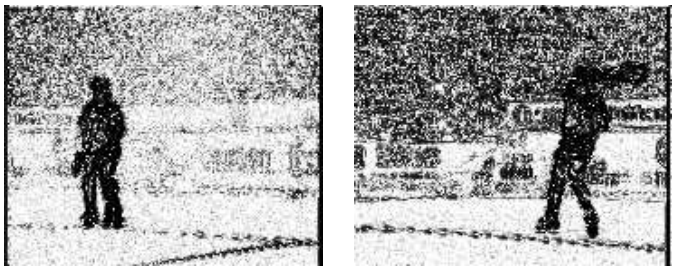
Once the motion estimation has obtained the dominant motion parameters at time t, $\mathbf{m}(t)$, it is important to use this information to discriminate between foreground and background regions in the scene. A first step is to perform the discrimination at pixel level. For this, a weight map of the image is constructed, denoted by $\mathbf{W}_{im}(t)$, at each time instant t.

Following the details in [62], the weight map $\mathbf{W}_{im}(t)$ is computed using a Lorentz function. For clarity, and as the weight map computation process is repeated at each time instant, the dependency on time is omitted. Therefore, each pixel in the weight map $\mathbf{W}_{im}(x, y)$ is related to the residual error $\mathbf{E}$ by:

$$\mathbf{W}_{im}(x, y) = \frac{2\sigma^2}{2\sigma^2 + \mathbf{E}^2(x, y)} \tag{6.11}$$

where σ is a constant fixed to 2 for all sequences. The residual error $\mathbf{E}$ is computed as in Eq. (6.5). Note that the estimated motion does not guarantee that pixel coordinates are integers, hence, a bilinear interpolation is used to obtain the gray value of compensated pixels. The weight map penalizes pixels that do not follow the dominant motion $\mathbf{m}(t)$ (i.e. are assumed to belong to foreground regions).

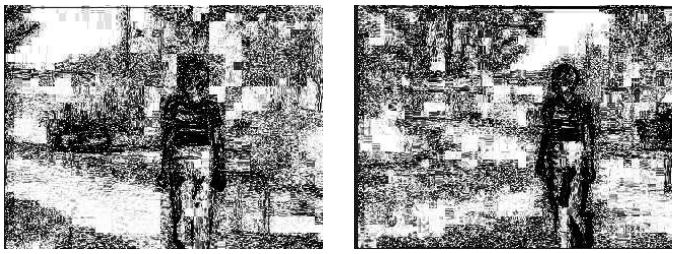
Figures 6.5 and 6.6 show some examples of the weight maps for the sequences *Stefan* and *NHKvideo7*. In Figure 6.5 two weight maps are shown corresponding to instants $t = 52$ and $t = 113$ on the original *Stefan* sequence. Pixels in the contours of moving regions are penalized (the weight map has a lower value in pixels that follow the background motion). In Figure 6.5, the motion estimation algorithm cannot estimate the motion of the people in the stands. Therefore, these people are weighted as belonging to the foreground. The tennis player, however, is well detected as its motion is not following the dominant motion of the scene.



a) b)

Figure 6.5: Two weight maps of the *Stefan* sequence corresponding to frame instants $t = 52$ and $t = 113$. To improve the clarity of the representation the dynamic range of the weight map is increased to $[0, 255]$ as $255 \cdot \mathbf{W}_{im}(t)$.

Figure 6.6 shows two weight maps for the *NHKvideo7* sequence at different time instants. This sequence is a more challenging one as the dominant motion is more difficult to estimate for two reasons. First, background objects are very close to the camera and are located in very different depth planes, and, therefore, the approximation for a planar surface is difficult. Second, foreground regions, such as the girl, represent a considerable portion of the scene. In this case the weight map does not clearly separate foreground and background regions even if, in general, higher weight values are given to pixels that belong to the background. The following section introduces connected operators to clearly separate outliers in the mosaic creation process.



a) b)

Figure 6.6: Two weight maps of the *NHKvideo7* sequence corresponding to frame instants $t = 16$ and $t = 60$ (several key-frames of the original sequence can be seen in Figure B.2). To improve the clarity of the representation the dynamic range of the weight map is increased to $[0, 255]$ as $255 \cdot \mathbf{W}_{im}(t)$.

6.4 Outliers Estimation

Using a weight map to penalize outliers is a very useful technique for the creation of mosaics [11, 86]. However, the weight map is created at pixel level and therefore it is neither

very reliable nor robust. In this case, morphological connected operators may better select foreground regions as they are region-based operators. The original pixel based weight maps can be further refined in the mosaic creation process by the use of these connected operators.

Gray level morphological connected operators are operators that act by merging elementary regions called *flat zones* [85]. They cannot create new contours nor modify the position of existing boundaries between regions and, therefore, have very good contour preservation properties. Several approaches can be used to create connected operators. In this thesis, one approach discussed in [84] is employed. The strategy consists in creating a region-based tree representation of the image and to apply a pruning strategy on the tree to simplify the image (in this case, without the outliers).

6.4.1 Max-tree Filtering Strategy

The tree representation is called Max-tree and is oriented towards signal maxima. Each node $\mathcal{N}_i$ in the tree represents a connected component of the space that is extracted by the following thresholding process: for a given threshold value T, consider the set of pixels X that have a gray level value larger than T and the set of pixels Y that have a gray level value equal to T:

$$X = \{x, \text{ such that } f(x) \geq T\} \quad \text{and} \quad Y = \{x, \text{ such that } f(x) = T\} \tag{6.12}$$

The nodes $\mathcal{N}_i$ represent the connected components of X such that $X \bigcap Y \neq \emptyset$. Figure 6.7 shows an example of a Max-tree construction. The original image is composed of seven flat zones identified by a letter $\{A, B, C, D, E, F, G\}$. The number following each letter defines the gray level value of the flat zone. In the example, the gray-level values range from 0 to 2. In the first step, the threshold T is fixed to the gray level value 0. The image is binarized: all pixels at level $T = 0$ (pixels of region A) are assigned to the root node of the tree $\mathcal{A}$. Furthermore, the pixels of gray-level value strictly higher than $T = 0$ form two connected components G and $BCDEF$. This creates the first tree (for gray-levels $[0, 1]$). In a second step, the threshold is increased by one: $T = 1$ and each node is processed as the original image. Consider, for instance, the node $\mathcal{BCDEF}$: all its pixels at level $T = 1$ remain unchanged and create the final node $\mathcal{BDF}$. However, pixels of gray-level values strictly higher than T (here C and E) create two different connected components and are moved to two child nodes $\mathcal{C}$ and $\mathcal{E}$. The complete tree construction is done by iterating this process for all nodes at level T and for all possible thresholds T (from 0 to the highest gray-level value). Note that in this procedure, some nodes may become empty. Therefore, at the end of the tree construction, the empty nodes are removed since they do not contribute any relevant information to the tree structure.

The filtering strategy consists in pruning the tree and in reconstructing the image from the resulting pruned tree. The simplification is governed by a criterion which may involve

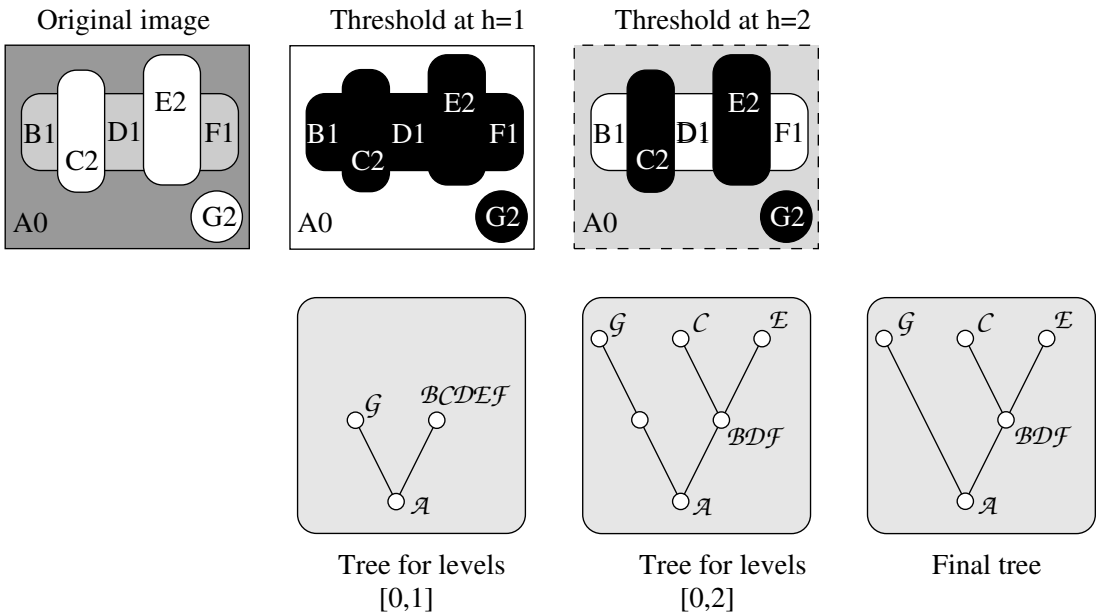


Figure 6.7: Max-tree creation example.

simple notions such as size, contrast or more complex ones such as texture, motion or even semantic criteria. Here, the detection of outliers is based on a motion criterion.

For each input frame $\mathbf{I}(t)$, the corresponding max-tree is created. A recursive version of the mean displaced frame difference is computed for all nodes of the trees using the dominant mosaic motion $\mathbf{m}(t)$ [84]. If N_k denotes all pixels in node $\mathcal{N}_k$, the motion criterion for each node $\mathcal{N}_k$ can be expressed as:

$$D_{\mathbf{I}(t)}^{\mathbf{I}(t-1)}(\mathcal{N}_k) = \frac{-\sum_{(x,y)\in N_k}\left|\hat{\mathbf{I}}(x,y,t-1) - \mathbf{I}(x,y,t)\right|}{\sqrt{G(\mathcal{N}_k)}} \tag{6.13}$$

where $\mathbf{I}(x,y,t)$ represents a pixel in the current image at time t and $\hat{\mathbf{I}}(x,y,t-1)$ represents the estimated pixel in the previous frame (at time $t-1$) using the dominant motion perspective parameters $\mathbf{m}(t)$. The function $\mathcal{G}(\mathcal{N}_k)$ is a measure of the gradient for pixels in region $\mathcal{N}_k$ and it is computed using a 4-connected gradient measure:

$$\begin{aligned}\mathcal{G}(\mathcal{N}_k) \quad = \tfrac{1}{4}\sum_{(x_1,y_1)\in N_k} \quad & \Big(|\mathbf{I}(x,y) - \mathbf{I}(x-1,y)| + |\mathbf{I}(x,y) - \mathbf{I}(x+1,y)| + \\ & + |\mathbf{I}(x,y) - \mathbf{I}(x,y-1)| + |\mathbf{I}(x,y) - \mathbf{I}(x,y+1)| \Big)\end{aligned} \tag{6.14}$$

As computing this motion criterion from a local motion between two frames may not be very robust, the final motion criterion for each node $\mathcal{N}_k$ is computed using a recursive measure. A memory factor is included by weighting two motion measures. One between the

current and previous frame $D_{\mathbf{I}(t)}^{\mathbf{I}(t-1)}$ and another between the current and the previous filtered frame $D_{\mathbf{I}(t)}^{\Psi(\mathbf{I}(t-1))}$ (where $\Psi(\cdot)$ is the connected operator defined in the following section).

$$\psi(\mathcal{N}_k) = \alpha \cdot D_{\mathbf{I}(t)}^{\mathbf{I}(t-1)}(\mathcal{N}_k) + (1-\alpha) \cdot D_{\mathbf{I}(t)}^{\Psi(\mathbf{I}(t-1))}(\mathcal{N}_k) \tag{6.15}$$

where α is a recursivity parameter between $[0, 1]$. If $\alpha \simeq 1$ the motion criterion does not have any memory of previous filtered results. If $\alpha \simeq 0$ then previous decisions have a great influence on the filtering. For the mosaic creation algorithm, the memory parameter has been fixed to $\alpha = 0.98$.

6.4.2 Filtering

The filtering process Ψ consists of removing nodes $\mathcal{N}_k$ of the tree that do not follow the given motion. A threshold λ is used to select nodes that should be removed, hence, if the motion descriptor for a node $\psi(\mathcal{N}_k)$ is larger than λ then the node must be preserved.

Unfortunately, the removal criterion Ψ is not increasing which means that there is no constraint stating that if a node has to be removed, its children have also to be removed (see, for instance, the example of Figure 6.8). As a result, the comparison between the motion criterion value and the threshold does not necessary define a pruning strategy. Several solutions may be used to deal with this issue. In particular, one can choose between different decision methods:

Direct A node is preserved if $\psi(\mathcal{N}_k) > \lambda$.

Min A node $\mathcal{N}_k$ is preserved only if all its parent nodes are also preserved (the motion descriptor of all parent nodes is greater than λ).

Max A node $\mathcal{N}_k$ is removed if all its descendant nodes are also removed.

Dynamic Programming A viterbi algorithm [101] is used in order to define a pruning strategy by changing the minimum number of preserve / remove decisions.

In general, the dynamic programming decision has proved to be more robust against local motion estimation errors [84] and it is the one chosen in the mosaic creation algorithm.

6.4.3 Outliers Removal

The motion filtering with morphological connected operators reviewed in the previous section is used to remove motion outliers. The filtering operator is applied to each image of the sequence. As the operation is based on a Max-Tree only bright outliers are removed using the motion criterion ψ. In order to remove bright and dark outliers a dual operator must be

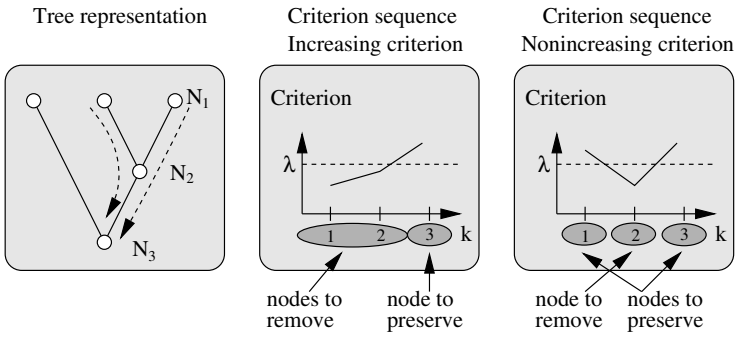


Figure 6.8: Max-tree non-increasing criteria example. The two graphs in the right represent the criterion values (vertical axes) for the two different tree paths in the left figure (node number in horizontal axes).

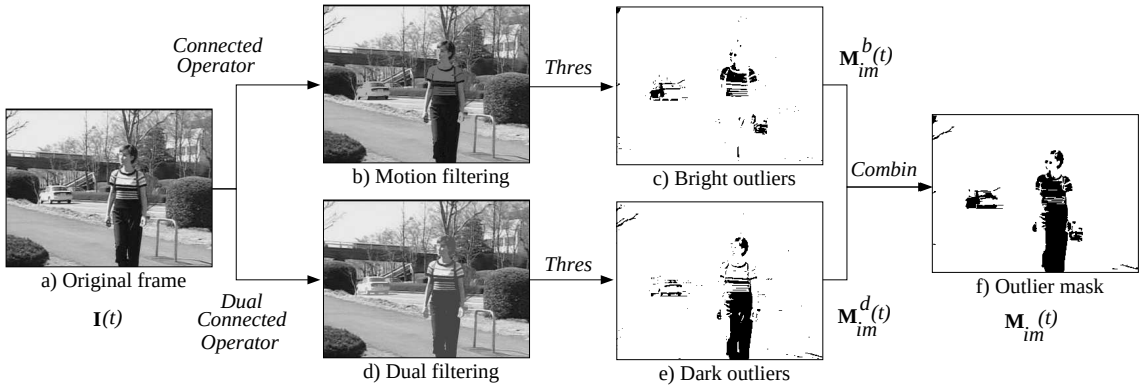


Figure 6.9: Estimation of outliers with connected operators for the *NHKvideo7* sequence.

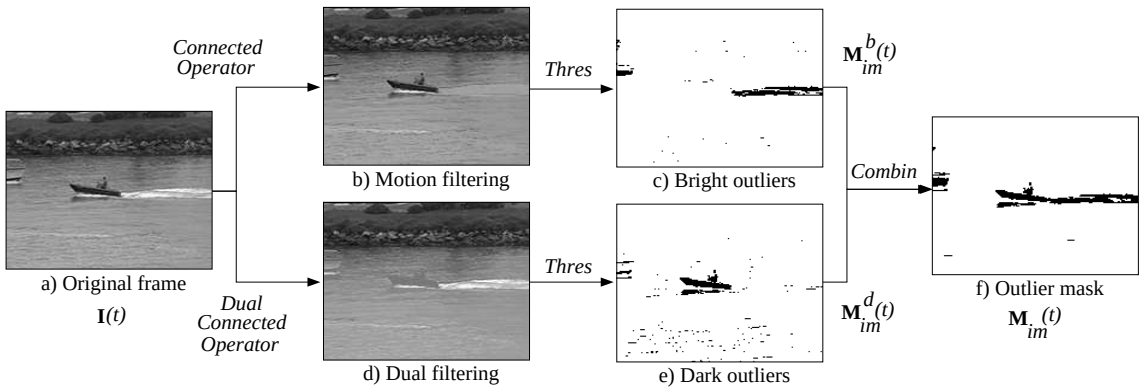


Figure 6.10: Estimation of outliers with connected operators for the *Coastguard* sequence.

used. The dual operator ψ^* is defined by: $\psi^*(f) = -\psi(-f)$ and has the same effects as ψ but on minima. Both operators are combined to obtain the final mask removing outliers.

Figure 6.9 shows an example of outliers estimation. Connected operators remove maxima of the image that do not follow the dominant motion $\mathbf{m}(t)$. The memory parameter is set to $\alpha = 0.98$ and the filtering parameter to $\lambda = 3$. Figure 6.9.a and 6.9.b show an original frame $\mathbf{I}(t)$ and the output of the connected filter $\Psi(\mathbf{I}(t))$. The filter has removed the bright components of the outliers (the girl and the car) and has preserved the background information. Comparison between the original and filtered frames gives the mask corresponding to bright outliers $\mathbf{M}_{im}^{b}(t)$:

$$\mathbf{M}_{im}^{b}(t) = \begin{cases} 0 & |\mathbf{I}(t) - \Psi\left(\mathbf{I}(t)\right)| = 0 \\ 255 & |\mathbf{I}(t) - \Psi(\mathbf{I}(t))| > 0 \end{cases} \tag{6.16}$$

A single gray value in Eq. (6.16) is used to consider pixels as outliers, thus, only the λ parameter of the filtering strategy controls the amount of motion that is filtered in the final mask.

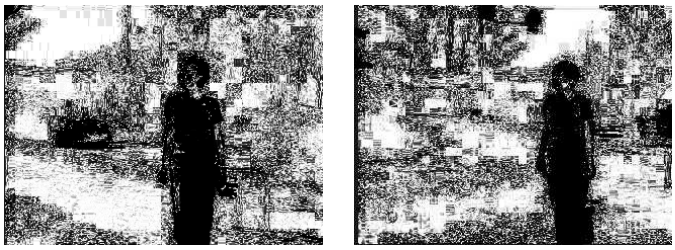
The estimation of dark outliers is done using the dual connected operator. Figure 6.9.d and 6.9.e show the filtered output and the mask corresponding to dark outliers. The final outlier mask $\mathbf{M}_{im}(t)$ is shown in Figure 6.9.f and is computed by adding (with a binary union) the bright and dark masks:

$$\mathbf{M}_{im}(t) = \mathbf{M}_{im}^{b}(t) \ \cup \ \mathbf{M}_{im}^{d}(t) \tag{6.17}$$

Figure 6.10 shows another example with the *Coastguard* sequence at QCIF resolution (a few key-frames of this sequence are represented on Figure B.5). The same filtering process to create an outlier mask is highlighted. First the filtering operator Ψ selects bright outliers (Figure 6.10.b and Figure 6.10.c). The dual operator highlights dark regions (Figure 6.10.d and Figure 6.10.e). The final mask in Figure 6.10.f successfully selects the moving regions that do not follow the dominant motion (the boat in this example). It also filters the moving water on the boat's trail. The motion of the water is too different from the dominant motion of the sequence (a translational motion to the left) which makes the connected operator to filter the corresponding boat's trail of water.

Finally, the outliers estimation step updates the weight maps calculated in the previous step. This is done by setting all pixels that have been detected as outliers in the outlier mask $\mathbf{M}_{im}(t)$ to a value of 0 in the weight map image $\mathbf{W}_{im}(t)$. Therefore, outlier pixels will not contribute in the creation of the mosaic. Figure 6.11 shows the updated weight map for the *NHKvideo7* test sequence at the same time instants as in Figure 6.6. As it can be seen, the outliers corresponding to the girl and the car have been better detected in the updated weight map by using the connected operators.

The Section 6.6 will discuss different mosaic results comparing the use (or not) of connected operators to remove foreground motion outliers (Figures 6.16, 6.18 and 6.17).



a) b)

Figure 6.11: Two updated weight maps of the *NHKvideo7* sequence corresponding to frame instants $t = 16$ and $t = 60$ using the outlier mask $\mathbf{M}_{im}(t)$. To improve the clarity of the representation the dynamic range of the updated weight map is increased to $[0, 255]$ as $255 \cdot \mathbf{W}_{im}(t)$.

6.5 Warping and Blending

After both the dominant motion $\mathbf{m}(t)$ and the outliers estimation are performed, the next step is to warp and blend the estimated frames into a single spatial reference which is usually the first frame of the sequence, $\mathbf{I}(0)$. The warping step consists in accumulating the estimated dominant motion into a first reference. The blending step merges the new image at time t into the current mosaic $\mathbf{I}_{mos}$ (referred to time $t = 0$).

6.5.1 Warping

The goal of this step is to accumulate the dominant motion parameters into a single common reference. Figure 6.12 shows an example of the accumulation of two perspective motions $\mathbf{m}(t)$ and $\mathbf{m}(t-1)$ into the motion parameters $\mathbf{a}$. The accumulation allows referring a pixel (x_2, y_2) in image $\mathbf{I}(t-2)$ directly to pixel position (x_0, y_0) in image $\mathbf{I}(t)$.

This accumulation can be easily done using the perspective parameters. If $\mathbf{m}(t-1)$ are the estimated perspective motion parameters between $\mathbf{I}(t-2)$ and $\mathbf{I}(t-1)$ and $\mathbf{m}(t)$ between images $\mathbf{I}(t-1)$ and $\mathbf{I}(t)$. The accumulated motion $\mathbf{a}$, which refers the last image with the first one, can be computed from the two local motions. If $\mathbf{p}_2$ denotes the homogeneous coordinates of a pixel in $\mathbf{I}(t-2)$, $\mathbf{p}_1$ the homogeneous coordinates of the corresponding pixel in $\mathbf{I}(t-1)$ and $\mathbf{p}_0$ the homogeneous coordinates of corresponding pixel in $\mathbf{I}(t)$ then, using the matrix form of the perspective motion parameters (as in Eq. (6.2)), the accumulated parameters $\mathbf{a}(t)$ can be computed as:

$$\begin{aligned} \mathbf{p}_1 &= \mathbf{M}(t-1) \cdot \mathbf{p}_2 \\ \mathbf{p}_0 &= \mathbf{M}(t) \cdot \mathbf{p}_1 \\ \mathbf{p}_0 &= \mathbf{M}(t) \cdot \mathbf{M}(t-1) \cdot \mathbf{p}_2 = \mathbf{A}(t) \cdot \mathbf{p}_2 \end{aligned} \tag{6.18}$$

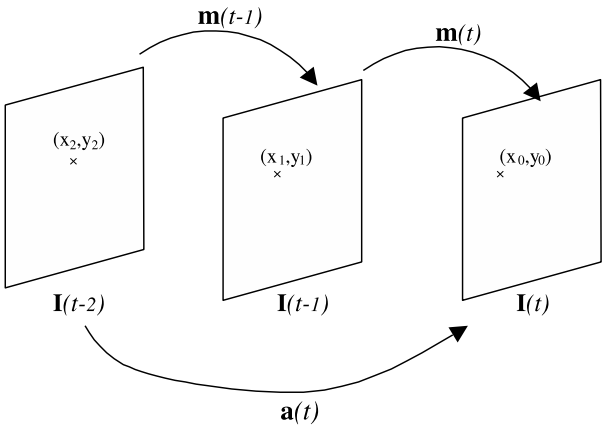


Figure 6.12: The warping step consist in referring all images to a single common reference, usually the first one.

In practice, the mosaic warping step does not only accumulate the local dominant motion parameters $\mathbf{m}$ as errors in one estimation may be propagated over the entire sequence. In order to minimize this propagation of errors, a new motion estimation is performed between the current image and the mosaic, hence, prediction errors can be corrected. The motion estimation algorithm chosen for this step is different from the Levenberg-Marquardt minimization algorithm used for the local estimation (reviewed in Section 6.2). In this case, a 4-vertices displacement algorithm (4VD) is used for the estimation between the image at time t and the mosaic.

The 4VD algorithm is a more precise approach than the Levenberg-Marquardt minimization although it can fall into local minima more easily. The 4VD algorithm starts from a prediction of the 4 corners of the current image $\mathbf{I}(t)$ into the current mosaic $\mathbf{I}_{mos}(t-1)$ (build with images from time 0 to $t-1$). The use of 4 vertices is due to the need of 4 points to define a perspective motion. Each one of the vertices is displaced one fraction of a pixel (depending on the selected resolution) and a similarity between the deformed image and the mosaic is measured. The algorithm iterates for all vertices until the lowest similarity value is found.

The similarity is measured using a weighted version of the quadratic error function $\mathbf{E}^2$:

$$\sum_{\forall(x,y)} \mathbf{W}_{im}(x,y) \cdot \mathbf{E}^2(x,y) \tag{6.19}$$

This time, the quadratic error function $\mathbf{E}^2$ is computed between the current mosaic $\mathbf{I}_{mos}(t-1)$ and the motion compensated current frame $\hat{\mathbf{I}}(t)$, instead of between current image and motion compensated previous frame as in Eq. (6.5). The weight map $\mathbf{W}_{im}$ is the one detailed in Section 6.3 and improved by the connected operators proposed in Section 6.4. At each step of the 4VD algorithm, one corner is moved to the position with lowest similarity

(Figure 6.13).

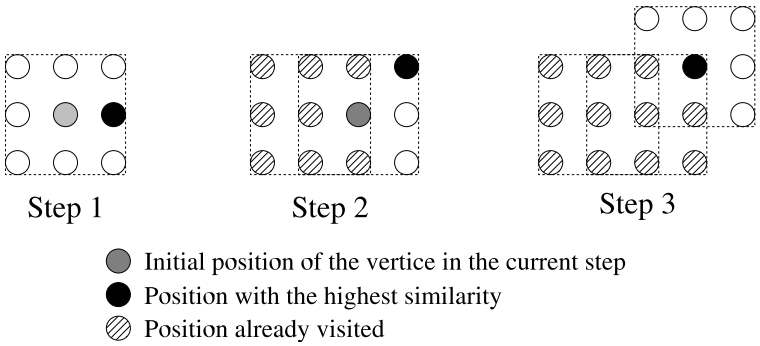


Figure 6.13: Vertice displacement for the 4VD algorithm (using 8 connectivity).

The 4VD algorithm is very precise but needs a very good initialization to avoid local minima. A good initialization is the accumulated local motion at time t, $\mathbf{a}(t)$. This prevents the algorithm to deviate from the accumulated motion but at the same time it gives an opportunity to correct errors in the local motion estimation step. The result of this algorithm is the 4 new vertices positions (x_{v_i}, y_{v_i}) resulting from the initial 4 vertices of the current image $\mathbf{I}(t)$. This 4 points can be easily converted again to 8 perspective parameters (denoted by $\mathbf{g}(t)$) resolving the 8 equations (2 for each coordinate) that relate the 4 vertices with the new positions (x_{v_i}, y_{v_i}). The 8 parameters $\mathbf{g}(t)$, called warping parameters, finally represent the estimated motion between the current frame $\mathbf{I}(t)$ and the first image of the sequence and are used to warp (motion compensate) the current frame to the same temporal reference as the mosaic.

6.5.2 Blending

Once the motion estimation is performed the current image can be blended into the mosaic. The blending step updates the previous mosaic $\mathbf{I}_{mos}(t-1)$ using the motion compensated current image $\hat{\mathbf{I}}(t)$ and the motion compensated weight map $\hat{\mathbf{W}}_{im}(t)$ (including the outliers estimation). Both images $\hat{\mathbf{I}}(t)$ and $\hat{\mathbf{W}}_{im}(t)$ are compensated version of $\mathbf{I}(t)$ and $\mathbf{W}_{im}(t)$ respectively using the warping parameters $\mathbf{g}(t)$ (which represent the motion between the original image at time t and $t=0$). Each coordinate (x, y) of the mosaic is updated as:

$$\mathbf{I}_{mos}(t) = \frac{\mathbf{W}_{mos}(t-1) \cdot \mathbf{I}_{mos}(t-1) + \hat{\mathbf{W}}_{im}(t) \cdot \hat{\mathbf{I}}(t)}{\mathbf{W}_{mos}(t-1) + \hat{\mathbf{W}}_{im}(t)} \tag{6.20}$$

where $\mathbf{W}_{mos}$ is a weight map for the mosaic where all compensated image weight maps $\hat{\mathbf{W}}_{im}$ are accumulated. Each pixel in the mosaic weight map is updated as follows:

$$\mathbf{W}_{mos}(t) = \mathbf{W}_{mos}(t-1) + \hat{\mathbf{W}}_{im}(t) \tag{6.21}$$

The mosaic weight map is taken into account to weight all frames that are used to build the mosaic. Intuitively, if 100 frames are used to build a mosaic, each image should contribute a fraction of 1/100 to the final mosaic. The mosaic weight map is used to create this weighting. At the same time, the current image weight map is used to discard pixels that have been identified as not belonging to the background. Figure 6.14 shows an example of the mosaic weight map for the *Stefan* sequence. In the example, the mosaic weight maps at time instants $t = 60$ and $t = 200$ are shown. It can be seen how new images are weighted low (dark values) as they enter in the mosaic. Foreground detected regions (the tennis player) are assigned weights minimizing their influence during the updating of the mosaic.

a) b)

Figure 6.14: Two mosaic weight maps for the *Stefan* sequence at times a) $t = 60$ and b) $t = 200$. To improve the clarity of the representation the dynamic range of the updated weight map is increased to $[0, 255]$ as $255 \cdot \mathbf{W}_{im}(t)$.

6.6 Final Mosaic

The previous sections have reviewed the technique used in this thesis to build mosaics. This technique is very similar to the one used in the MPEG-4 mosaic coding scheme where mosaics are used as static references for sequence coding. The contribution in our approach is to use morphological connected operators to filter out outliers that may influence the creation of the mosaic. The final algorithm to build mosaics can be summarized by the following steps. Each frame of the sequence is processed as follows:

1. Estimate the dominant motion estimation $\mathbf{m}(t)$ between two consecutive frames by a Levenberg-Marquardt gradient descend.
2. Create a weight map $\mathbf{W}_{im}(t)$ for the current frame by weighting pixels that do not follow the dominant motion.
3. Update the previous weight map by filtering out outliers using morphological connected operators.

4. Warp the input frame $\mathbf{I}(t)$ with the current mosaic $\mathbf{I}_{mos}(t-1)$ with the 4VD motion estimation to obtain the warping parameters $\mathbf{g}(t)$ preventing the propagation of errors.

5. Update the mosaic $\mathbf{I}_{mos}(t)$ and mosaic weight map $\mathbf{W}_{mos}(t)$ with the current frame by blending the warped image $\hat{\mathbf{I}}(t)$ with the mosaic.

Figure 6.15: Mosaic for the *Coastguard* sequence using a perspective model in the motion estimation and connected operators to remove outliers.

The dominant motion estimation step is important to globally follow the camera motion of the scene. For instance, in the *Coastguard* sequence, the Levenberg-Marquardt minimization is confused by the water as there is too much motion in water regions. In this case, the dominant motion estimation is not able to follow the camera motion and the final mosaic represented in Figure 6.15 does not exactly represent the background information of the scene.

However, if the perspective motion estimation is forced to be translational (only the m_2 and m_5 perspective parameters are allowed to change) then the final mosaic is able to better represent the real motion of the scene (Figure 6.17.b). In general, the perspective Levenberg-Marquardt estimation is a very good option to estimate the dominant motion for the mosaic creation algorithm but fails in video scenes with noisy regions (such as *Coastguard* due to the water) or with a high mixture of heterogeneous motions.

Figure 6.16 compares the classical solution with the one proposed using connected operators. The classical approach does not allow the elimination of outliers that occupy a significant portion of the image (as the girl). A dark shadow is clearly visible in the lower right part of the mosaic of Figure 6.16.a. Moreover, the partial elimination of outliers has a strong effect on the successive warping and blending steps: strong geometrical deformations appear on the lower right and upper left part of Figure 6.16.a.

Figure 6.17 compares the outliers estimation for the *Coastguard* sequence. Both mosaics are computed using the first 280 frames of the *Coastguard* sequence using (in Figure 6.17.b)

a) Without connected operators

b) Connected operators applied to remove outliers

Figure 6.16: Comparison of mosaic creation without a) and with b) connected operators for 100 frames of the *NHKvideo7* sequence.

and not using (Figure 6.17.a) connected operators to remove outliers. In this sequence, the use of connected operators is visible in the water where the connected operators are able to better remove the boat and the ship. In the mosaic of Figure 6.17.a, a darker area can be seen in the upper part of the water as foreground regions are not completely deleted from the mosaic.

The next example shows the use of connected operators in the *Stefan* sequence. In this sequence, the foreground region (the tennis player) has a very different motion from the background and therefore the detection of outliers is not as critical for the mosaic creation as in previous sequences. Figure 6.18 shows both the mosaic of the first 240 frames of the

a) Without connected operators

b) Connected operators applied to remove outliers

Figure 6.17: Comparison of mosaic creation without a) and with b) connected operators for 280 frames of the *Coastguard* sequence.

sequence. In this example, the improvement of detecting outliers is difficult to detect. The "ghost" player is slightly better removed and the white trail on the first referee is greater in mosaic Figure 6.18.a than Figure 6.18.b.

In general, when using connected operators to remove motion outliers, if the foreground motion is very different from the dominant motion (such as the *Stefan* sequence), the weight map image is able to successfully remove the foreground contribution in the mosaic. If the motion is not very different (such as the other sequence examples), morphological operators (based on connected regions) can further help improving the mosaic.

The solution used for the warping step prevents the propagation of motion estimation errors. Using a motion estimation algorithm such as the 4VD algorithm also greatly improves the precision of the perspective motion parameters. If the refined estimation is not used,

a) Without connected operators

b) Connected operators applied to remove outliers

Figure 6.18: Comparison of mosaic creation without a) and with b) connected operators for 240 frames of the *Stefan* sequence.

motion estimation errors are accumulated and the final mosaic has a very poor quality. Figure 6.19 shows the mosaic for the *NHKvideo7* sequence without the 4VD motion refinement. It shows how the mosaic is not as sharp as the one in Figure 6.16.b. The 4VD refinement is able to better follow the motion of the scene and the merging of frames with the mosaic is more accurate. Figure 6.19 clearly shows how the accumulation of errors in the motion estimation greatly degrades the final mosaic.

In this thesis, the final mosaic is not only used to improve coding efficiency (such as the MPEG-4 mosaic coding scheme). Here, a new functionality is investigated. The mosaic is exploited to extract foreground regions of the shot. These foreground regions are represented using several images called key-regions. The mosaic and key-regions can make a useful and compact overall description of the shot. The following chapter reviews the algorithm to extract foreground regions on a video sequence.

Figure 6.19: Mosaic for 100 frames of the *NHKvideo7* sequence without the 4VD motion refinement.

Chapter 7

Key-region Extraction and Modeling

The mosaic extracted in the previous chapter can be used not only to increase the coding efficiency of compression representations (as it will be seen on Section 8.2) but also to extract foreground regions of the scene. These foreground regions can be modeled in key-regions that are used to represent all foreground objects of the scene. The steps of the extraction algorithm are sketched on the second and third row of Figure 5.5. The first part (second row) is responsible for extracting, for each key-region k, a key-region mask $\mathbf{M}_{kr}^{k}(t)$ where the foreground region is segmented from the background at time t. The second part of the extraction algorithm (third row of Figure 5.5) involves the modeling of the key-region into three images and the grouping of key-regions that represent the same foreground object.

The foreground extraction starts by the mosaic alignment (detailed in Section 7.1). Its goal is to produce an estimation of the background information, $\mathbf{I}_b(t)$, at time t. Taking into account the dominant motion, the relevant part of the mosaic is un-warped to be compared to the current image $\mathbf{I}(t)$. A foreground mask, $\mathbf{M}_{for}(t)$, is computed by comparing the original image $\mathbf{I}(t)$ with the background estimation $\mathbf{I}_b(t)$ using a watershed algorithm [100] (detailed in Section 7.2). The foreground mask $\mathbf{M}_{for}(t)$ is an estimation of the key-regions at time t. However, this estimation is not very reliable because it is obtained on the basis of the observation of a single time instant. To improve the robustness of the analysis, an additional step combines the contour information of the foreground masks extracted at each past time instant and selects the most reliable sections that have been observed to create the mask of the key-region, $\mathbf{M}_{kr}^{k}(t)$. Section 7.3 will show how a watershed algorithm can also be used to combine a set of contours taking into account their reliability.

Finally, the last step of the algorithm (key-region update block of Figure 5.5) takes into account the key-region masks, $\mathbf{M}_{kr}^{k}(t)$, as well as the original image, $\mathbf{I}(t)$, to update the key-region models that are used to represent the scene. Once the entire video is processed and

all key-regions are extracted, key-regions that represent the same non-rigid foreground object are grouped together (detailed in Section 7.4).

7.1 Mosaic Alignment

The first step in the key-region extraction process is to create a background representation of the scene at a specific time instant t. In order to do that, the dominant motion of the scene (estimated on the mosaic creation process and detailed in Section 6.5) is used. The warping parameters $\mathbf{g}(t)$ are used to un-warp the final mosaic $\mathbf{I}_{mos}(t_{end})$ and to produce an estimation of the background information $\mathbf{I}_b(t)$ at time t which can be compared with the current frame $\mathbf{I}(t)$. The final mosaic at time $t = t_{end}$, computed using all frames of the sequence, is used since foreground regions are completely removed from the mosaic. However, using the final mosaic in the key-region extraction implies that a second pass over the entire sequence is necessary and may only be used in non-real time situations.

The warping parameters $\mathbf{g}(t)$, estimated to create the mosaic image, relate the motion from the image at time t to the mosaic image. In order to create the background image $\mathbf{I}_b(t)$, the inverse warping parameters are needed (to relate the mosaic with the image at time t). Using the matrix notation for the warping parameters (as in Eq.(6.2)) the pixels in the mosaic image can be related to the background image by:

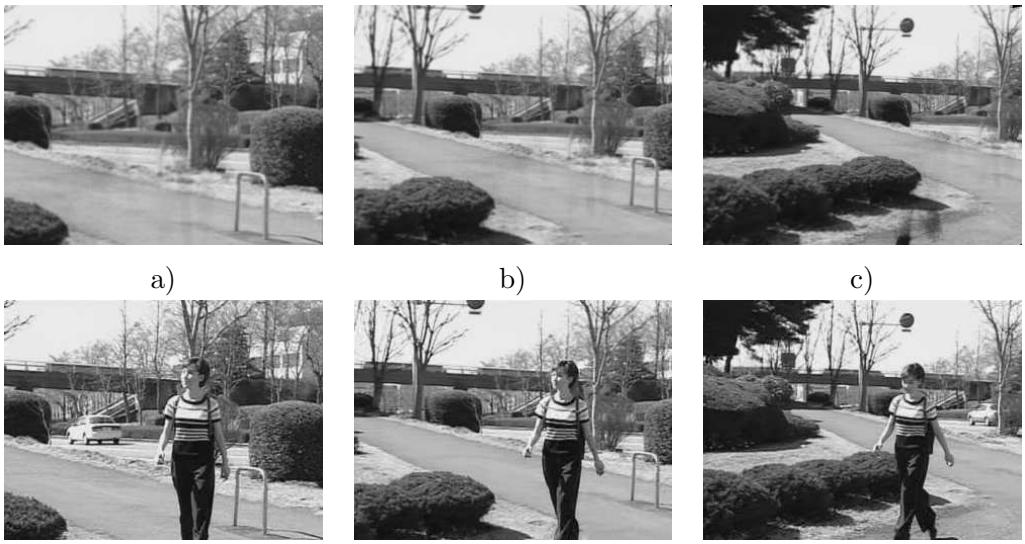
$$\mathbf{p}_b = \mathbf{G}^{-1}(t) \cdot \mathbf{p}_m \tag{7.1}$$

where $\mathbf{p}_m$ corresponds to the homogeneous coordinates of a pixel (x_m, y_m) in the mosaic image and $\mathbf{p}_b$ denotes the homogeneous coordinates of the pixel in the background image. $\mathbf{G}^{-1}(t)$ corresponds to the inverse matrix of $\mathbf{G}(t)$ (matrix form of warping parameters $\mathbf{g}(t)$).

Figure 7.1 shows an example of the background information extracted from the mosaic of the *NHKvideo7* sequence at different time instants. The three images in the top row represent the background information $\mathbf{I}_b(t)$ contained in the mosaic and aligned to the corresponding time reference. The three images in the bottom row represent the original sequence at the same time instants. Figures 7.2 and 7.3 show examples of the reconstructed background and original frames for *Stefan* and *Coastguard*. It can be noted that the foreground information (the girl and car in *NHKvideo7*, the player in *Stefan* or the boat in *Coastguard*) has been practically removed.

7.2 Foreground Mask Extraction

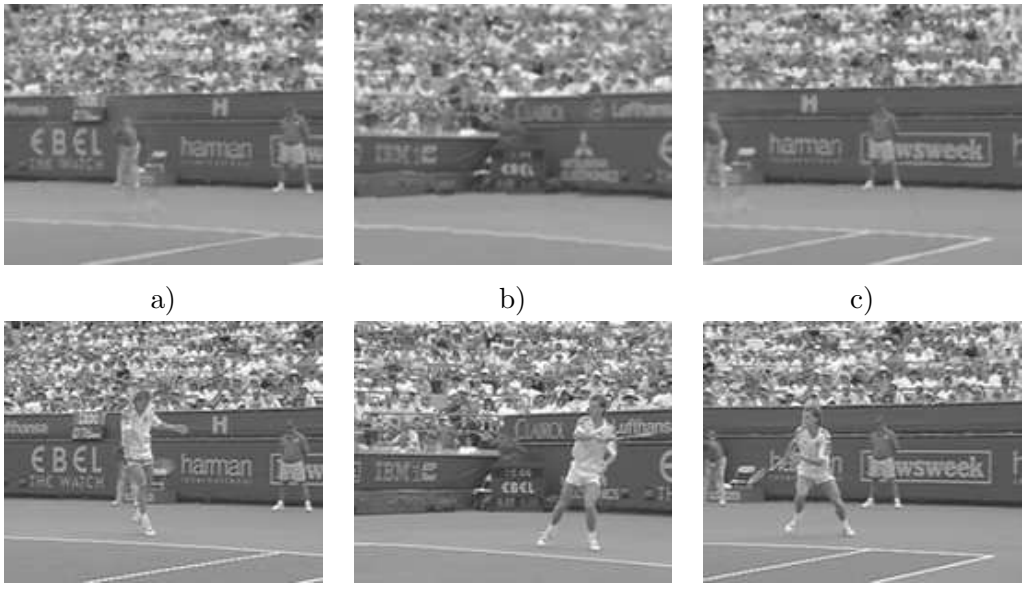
The background image $\mathbf{I}_b(t)$ representing the background information at time t is compared with the current frame $\mathbf{I}(t)$ in order to extract foreground masks at that specific time instant. All relevant foreground information is concentrated in the difference image: $\mathbf{I}(t) - \mathbf{I}_b(t)$.



a) b) c)

d) e) f)

Figure 7.1: Top row shows three frames representing the background information at instants a) $t = 20$, b) $t = 50$ and c) $t = 80$ of the *NHKvideo7* sequence. Bottom row shows the original frames at the same instants.



a) b) c)

d) e) f)

Figure 7.2: Top row shows three frames representing the background information at instants a) $t = 20$, b) $t = 120$ and c) $t = 220$ of the *Stefan* sequence. Bottom row shows the original frames at the same instants.

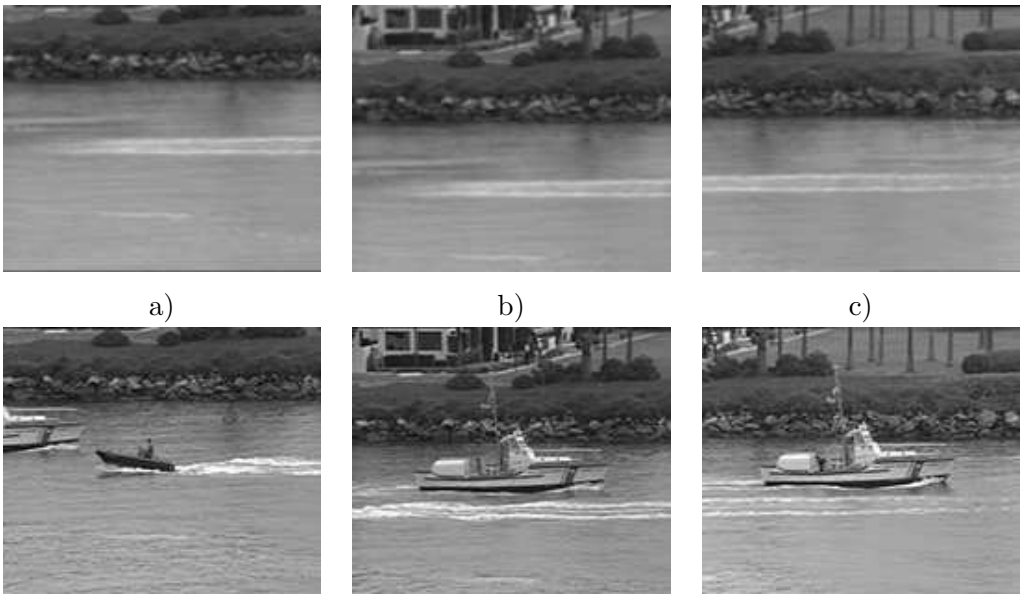


Figure 7.3: Top row shows three frames representing the background information at instants a) $t = 20$, b) $t = 120$ and c) $t = 220$ of the *Coastguard* sequence. Bottom row shows the original frames at the same instants.

Figure 7.4 show the results of the difference between the current frame and the background image for *NHKvideo7*, *Coastguard* and *Stefan*. By analyzing the difference image, foreground regions can be segmented and separated from the background. The analysis is performed using a watershed algorithm [100] and it will provide a foreground label image $\mathbf{L}_{for}(t)$. Each connected component of the foreground label $\mathbf{L}_{for}(t)$ is an estimation of a foreground region at time t. The use of a watershed algorithm allows the segmentation to take advantage of spatial information instead of just motion information (as it was performed in the outliers estimation in the mosaic creation step).

Figure 7.4: Three frames representing the difference $\frac{\mathbf{I}(t)-\mathbf{I}_b(t)}{2} + 128$ for the *NHKvideo7*, *Coastguard* and *Stefan* sequences at $t = 20$.

The watershed algorithm is applied on a gradient image and uses markers to initiate the propagation process and to segment the foreground masks. Markers are obtained by thresholding $|\mathbf{I}(t) - \mathbf{I}_b(t)|$ and by eroding the resulting mask. Two different thresholds, h_{for} and h_{back}, are used to extract foreground and background makers. Let us assume that $\epsilon_s\{\cdot\}$ denotes a binary erosion with a structuring element, s. The foreground and background markers are defined by $\mathbf{E}_{for}(t) = \epsilon_s\{|\mathbf{I}(t) - \mathbf{I}_b(t)| > t_{for}\}$ and $\mathbf{E}_{back}(t) = \epsilon_s\{|\mathbf{I}(t) - \mathbf{I}_b(t)| < t_{back}\}$ respectively. The threshold values were empirically chosen to be $h_{for} = 35$, $h_{back} = 10$ and s a square structuring element whose length is 2 per cent of the original image size. Results have shown these values to be very robust even across different types of sequences and are kept for all results shown in this thesis.

Foreground $\mathbf{E}_{for}(t)$ and background $\mathbf{E}_{back}(t)$ markers are combined in a single marker image $\mathbf{E}(t)$ with a simple binary union operation:

$$\mathbf{E}(t) = \mathbf{E}_{for}(t) \cup \mathbf{E}_{back}(t) \tag{7.2}$$

Figures 7.5 and 7.6 show examples of the extracted markers at specific time instants for the *NHKvideo7* and *Coastguard* sequence. The number of connected components in the marker image is not an issue as a later step of the foreground mask extraction process will merge all connected regions of the resulting watershed segmentation.

a) b) c)

Figure 7.5: Marker images for foreground and background regions for the *NHKvideo7* sequence at frame $t = 5$. a) Shows the markers for background regions (in white), $\mathbf{E}_{back}(t)$. b) Shows selected markers for foreground regions (in grey values), $\mathbf{E}_{for}(t)$. c) Represents the combined markers for foreground and background regions, $\mathbf{E}(t)$. In all images black pixels denote the uncertainty area; that is, regions that will not be used as markers.

The gradient image should indicate the contours of the foreground mask. It is computed from the spatial gradient of $\mathbf{I}(t) - \mathbf{I}_b(t)$, denoted as $\mathcal{G}\{\mathbf{I}(t) - \mathbf{I}_b(t)\}$, where $\mathcal{G}\{\cdot\}$ represents the magnitude of the spatial gradient operator and it is implemented by a standard Sobel filter [54] with a couple of convolution kernels $\mathbf{K}_v$ and $\mathbf{K}_h$ to obtain vertical and horizontal contours respectively:

a) b) c)

Figure 7.6: Marker images for foreground and background regions for the *Coastguard* sequence at frame $t = 106$. As in Figure 7.5, image a) shows the background markers, b) foreground markers and c) the combination of both.

$$\mathbf{K}_v = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad \mathbf{K}_h = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{7.3}$$

However, this gradient highlights contours but also textured areas. To solve this drawback, the gradient is weighted (pixel by pixel) by another gradient between the current image and the previous image, $\mathcal{G}\{\mathbf{I}(t) - \mathbf{I}(t-1)\}$:

$$\mathbf{F}(t) = \mathcal{G}\{\mathbf{I}(t) - \mathbf{I}_b(t)\} \cdot (\mathcal{G}\{\mathbf{I}(t) - \mathbf{I}(t-1)\} \vee \mathcal{G}_0) \tag{7.4}$$

where $\vee$ denotes the maximum operator and $\mathcal{G}_0 = 10$ is used as lower-bound of the weighting gradient, hence, the weight is not too low on static areas.

Figures 7.7 and 7.8 show an example of the proposed gradients for the *NHKvideo7* and the *Coastguard* sequences. The combined gradient image (on the right of the Figures) is finally used by the watershed algorithm to expand the markers. It can be seen as the combined gradient $\mathbf{F}(t)$ is able to remove the contours of texture areas (such as the interior of the ship in the *Coastguard* sequence) while preserving the external contour of the foreground masks.

The watershed segmentation propagates markers on the marker image $\mathbf{E}(t)$ over the combined gradient $\mathbf{F}(t)$. In the resulting image (after the watershed), regions propagated from markers in $\mathbf{E}_{for}(t)$ are treated as foreground regions and regions propagated from markers in $\mathbf{E}_{back}(t)$ as background regions. As the initial markers do not assure than a single marker is used for each connected region in the original image, a final step groups all regions that are connected in the resulting image as the same region. Also, foreground regions that are smaller than the 2% of the image size are not considered and removed. The final foreground label $\mathbf{L}_{for}(t)$ labels the non-connected regions, hence, they can be distinguished and used by later steps of the key-region extraction algorithm.

a) b) c)

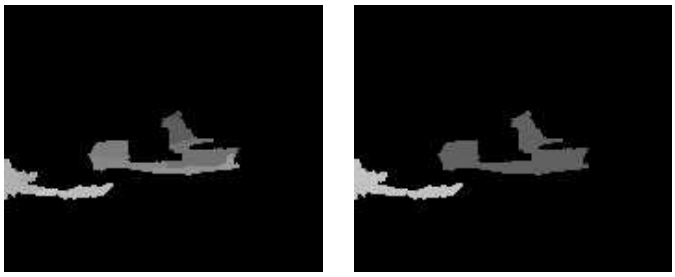
Figure 7.7: Gradient images for the *NHKvideo7* sequence at frame $t = 5$. a) Shows the gradient of $\mathbf{I}(t) - \mathbf{I}_b(t)$. b) Shows the gradient of $\mathbf{I}(t) - \mathbf{I}(t-1)$. c) Represents the combined gradients. To improve the clarity of the images, gradients are stretched to a dynamic range of $[0, 255]$ and inverted, thus, darker pixels represent a higher magnitude on the gradient image.

a) b) c)

Figure 7.8: Gradient images for the *Coastguard* sequence at frame $t = 106$. a) Shows the gradient of $\mathbf{I}(t) - \mathbf{I}_b(t)$. b) Shows the gradient of $\mathbf{I}(t) - \mathbf{I}(t-1)$. c) Represents the combined gradients. To improve the clarity of the images, gradients are stretched to a dynamic range of $[0, 255]$ and inverted, thus, darker pixels represent a higher magnitude on the gradient image.

a) b)

Figure 7.9: Foreground label for the *NHKvideo7* sequence at frame $t = 5$. a) Shows the resulting watershed segmentation where black pixels denote background regions from white markers in Figure 7.5.a. b) Shows the final foreground label $\mathbf{L}_{for}(t)$. Two different foreground regions representing the car and the girl are segmented.



a) b)

Figure 7.10: Foreground label for the *Coastguard* sequence at frame $t = 106$. a) Shows the watershed segmentation. b) Shows the final label $\mathbf{L}_{for}(t)$ with all connected regions grouped. Two different foreground regions representing the boat and the ship are segmented.

Figures 7.9 and 7.10 show the final foreground label for the sequences *NHKvideo7* and *Coastguard*. Each connected component in the foreground label image $\mathbf{L}_{for}(t)$ can be seen as a first approximation of the mask of a foreground region at time t. The next step of the algorithm will show how to improve this approximation based on previous estimations.

7.3 Key-region Mask Estimation

In the previous section, it has been seen how the foreground label image $\mathbf{L}_{for}(t)$ is an estimation of the key-region masks at time t. However, this estimation is not very reliable because it is obtained on the basis of the observation of a single time instant. Therefore, it is possible that, for a specific time instant, the foreground region may have the same gray value as the background and the difference image may be zero. To improve the robustness of the analysis, the key-region mask estimation step combines the contour information of the foreground masks extracted at past time instants and selects the most reliable sections to create the final mask of the key-region k at time instant t, $\mathbf{M}_{kr}^{k}(t)$.

Prior to the combination of contours with past time instants, a previous step to associate each connected component of the foreground label to existing key-regions that are stored in the key-region memory is performed. The key-region memory (bottom-right block of Figure 5.5) is responsible for keeping all key-regions that have been extracted at previous time instants. For each key-region, the appearance, texture and contour images are stored in the key-region memory.

7.3.1 Key-region Association with Foreground Masks

Several strategies can be followed in order to associate connected components of $\mathbf{L}_{for}(t)$ to existing key-regions. To clarify the notation, each connected component of the foreground

label $\mathbf{L}_{for}(t)$ will be referred as a candidate foreground mask through the rest of the section. For instance, one strategy to associate candidate foreground masks to key-regions may consist of measuring the mean square error between the texture corresponding to the candidate foreground mask (in the original image at time t) and the texture of the existing key-regions k (in the texture image of the key-region). Candidate foreground masks will be assigned to the key-region with the minimum error between texture images. Another technique may involve measuring the overlap between candidate foreground masks and key-regions. More complicated techniques may include computing the histogram of candidate foreground masks and comparing similarities between histograms of the existing key-regions.

If standard video sequences of 25 or 30 fps are to be considered, the overlapping approach works very well and presents a very good relation between its complexity and its robustness to associate candidate foreground masks to existing key-regions. In particular, a candidate foreground mask is assigned to an existing key-region k if it sufficiently overlaps with the corresponding key-region mask. If the current foreground mask does not correspond to any known key-region, a new key-region is created.

The process to associate candidate foreground masks to existing key-regions is as follows. Each candidate foreground mask (denoted by the index c) is compared against all existing key-regions extracted from previous frames. The comparison is made with the appearance image of the key-region k (see Section 5.1) where all pixels with value greater than 0 are considered as the mask for the key-region k.

As all three key-region images are aligned to the first frame reference, before the overlap can be computed, each candidate foreground mask must be aligned to the same temporal reference ($t = 0$) for all key-regions k extracted up to time t. Again, if real sequences of 25 or 30 fps are to be considered, there is no need to estimate the motion of the candidate foreground mask between instants $t - 1$ and t. The overlap is computed using the pixels of both the aligned candidate foreground mask and the key-region appearance image. Eq. (7.5) defines the overlap between a candidate foreground mask, with index c, and a key-region, with index k, as:

$$o_c^k = \frac{N_a - N_a^k}{N_a} \tag{7.5}$$

where N_a is the number of pixels in the aligned candidate foreground mask and N_a^k is the number of common pixels between the aligned candidate foreground mask and the appearance image for key-region k. The normalization by N_a in Eq. (7.5) ensures that the overlap keeps a dynamic range between 0 and 1, it also guarantees that if one foreground region is over-segmented into two different candidate foreground masks, both candidates are associated to the same key-region. New regions are created if the minimum overlap o_c^k against all previous key-regions in the key-region memory exceeds a simple threshold equal to $o_t = 0.8$.

Figure 7.11 shows an example of the association with existing key-regions for the two

candidate foreground masks (corresponding to the girl and the car) of the *NHKvideo7* sequence extracted from the connected components of the foreground label $\mathbf{L}_{for}(t)$ (shown in Figure 7.9). Each candidate foreground mask c is compared with all existing key-region in memory. In this example two key-regions existed. Figure 7.11.a and 7.11.b show the overlapping of the candidate foreground mask $c = 1$ (dark gray region in Figure 7.9.b corresponding to the girl) with the stored images of the two existing key-regions. On the bottom row (Figures 7.11.c and 7.11.d), the second candidate foreground mask $c = 2$ (light gray pixels corresponding to the car) is compared against the same key-regions (the girl in 7.11.c and the car in 7.11.d).

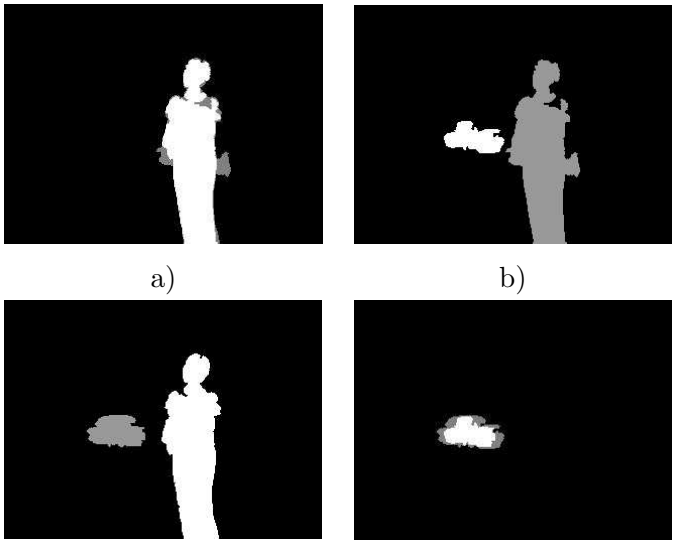


Figure 7.11: Example of the overlapping of candidate foreground masks in $\mathbf{L}_{for}(t)$ and existing key-regions for the *NHKvideo7* sequence at frame $t = 5$. a) Shows the overlap between the aligned first candidate $c = 1$ (the girl) and key-region $k = 1$ (also corresponding to the girl). b) Shows the overlap between the aligned first candidate $c = 1$ (the girl) and key-region $k = 2$ (corresponding to the car). c) Shows the overlap between the aligned second candidate $c = 2$ (the car) and key-region $k = 1$ (corresponding to the girl). d) Shows the overlap between the aligned second candidate $c = 2$ (the car) and key-region $k = 2$ (also corresponding to the car). Grey pixels correspond to the candidate foreground mask and white pixels correspond to the key-region mask.

In this example, the overlap between candidate foreground masks and key-regions give $o_1^1 = 0.07$, $o_1^2 = 1.0$, $o_2^1 = 1.0$, $o_2^2 = 0.27$. A candidate foreground mask is associated with the key-region with the largest overlap (smallest overlap measure) and therefore candidate $c = 1$ is associated with the key-region $k = 1$ (the girl) and the candidate mask $c = 2$ with key-region $k = 2$ (the car).

After each connected foreground mask in the foreground label is associated with a key-

region, the foreground mask can be improved based on the previous knowledge of the extracted key-region. In particular, the contour information is used to improve the foreground mask extraction on a frame by frame basis.

7.3.2 Foreground Mask Contour Definition

In the previous step, each candidate foreground mask is assigned to an existing key-region k. Let us call each assigned candidate foreground mask as *associated foreground mask* and represent it by the image $\mathbf{M}^c_{for}(t)$ where the super-index c represents the associated foreground mask index. Before any improvement on $\mathbf{M}^c_{for}(t)$ can be made, the associated foreground mask c and the key-region k must be aligned to the same temporal reference. This alignment is performed by estimating the motion between the texture of the associated foreground mask and the texture of the corresponding key-region. This process is the same as the warping process when constructing the mosaic (Section 6.5). First, the local motion of each associated foreground mask is estimated (between the current image at time t and the previous image at $t-1$). Secondly, this motion is accumulated with the motion of the associated key-region k. In order to prevent the propagation of errors, a 4VD motion estimation is performed between the texture of the associated foreground region at time t and the texture of the key-region (as it was performed on the mosaic). The result of this motion estimation step is a set on motion parameters $\mathbf{g}^c_k(t)$ which relates each associated foreground mask c at time instant t with its correspondent key-region k.

After the previous motion estimation, both associated foreground mask and key-region can be aligned to the same temporal reference. Let us denote by $\hat{\mathbf{M}}^c_{for}(t)$ the motion compensated versions of the associated foreground mask using the estimated motion parameters $\mathbf{g}^c_k(t)$. $\hat{\mathbf{M}}^c_{for}(t)$ and the key-region images are now aligned at the same temporal reference. Now, before the updating of the key-region images with the new information at time t is done (in the key-region modeling step described in Section 7.4), the associated foreground masks are improved based on the reliability of their contours.

An example of the associated foreground mask extracted from the *NHKvideo7* sequence can be seen on the left side of Figure 7.12. Note that, the contour of the associated foreground region is not always reliable. The goal is to combine its contour information with the contour information of the associated key-region k extracted at previous time instants taking into account the reliability of contours in time. The update of the foreground shape starts from the motion compensated associated foreground mask $\hat{\mathbf{M}}^c_{for}(t)$ and is performed as follows. Let us assume that $\mathbf{I}$ is an image and $\mathbf{M}$ a mask, $\mathcal{C}\{\mathbf{I},\mathbf{M}\}$ denotes an image equal to zero except on the contours of $\mathbf{M}$ where it takes the values of $\mathbf{I}$. The contour reliability of the associated foreground region c is obtained by:

$$\mathbf{C}_{\hat{\mathbf{M}}^c_{for}}(t) = \mathcal{C}\left\{\mathcal{G}\{\hat{\mathbf{I}}(t)\}, \hat{\mathbf{M}}^c_{for}(t)\right\} \tag{7.6}$$

where $\mathcal{G}$ denotes the gradient operator as in Section 7.2 and $\hat{\mathbf{I}}(t)$ corresponds to the original image compensated with the motion parameters $\mathbf{g}_k^c(t)$, thus, the corresponding texture of the associated foreground mask is also compensated to the same temporal reference.

The pixels value of this contour reliability is a confidence measure of the contours of the associated foreground region $\mathbf{R}_{for}^c(t)$. Low values imply that the corresponding contour does not correspond to contrasted edges. This can occur when the foreground occludes a background region of the same color. High values on the contour measure correspond to strong edges on the original image and therefore to reliable contours. Figure 7.12 illustrates the use of the contour image to correct possible segmentation errors in the foreground mask extraction algorithm. In this example, the foreground mask extracted at frame $t = 2039$ of the *NHKvideo7* sequence is of poor quality due to a low contrast between the girl and the background in that specific time instant.

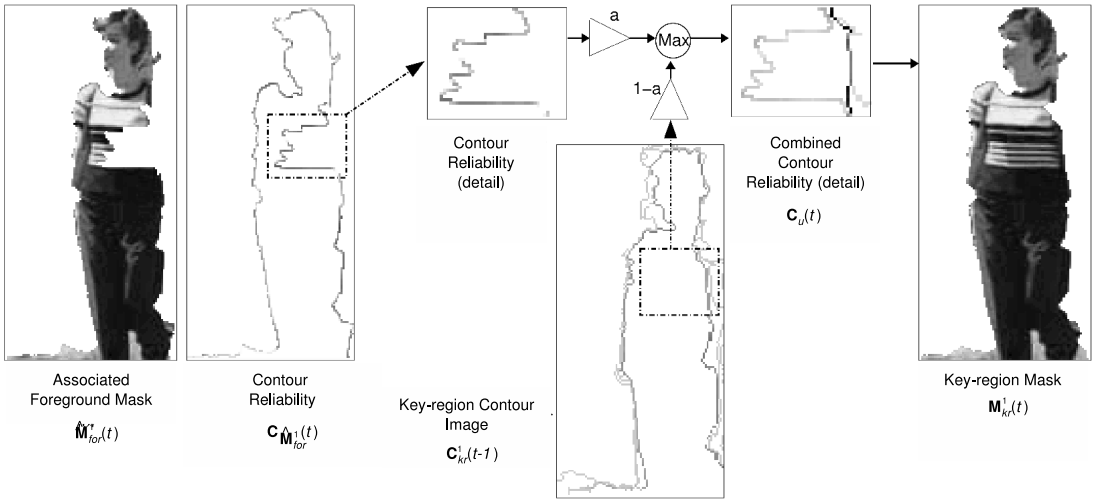


Figure 7.12: Creation of the key-region mask $\mathbf{M}_{kr}^1(t)$ taking into account the contour reliability $\mathbf{C}_{\hat{\mathbf{M}}_{for}^1}(t)$ of the associated foreground mask $\hat{\mathbf{M}}_{for}^1(t)$ and of past contour information of the key-region contour image $\mathbf{C}_{kr}^1(t-1)$. For a clearer representation, pixels in the mask images have been replaced by the corresponding texture.

The two images on the left side of Figure 7.12 show the associated foreground region $\hat{\mathbf{R}}_{for}^c(t)$ and the measure of contour reliability $\mathbf{C}_{\hat{\mathbf{M}}_{for}^c}(t)$ (as in Eq. (7.6)). This measure of the contour reliability on the associated foreground mask is compared with the accumulated measures from the previous associated foreground masks assigned to the same key-region, $\mathbf{C}_{kr}^k(t-1)$. This accumulated measure is part of the key-region model (contour image in Section 7.4). The corresponding contour image of the key-region is shown on the bottom image of Figure 7.12. The combination of the contour reliability at time t, $\mathbf{C}_{\hat{\mathbf{M}}_{for}^c}(t)$, and the contour image of the key-region $\mathbf{C}_{kr}^k(t-1)$ at time $t-1$ is done by a maximum operation:

$$\mathbf{C}_u(t) = a\mathbf{C}_{\hat{\mathbf{M}}^c_{for}}(t) \vee (1-a)\mathbf{C}^k_{kr}(t-1) \tag{7.7}$$

The parameter $a \in [0,1]$ controls the memory of the allowed modifications to the shape of extracted foreground masks. If $a \simeq 0$, previously segmented key-region contours are trusted more than the current contours from the foreground mask. In this case, errors in the foreground mask are easier to fix but tracking non-rigid foreground regions becomes more difficult. On the other hand, if $a \simeq 1$, non-rigid regions are easier to track but segmenting errors are also more difficult to correct. In our case, a value of $a = 0.7$ has been selected as a good trade off between robustness and adaptability.

As in the initial foreground mask extraction step (Section 7.2), the estimation of the final mask of the foreground region $\mathbf{M}^k_{kr}(t)$ is done with a watershed algorithm. The markers for this watershed consist of two points, one inside the foreground mask and one outside (in the background). The gradient image used in the watershed is the combined contour reliability $\mathbf{C}_u(t)$ as in Eq. (7.7). The output of the watershed algorithm is the key-region mask at time t $\mathbf{M}^k_{kr}(t)$ where the most reliable contour parts from the associated foreground mask and from the corresponding key-region have been used. An example of the resulting key-region mask is shown on the right side of Figure 7.12. The initial error in the associated foreground mask shape has been eliminated and replaced by the most reliable contour observed in the past. In general, this procedure allows the progressive improvement of the key-region contours on a frame by frame basis taking into account the reliability of past extracted key-region contours.

7.4 Key-region Modeling

7.4.1 Key-region Updating

The final step of the algorithm creates and updates a model for each key-region observed in the scene (key-region update block of Figure 5.5). The key-region model consists of three images. An appearance image, a contour image and a texture image. The appearance image $\mathbf{A}^k_{kr}(t)$ represents the frequency with which a pixel has been estimated as belonging to key-region k. If a pixel of $\mathbf{A}^k_{kr}(t)$ has a value of 5 then it means that the pixel has been segmented as belonging to key-region k in 5 different frames. The contour image $\mathbf{C}^k_{kr}(t)$ stores the confidence of the key-region contours and is used to modify the input foreground masks on a frame basis as seen in the previous section. Finally, the texture image $\mathbf{T}^k_{kr}(t)$ represents the overall texture of the key-region.

The key-region mask $\mathbf{M}^k_{kr}(t) = 1$ denotes pixels that have been extracted and assigned to key-region k at time t. The contour reliability is $\mathbf{C}_{\mathbf{M}^k_{kr}}(t) = \mathcal{C}\left\{\mathcal{G}\{\hat{\mathbf{I}}(t)\}, \hat{\mathbf{M}}^k_{kr}(t)\right\}$ and is extracted as described by Eq. (7.6) but this time using the final key-region mask $\mathbf{M}^k_{kr}(t)$ obtained in the last step. The equations that update each key-region image can be summarized

as (pixel by pixel operations are implied):

$$\mathbf{T}^k_{kr}(t) = \frac{\mathbf{A}^k_{kr}(t-1)\cdot\mathbf{T}^k_{kr}(t-1) + \mathbf{M}^k_{kr}(t)\hat{\mathbf{I}}(t)}{\mathbf{A}^k_{kr}(t)} \qquad \mathbf{C}^k_{kr}(t) = \frac{\mathbf{A}^k_{kr}(t-1)\cdot\mathbf{C}^k_{kr}(t-1) + \mathbf{C}_u(t)}{\mathbf{A}^k_{kr}(t)}$$
$$\mathbf{A}^k_{kr}(t) = \mathbf{A}^k_{kr}(t-1) + \mathbf{M}^k_{kr}(t) \tag{7.8}$$

The equations above are similar to the Eq. (6.20) which is used to update the mosaic in the blending step. Each updated pixel in the key-region images is weighted by $\mathbf{A}^k_{kr}(t)$. Therefore, every key-region mask which contributes to build the key-region images only contributes the corresponding fraction on the final key-region images. For instance, if the sequence is composed of 100 frames, each key-region mask (computed at each time instant) contributes a fraction of 1/100 to the key-region images.

Figure 7.13: Appearance images $\mathbf{A}^1_{kr}(t)$ for the key-region $k = 1$ in the *ETRI_od_B* sequence at time instants, from left to right and top to bottom: $t = 120$, $t = 135$, $t = 150$, $t = 165$, $t = 180$ and $t = 195$.

Figures 7.13, 7.14 and 7.15 show an example of how the key-region images are created over time. Some key-frames of the *ETRI_od_B* sequence used in the experiment can be seen on Figure B.3. This sequence is a good example to show some results of the key-region extraction as it corresponds to a static scene with no motion of the camera. The sequence consists of a view of an outdoor field where some people are walking in front of the camera. The example shows how the three images are constructed over time using the input frames of the sequence and the previously computed mosaic (in this case, the mosaic is just a frame without foreground people). It is important to remember that all images are aligned to the same temporal reference which corresponds to the first frame where the key-region appears.

Figure 7.13 shows the construction of the appearance image $\mathbf{A}^1_{kr}(t)$ at different time

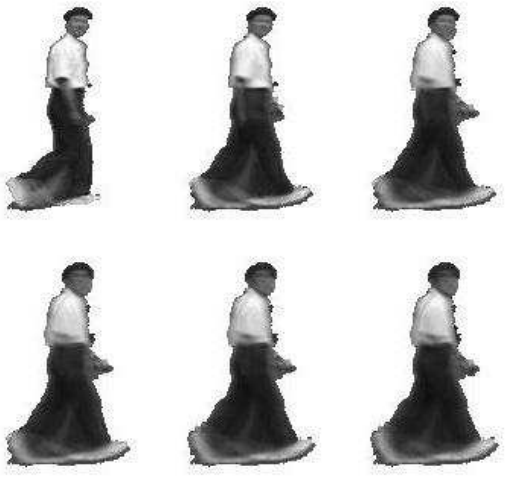


Figure 7.14: Texture images $\mathbf{T}_{kr}^{1}(t)$ for the key-region $k = 1$ in the *ETRI_od_B* sequence at time instants, from left to right and top to bottom: $t = 120$, $t = 135$, $t = 150$, $t = 165$, $t = 180$ and $t = 195$.

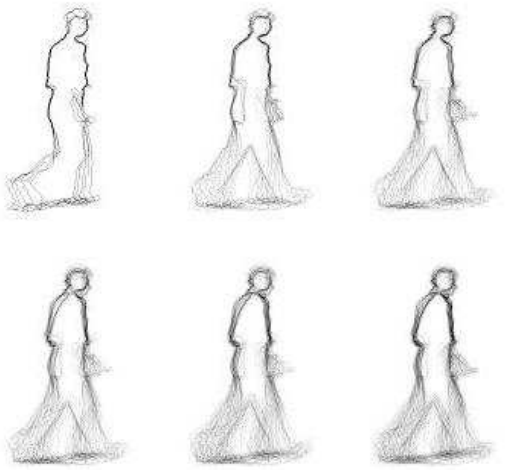


Figure 7.15: Contour images $\mathbf{C}_{kr}^{1}(t)$ for the key-region $k = 1$ in the *ETRI_od_B* sequence at time instants, from left to right and top to bottom: $t = 120$, $t = 135$, $t = 150$, $t = 165$, $t = 180$ and $t = 195$. To improve the clarity of the images, gradients are stretched to a dynamic range of $[0, 255]$ and inverted, thus, darker pixels represent a higher magnitude on the gradient image.

instants. For a clear representation of the appearance image, the images on Figure 7.13 are inverted, thus, darker pixels denote pixels that are assigned to the key-region more often than lighter pixels. Figure 7.14 show the construction of the texture image $\mathbf{T}_{kr}^{1}(t)$ for the same time instants. The texture image is a combination of the foreground regions at all time

instants. Therefore, if the key-region is not rigid, for instance the legs, the texture is not very well defined as a result of the mixture of all previous frames. Finally, Figure 7.15 shows the construction of the contour image $\mathbf{C}^1_{kr}(t)$. This image is representative of the contour information of the extracted key-region. Zones of the key-region with well defined boundaries (such as the torso or the head) have a higher contour value than zones as the legs.

All appearance, contour and texture images contain information of the activity followed by the key-region. In this case, higher body parts (body, chest) show no relative motion while lower parts (legs) show a considerable amount of relative motion. As more frames of the input sequence are analyzed, more information is combined into the images. Therefore, in the case of non-rigid key-region, images may diverge from the current key-region mask and new key-regions may be created (even if they correspond to the same semantic object). The following section details the approach followed to group resulting key-regions that represent the same non-rigid foreground object.

7.4.2 Key-region Grouping

The grouping of key-regions that represent the same semantic foreground object is performed when all input frames are processed and all the resulting key-regions of the sequence are extracted and updated. Key-regions are grouped based on histogram similarity of the final texture image $\mathbf{T}^k_{kr}$ of the key-regions.

A weighted histogram $\mathbf{h}_k$ is computed for each key-region k of the sequence. Each pixel (x, y) in the texture image is weighted by its contribution to the appearance image $\mathbf{A}^k_{kr}$. The appearance image can be seen as a measure of the probability of one pixel to belong to the key-region k. If each pixel contribution to the histogram is weighted by the factor $\mathbf{A}^k_{kr}(x, y)/max(\mathbf{A}^k_{kr})$, pixels that have been segmented as part of the key-region more often than others will have a greater contribution to the weighted histogram $\mathbf{h}_k$.

The extraction of the weighted histogram of N bins is as follows. Let us denote by $\mathbf{P}_i(x, y)$ an auxiliary image where a pixel is equal to 1 if the corresponding pixel in the texture image $\mathbf{T}^k_{kr}$ has a gray value included in the weighted histogram bin $\mathbf{b}(i)$ or 0 otherwise. Each weighted histogram bin $\mathbf{b}(i)$ corresponds to the uniform quantization of the dynamic range of the texture image (in our case $[0, 255]$) into N bins.

$$\mathbf{P}_i(x, y) = \begin{cases} 1 & if \quad \mathbf{T}^k_{kr}(x, y) \in \mathbf{b}(i) \\ 0 & other \end{cases} \tag{7.9}$$

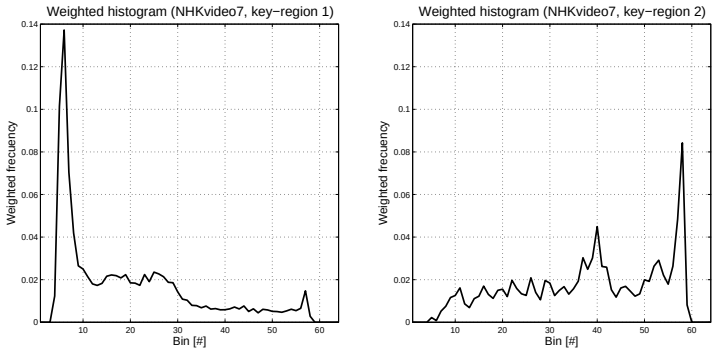
The weighted histogram $\mathbf{h}_k$ is computed over all the pixels that have been associated to the key-region k. The pixels that contribute to the weighted histogram are the ones where the appearance image have a value greater than 0 (they have been selected as belonging to key-region k at least once). To compensate the different resolution of the key-regions, the weighted histogram is normalized by the total number of elements in $\mathbf{h}_k$:

$$\mathbf{h}_k(i) = \frac{\sum_{(x,y)\in \mathbf{A}^k_{kr}>0} \frac{\mathbf{A}^k_{kr}(x,y)}{max(\mathbf{A}^k_{kr})} \cdot P_i(x,y)}{\sum_{i=0}^{N} \mathbf{h}_k(i)} \tag{7.10}$$

The similarity between two weighted histograms of two different key-regions is computed using the histogram intersection [93]. For two weighted histograms $\mathbf{h}_k$ and $\mathbf{h}_{k'}$ containing N bins and corresponding to two key-regions k and k', the weighted histogram intersection $s_k^{k'}$ is defined as:

$$s_k^{k'} = \sum_{i=0}^{N} min(\mathbf{h}_k(i), \mathbf{h}_{k'}(i)) \tag{7.11}$$

The weighted histogram intersection represents a histogram similarity. If $s_k^{k'} = 0$ then the two histograms do not share any information. However, if $s_k^{k'} = 1$, then both histograms are the same. In the case of the key-region grouping, key-regions are grouped together to represent the same foreground object if the weighted histogram intersection between the two weighted histograms is greater than a simple threshold equal to $s_t = 0.7$. The number of bins in the weighted histogram is chosen to be $N = 64$.



a) b)

Figure 7.16: Weighted histograms for the key-regions in *NHKvideo7* test sequence. a) Weighted histogram $\mathbf{h}_1$ for key-region $k = 1$ representing the girl. b) Weighted histogram $\mathbf{h}_2$ for key-region $k = 2$ representing the car.

For instance, in the *NHKvideo7* test sequence, the two key-regions (displayed in Figure 5.3) have weighted histograms such as those represented on Figure 7.16. The key-region extraction algorithm is able to extract only two key-regions which correspond to the girl and the car respectively. The weighted histogram intersection between both key-regions results in a value of $s_1^2 = 0.49$. Therefore, both key-regions are considered to represent different foreground objects in the scene.

In the case, for instance, of the *Stefan* test sequence, the key-region extraction algorithm detects three key-regions (detailed in Figure 8.6 on the following Section 8.1). The tennis

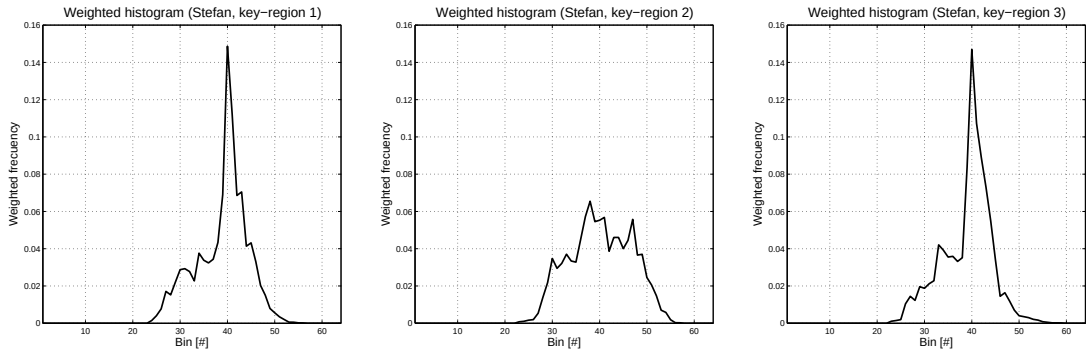


Figure 7.17: Weighted histograms for the key-regions in *Stefan* test sequence. a) Weighted histogram $\mathbf{h}_1$ for key-region $k = 1$. b) Weighted histogram $\mathbf{h}_2$ for key-region $k = 2$. c) Weighted histogram $\mathbf{h}_3$ for key-region $k = 3$. All three key-regions represent the same foreground object.

player is a clear example of a non-rigid object and, therefore, the proposed algorithm extracts up to 3 key-regions corresponding to the same tennis player. Figure 7.17 shows the weighted histograms of the key-regions. The weighted histogram intersection for the key-regions are: $s_1^2 = 0.75$, $s_1^3 = 0.91$ and $s_2^1 = 0.71$. Therefore, in the proposed method, all three key-region are grouped to represent the same foreground object, in this case, the tennis player.

The following chapter will show several results of the mosaic and key-region extraction algorithm for different test sequences. The chapter will also study the synergy between this indexing representation and the compression one.

Chapter 8

Results

8.1 Shot Representation

The main goal of the shot representation is to summarize the video shot using a mosaic and a set of key-regions. The mosaic by itself can easily represent the background information of the entire scene. In order to represent foreground information a set of key-region are extracted and modeled. As reviewed in the previous chapters of Part II of this thesis, the main advantage of this approach is the ability to represent complex scenes with non-static backgrounds and with moving foreground objects. At the same time, the representation combines both compression and browsing functionalities by sharing a mosaic for the background. Foreground regions are modeled using key-regions which may be also an attractive representation to include other functionalities such as the activity of non-rigid foreground objects.

On the other hand, disadvantages of the algorithm include the computational complexity as several motion estimations (for each foreground object in the scene) must be performed at each time instant. The algorithm requires a two-pass approach, first to compute the mosaic and then to model key-regions, and therefore it is not suitable for real-time applications. The extraction and modeling of key-regions is robust against camera motions but fails in the presence of occlusions between foreground objects.

Different results for several test sequences are shown in this chapter in order to illustrate the advantages and limits of the representation. Figure 8.1 represents two of the key-regions extracted from the *ETRI_od_B* sequence (Figure B.3). The key-regions correspond to two different persons walking in front of the camera. The deformation of the non-rigid key-regions is captured in the three images where the upper parts of the body are less blurred (and the contours are more defined) than the lower part of the body. Therefore, only one key-region represents each foreground object of the scene.

In order to complete the shot representation of the scene, the motion followed by all key-regions can be superimposed on the mosaic. In particular, the motion of the center

$\mathbf{A}_{kr}^1$ $\mathbf{C}_{kr}^1$ $\mathbf{T}_{kr}^1$

a) Key-region $k = 1$ corresponding to the first person that appears from times $t = 115$ to $t = 225$

$\mathbf{A}_{kr}^2$ $\mathbf{C}_{kr}^2$ $\mathbf{T}_{kr}^2$

b) Key-region $k = 2$ corresponding to the second person that appears from times $t = 405$ to $t = 505$

Figure 8.1: Two key-regions, a) and b), for 500 frames of the *ETRI_od_B* sequence. From left to right, the appearance $\mathbf{A}_{kr}$, contour $\mathbf{C}_{kr}$ and texture $\mathbf{T}_{kr}$ images are represented for each key-region.

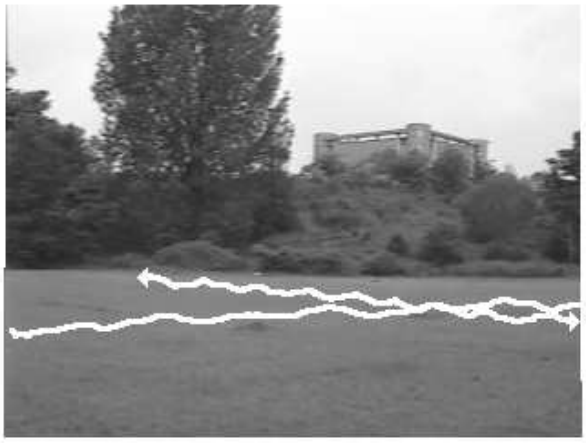


Figure 8.2: Mosaic for 500 frames of the *ETRI_od_B* video sequence (352x240 at 30 fps) with superimposed trajectories for the two key-regions in Figure 8.1.a and 8.1.b.

of gravity of the key-region is used. Figure 8.2 shows, in white arrows, the motion of the extracted foreground key-regions. The motion trajectories are computed using the centroids

of the extracted key-regions over time and an arrow is painted at the end of the trajectory to signal the direction of the motion followed by each key-region in the shot. The final shot representation is composed of the mosaic (Figure 8.2) and the key-regions $k = 1$ and $k = 2$ (Figure 8.1).

Figure 8.3: Mosaic representing 100 frames of the *NHKvideo7* video sequence ($352x240$ at 30 fps).

$\mathbf{A}^1_{kr}$ $\mathbf{C}^1_{kr}$ $\mathbf{T}^1_{kr}$

a) Key-region $k = 1$

$\mathbf{A}^2_{kr}$ $\mathbf{C}^2_{kr}$ $\mathbf{T}^2_{kr}$

b) Key-region $k = 2$

Figure 8.4: Two key-regions, a) and b), for the *NHKvideo7* sequence. From left to right, the appearance $\mathbf{A}^k_{kr}$, contour $\mathbf{C}^k_{kr}$ and texture $\mathbf{T}^k_{kr}$ images are represented for each key-region.

In the case of non-static backgrounds with complex motions, the shot representation can also provide good models for the corresponding key-regions. The example for 100 frames of the *NHKvideo7* sequence is very representative of foreground extraction on complex backgrounds. The final shot representation using a mosaic and key-regions is shown in Figures 8.3 and 8.4. The mosaic shows the background information and the trajectory followed by the two key-regions of the scene (the car at the top and the girl at the bottom). Again, arrows indicate the direction of the motion followed by both key-regions.

The two extracted key-regions are represented on Figure 8.4. In this example, the girl on the front of the scene creates the first key-region $k = 1$ while the car at the far end constitutes the second key-region $k = 2$. The shot representation for the *NHKvideo7* sequence successfully follows both foreground regions and manages to create one key-region for each of them.

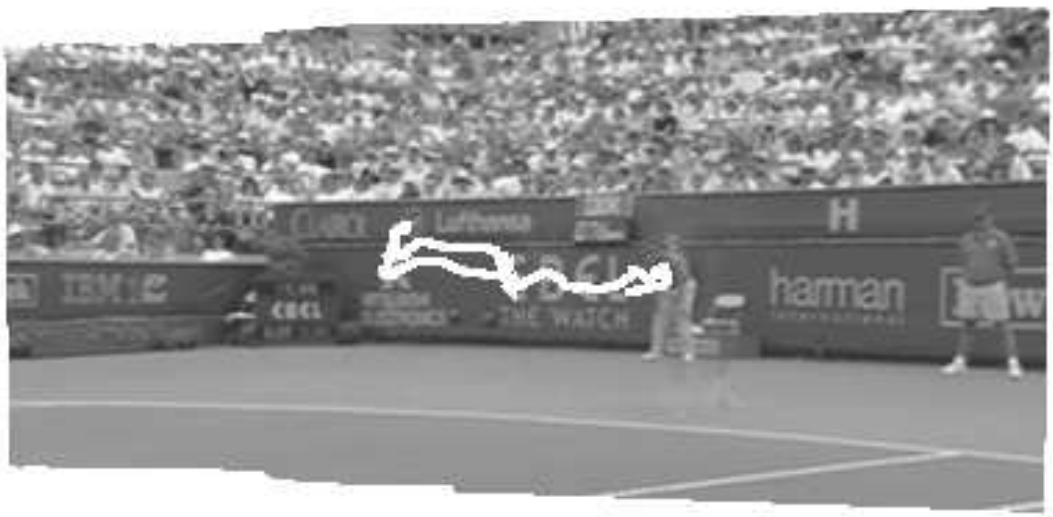


Figure 8.5: Mosaic representing 150 frames of the *Stefan* video sequence. The trajectories of the extracted key-regions are added to the mosaic as white arrows.

In the case, for instance, of the *Stefan* video sequence, several key-regions are created due to the fact that the tennis player is deforming too much and that sometimes it also has a similar gray value than the background. Figure 8.6.a, 8.6.b and 8.6.c show the three key-regions created. As the tennis player is moving very rapidly, the key-region images, and especially the texture images, are blurred as the perspective motion model is not able to represent the foreground object deformations. However, the key-region grouping step, as described in Section 7.3.1, is capable to detect that all three key-regions represent the same foreground object.

Figure 8.5 represents the mosaic with the background information and the trajectories of the three extracted key-regions. As all three key-regions correspond to the same object, all three trajectories link together, thus, the end of the trajectory of key-region $k = 1$ corresponds

$\mathbf{A}^1_{kr}$ $\mathbf{C}^1_{kr}$ $\mathbf{T}^1_{kr}$

a) Key-region $k = 1$ corresponding to the tennis player that appears from times $t = 0$ to $t = 59$

$\mathbf{A}^2_{kr}$ $\mathbf{C}^2_{kr}$ $\mathbf{T}^2_{kr}$

b) Key-region $k = 2$ corresponding to the tennis player that appears from times $t = 60$ to $t = 104$

$\mathbf{A}^3_{kr}$ $\mathbf{C}^3_{kr}$ $\mathbf{T}^3_{kr}$

c) Key-region $k = 3$ corresponding to the tennis player that appears from times $t = 105$ to $t = 150$

Figure 8.6: Three extracted key-regions for the *Stefan* sequence. All key-regions correspond to the same semantic object, the tennis player.

to the beginning of key-region $k = 2$.

In order to improve the key-region representation of non-rigid foreground objects, such as the tennis player in the *Stefan* sequence, more key-regions can be extracted to represent the same foreground object. Section 7.3.1 reviewed the approach employed by the key-region extraction algorithm to associate new foreground regions to existing key-regions (by using a threshold o_t). If the threshold o_t is lowered, more key-regions are created. Figure 8.7 shows the 6 key-regions extracted for the *Stefan* sequence if a threshold of $o_t = 0.5$ is used. The 6 key-regions are able to represent the non-rigid tennis player more accurately than when only 3 key-regions were used. One drawback of using more key-regions to represent a single non-rigid foreground object is that the proposed grouping algorithm sometimes fails and it is not able to group together all key-regions. For instance, in the case of the *Stefan* sequence, only regions $k = 1, 2, 3, 5, 6$ are grouped together automatically. Due to the pose of the tennis player, the key-region $k = 4$ (represented in Figure 8.7) is brighter than the rest of key-regions

and the weighted histogram corresponding to key-region $k = 4$ is sufficiently different to be considered as a different foreground region.

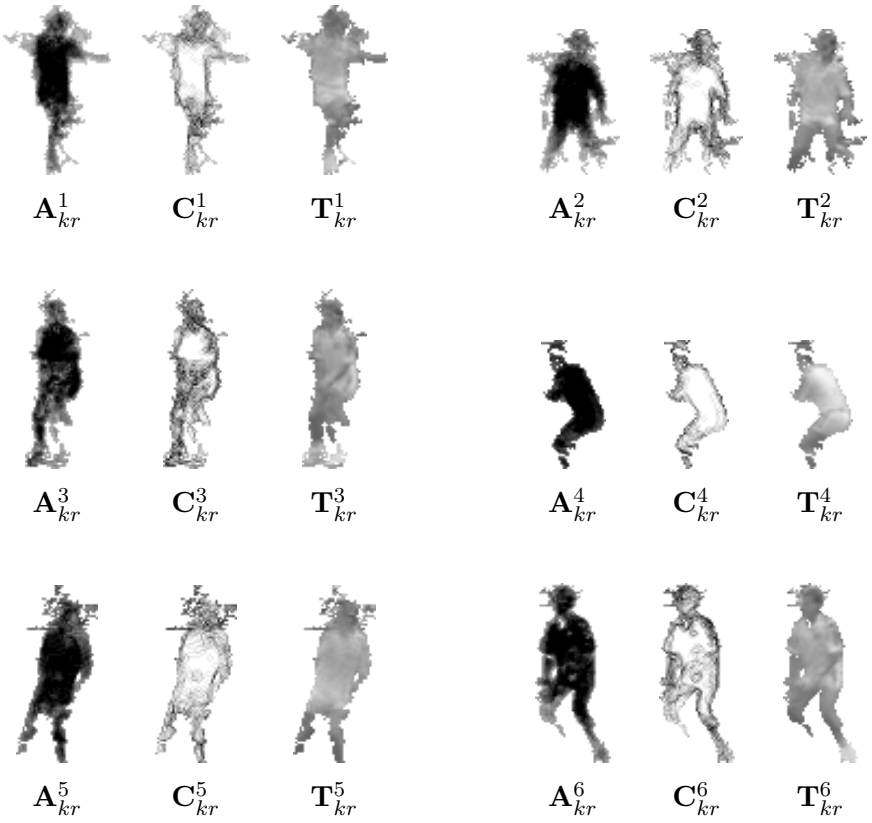


Figure 8.7: Six key-regions for the *Stefan* sequence with $o_t = 0.5$. Key-region 1 corresponds to the player that appears from time instants $t = 0$ to $t = 28$, key-region 2 appears from times $t = 29$ to $t = 47$, key-region 3 appears from times $t = 48$ to $t = 81$, key-region 4 appears from times $t = 82$ to $t = 98$, key-region 5 appears from times $t = 99$ to $t = 125$ and, finally, key-region 6 with the player from times $t = 126$ to $t = 150$.

One of the main disadvantages of the proposed representation algorithm is the handling of occlusions between foreground objects. In this case, the key-region representation is not able to separate foreground objects and new key-regions are created with the merged objects. A good example of this can be seen in the representation of the *Coastguard* sequence. This sequence represents a ship and a boat that at one point cross each other. The sequence representation algorithm cannot split both objects and it creates a new key-region when the boat occludes the ship.

The 5 key-regions extracted on the *Coastguard* sequence are represented on Figure 8.8. Key-region on Figure 8.8.c shows how the crossing of the boat and the ship of the original sequence creates a new key-region, $k = 3$, with both foreground objects merged. As a consequence of the merging and as both the boat and the ship have opposite motions, the

$\mathbf{A}^1_{kr}$ $\mathbf{C}^1_{kr}$ $\mathbf{T}^1_{kr}$

a) Key-region $k = 1$ corresponding to the boat ($t = 0$ to $t = 35$)

$\mathbf{A}^2_{kr}$ $\mathbf{C}^2_{kr}$ $\mathbf{T}^2_{kr}$

b) Key-region $k = 2$ corresponding to the ship ($t = 0$ to $t = 35$)

$\mathbf{A}^3_{kr}$ $\mathbf{C}^3_{kr}$ $\mathbf{T}^3_{kr}$

c) Key-region $k = 3$ corresponding to the merging of the boat and the ship ($t = 36$ to $t = 95$)

$\mathbf{A}^4_{kr}$ $\mathbf{C}^4_{kr}$ $\mathbf{T}^4_{kr}$

d) Key-region $k = 4$ corresponding to the boat ($t = 96$ to $t = 120$)

$\mathbf{A}^5_{kr}$ $\mathbf{C}^5_{kr}$ $\mathbf{T}^5_{kr}$

e) Key-region $k = 5$ corresponding to the ship ($t = 96$ to $t = 280$)

Figure 8.8: Extracted key-regions for 280 frames of the *Coastguard* sequence.

resulting key-region forms a non-rigid object. The motion estimation is not able to follow the key-region motion of this complex region, and, as a result, the texture and contour images (in Figure 8.8.c) are blurred.

The extracted key-regions $k = 2$ and $k = 4$ also illustrate the ability of the algorithm to cope with objects appearing and disappearing from the sequence. Key-region $k = 2$ in

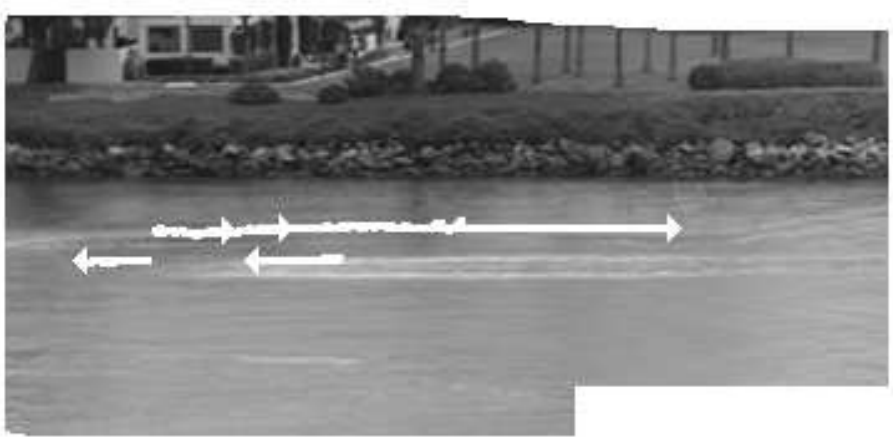


Figure 8.9: Mosaic representing 280 frames of the *Coastguard* video sequence. The trajectories of the extracted key-regions are added to the mosaic as white arrows.

Figure 8.8.b shows the key-region when the ship is entering the scene. Key-region $k = 4$ in Figure 8.8.d represents the boat going out of the image frame. This can be seen, for instance, in the appearance image $\mathbf{A}_{kr}^{4}$ on Figure 8.8.d where the left pixels of the key-region are less dark as they disappear before than pixels on the right of the boat.

For this example, the key-region grouping creates three different foreground objects that are represented by key-region $k = 1$, key-region $k = 4$ and finally the group of key-regions $k = 2, 3, 4$. In the case of key-regions $k = 1$ and $k = 4$, even if they represent the same foreground object (the boat), the key-region grouping is not able to group them together. This is basically due to the fact that, especially for key-region $k = 4$, the water is also segmented and this has a significant impact on the final weighted histogram of the key-region. The other key-regions are grouped together to represent another foreground object of the scene (the ship). In this case, the boat in the key-region $k = 3$ is not very noticeable and all weighted histograms of the key-regions are similar enough to be grouped representing the same foreground object.

The indexing representation of the *Coastguard* sequence is completed with the background information of the sequence which is shown in Figure 8.9. The superimposed trajectories show the boat and the ship motions through the sequence. As the key-regions corresponding to the boat and the ship are merged into a new one when they cross each other, some of the trajectory of the boat, in the lower part, is missing.

8.2 Mosaic Video Coding

The proposed indexing representation is very attractive as it combines both compression and browsing functionalities into a similar representation. The previous section has demonstrated

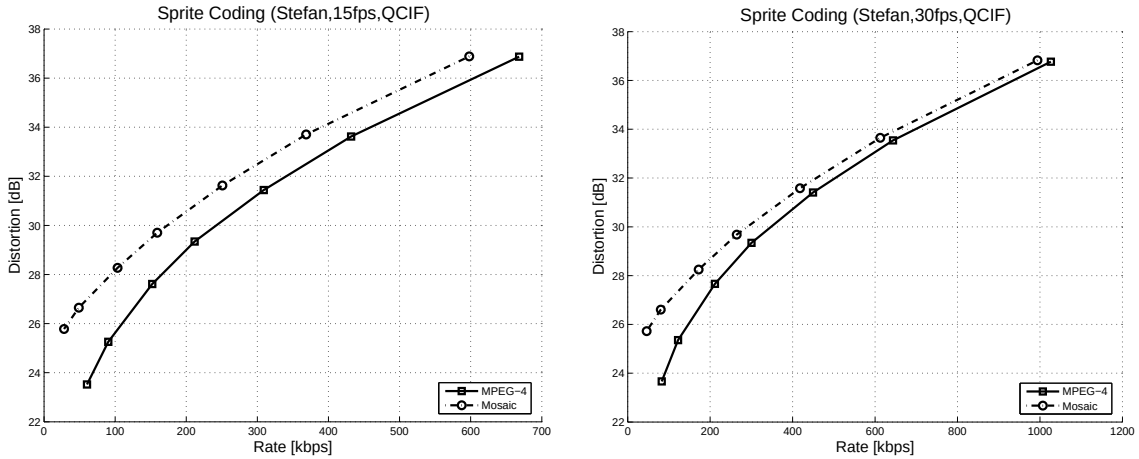
the interest of the representation to describe video shots, this section is devoted to investigating the usefulness of the proposed algorithm for coding. Several techniques in the literature have shown how mosaics may increase the coding efficiency of both the MPEG-4 [16, 57, 89] and the newer H.264 [58] (Section 3.2 provides a detailed state-of-the-art review). For completeness, both application scenarios, when the proposed indexing representation is available to the encoder and decoder and when it must be streamed with the content are studied in this section.

8.2.1 Mosaic available in the Decoder

The scenario where the proposed indexing representation, in this case the mosaic and key-regions, is available for both the encoder and decoder assumes that, before the encoding of the video content, a searching step is performed. During this search phase, the user is able to access the indexing representations associated with the content. This allows the user to benefit from all the functionality provided by the indexing representations such as navigation, search & retrieval, summarization, etc. In the mosaic coding scheme case, the important part of the search step is that, after the user has selected the interesting content, a copy of the indexing representation used in the search is stored locally. Therefore, before the transmission of the content itself is performed, the decoder has already access to the indexing representation. The results on this section assume that the user has obtained the mosaic and key-regions, in order to browse, for instance, a set of videos. The test sequences *Stefan*, *Coastguard* and *NHKvideo7* are analyzed in order to study the benefits of using the mosaic in the compression representation.

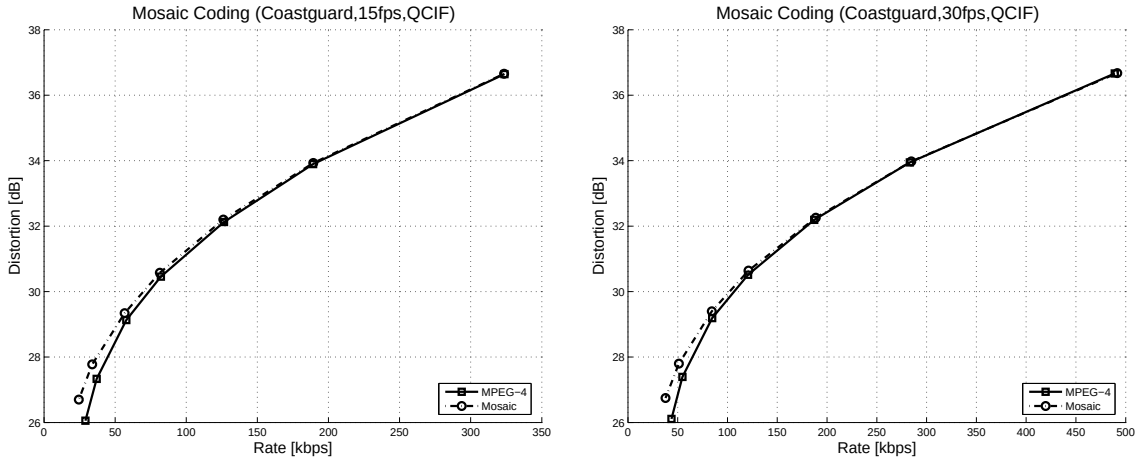
Figure 8.10 shows the rate-distortion curves comparison for the sequence *Stefan* of the MPEG-4 video codec (using the MoMuSys reference software [40]) at both 15 and 30 fps. The filled square marked line denotes the standard baseline MPEG-4 coding. The dotted circle marked line represents the MPEG-4 static mosaic coding scheme. This later scheme uses the *Stefan* mosaic in Figure 6.1 employed in the indexing representation as a reference when coding the test sequence. If the mosaic is used in the static coding scheme, the same mosaic image is used as reference for all bitrates. Therefore, Figure 6.1 shows how static mosaic coding presents higher efficiency (up to 4 dB at 15 fps) at lower rates. However, higher bitrates result in smaller gains (around 1 dB at 15 fps and 0.5 dB at 30 fps).

Figure 8.11 shows a comparison of the MPEG-4 static mosaic coding for 280 frames of the *Coastguard* sequence at 15 fps and 30 fps. The mosaic used for these results is represented on Figure 8.9 (without the super-imposed trajectories). In this sequence, the gain when using the mosaic decreases to 0.8 dB at lower bitrates (and almost nothing at higher bitrates). This is mainly due to the effect of the water in the sequence. Water is very textured and moves through the sequence. The mosaic is not able to capture this motion and therefore the coding efficiency of the mosaic coding scheme in the *Coastguard* sequence is lower.



a) 15 fps b) 30 fps

Figure 8.10: Rate-distortion curve of the baseline MPEG-4 and the static mosaic coding for 240 frames of the sequence *Stefan* at QCIF resolution.

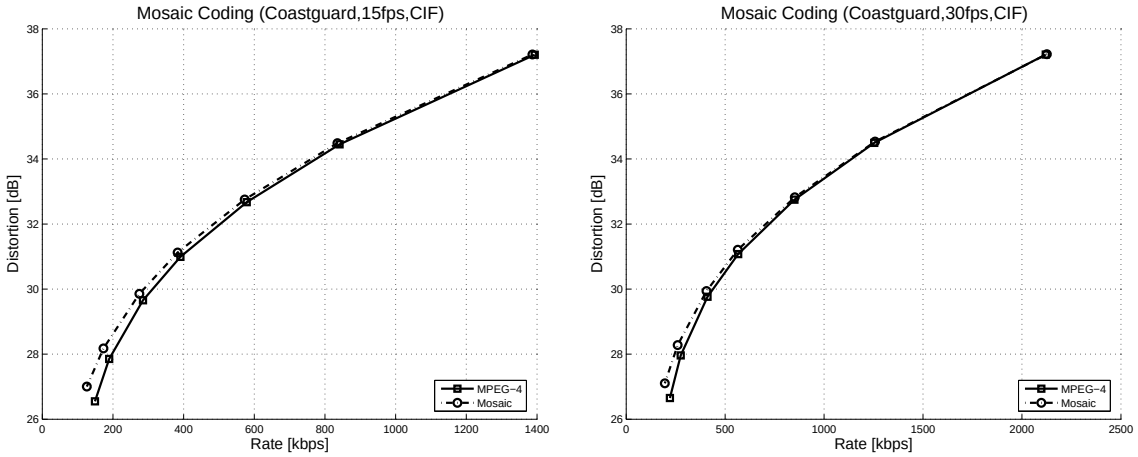


a) 15 fps b) 30 fps

Figure 8.11: Rate-distortion curve of the baseline MPEG-4 and the static mosaic coding for 280 frames of the sequence *Coastguard* at QCIF resolution.

To evaluate the results at different resolutions, the same tests were run using the CIF *Coastguard* version. The results are very similar to the QCIF resolutions. In this case, the mosaic has also been generated from the CIF sequence. Figure 8.12 shows the rate-distortion curves for the CIF sequence at 15 fps and 30 fps.

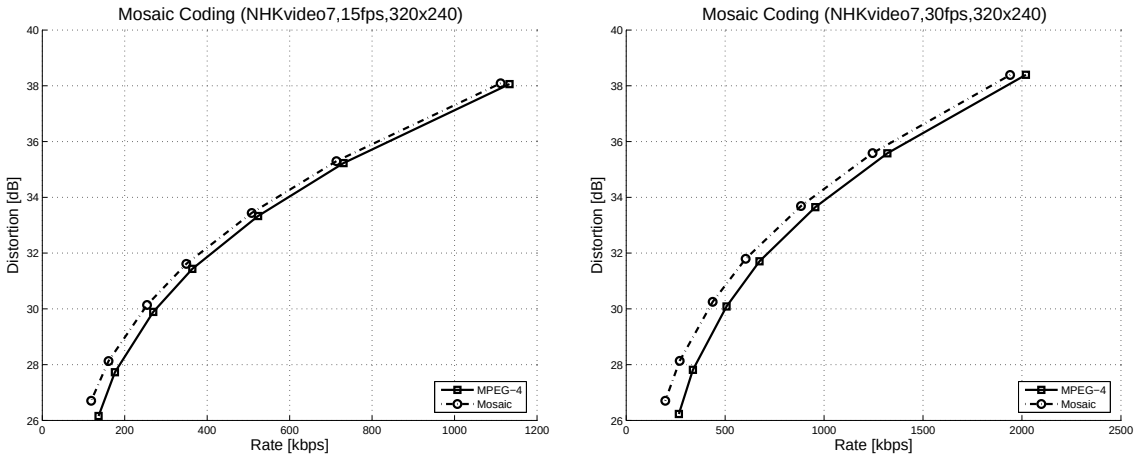
Finally, the *NHKvideo7* sequence is tested. Again, the mosaic (in Figure 8.3) is used as the static mosaic in the MPEG-4 static mosaic coding scheme. Results in Figure 8.13 show how using the mosaic increases the efficiency of the baseline MPEG-4 coding up to 1.8 dB for



a) 15 fps b) 30 fps

Figure 8.12: Rate-distortion curve of the baseline MPEG-4 and the static mosaic coding for 280 frames of the sequence *Coastguard* at CIF resolution.

this specific sequence.



a) 15 fps b) 30 fps

Figure 8.13: Rate-distortion curve of the baseline MPEG-4 and the static mosaic coding for 100 frames of the sequence *NHKvideo7*.

For all the above results, the difference in coding efficiency of using the morphological enhanced mosaic proposed in this thesis or the standard MPEG-4 mosaic is not noticeable. This is due to the fact that the morphological filters are more suited for a browsing representation. Thus, for instance, foreground regions are better removed from the mosaic. As these regions are not well represented in the mosaic, the coding efficiency of these regions is not as good as using the standard MPEG-4 mosaic. For background regions, both mosaics are

similarly good when used as references to encode the video sequence.

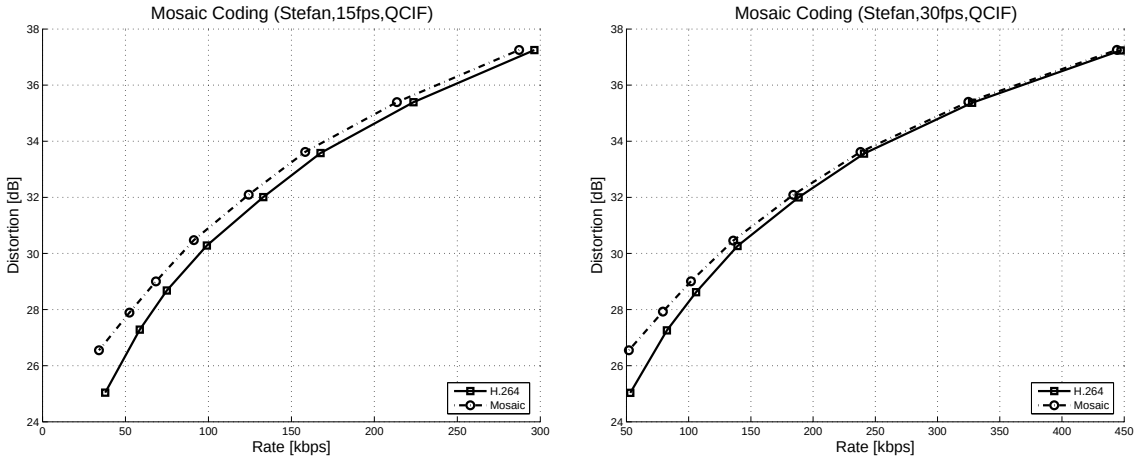
For completeness, the mosaic coding scheme is also studied in the H.264 video codec. As the H.264 standard does not include a mosaic coding scheme, the codec was extended to support the static mosaic coding scheme. The extension is made by modifying the long term prediction buffer, used in the long term temporal prediction scheme (also known as multi-frame prediction) of the H.264 standard. The long term prediction buffer is filled with the N previously encoded frames closest in time (with usually $N = 5$). Images in the long term prediction buffer are used as references to encode the original frames. In the case of the static mosaic extension, the long term prediction buffer is extended to support a new reference frame. This time, however, the new reference is not extracted from previous coded frames but from the current background information $\mathbf{I}_b(t)$ (computed from the mosaic and the warping parameters $\mathbf{g}(t)$ such as in Section 7.1). As with the MPEG-4 static mosaic coding scheme, the mosaic extended H.264 also needs the mosaic and the warping parameters at the decoder end. If the scenario where the indexing representation is assumed to be available to the decoder, similar results as the MPEG-4 mosaic coding scheme are obtained. The JM-6.0a [42] reference software implementation is used for the results of the H.264 video codec. Table 8.1 summarizes the settings for the H.264 video codec in all the experimental results.

Setting	**Status**
Intra period	0
Hadamard transform	on
Arithmetic coding (CABAC)	on
Error resilience	off
RD mode decision	on
MV search range	± 16
Variable size motion modes	all
Motion pixel accuracy	1/8

Table 8.1: Basic H.264 settings for experimental results on mosaic video coding.

Figure 8.14 shows the results for the sequence *Stefan* at 15 and 30 fps and QCIF resolution. As the H.264 codec is more advanced (with several more coding block modes) the efficiency of using the mosaic is not as high as in the MPEG-4 codec. In this codec, the mosaic is used as an extra reference in the long term prediction buffer and, therefore, in the case of high bitrates, the previous encoded frames of the sequence are enough to be a very good reference. In any case, gains are still achieved for low bitrates (up to 1.8 dB).

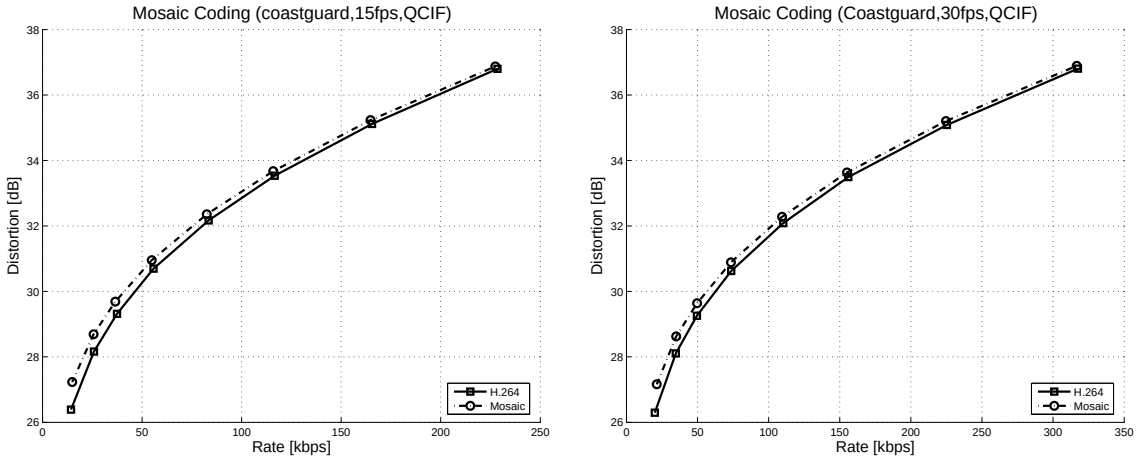
For the *Coastguard* sequence the gain is greater than the one obtained with the MPEG-4 codec. Figure 8.15 and 8.16 represent the rate-distortion curves comparing the H.264 video codec and the mosaic extended H.264 for the *Coastguard* sequence at QCIF and CIF



a) 15 fps b) 30 fps

Figure 8.14: Rate-distortion curve of the H.264 and the mosaic extended H.264 for 240 frames of the sequence *Stefan* at QCIF resolution.

resolutions respectively. The mosaic extended H.264 codec uses 5 previously encoded frames and the mosaic as references in the long term prediction buffer while the mosaic coding scheme on the MPEG-4 codec only uses the mosaic as reference (and not the previously encoded frames). This makes the possibility of selecting previous frames or the mosaic as reference allowing for a better efficiency in the final coding of the sequence.



a) 15 fps b) 30 fps

Figure 8.15: Rate-distortion curve of the H.264 and the mosaic extended H.264 for 280 frames of the sequence *Coastguard* at QCIF resolution.

Finally, the mosaic extended H.264 codec is used in 100 frames of the sequence *NHKvideo7*. Figure 8.17 shows how using the mosaic as reference is not as efficient as in the MPEG-4 codec

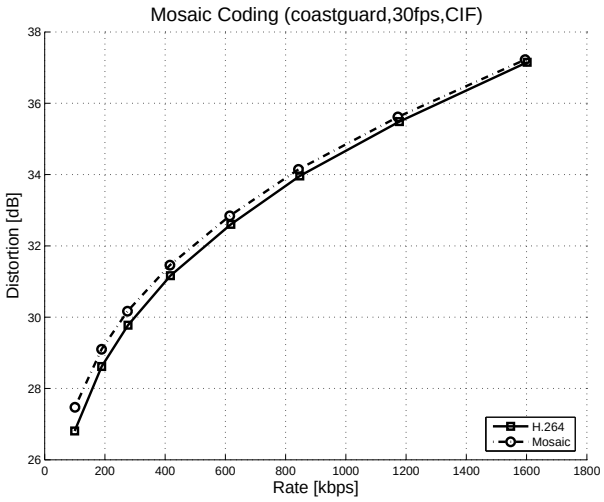
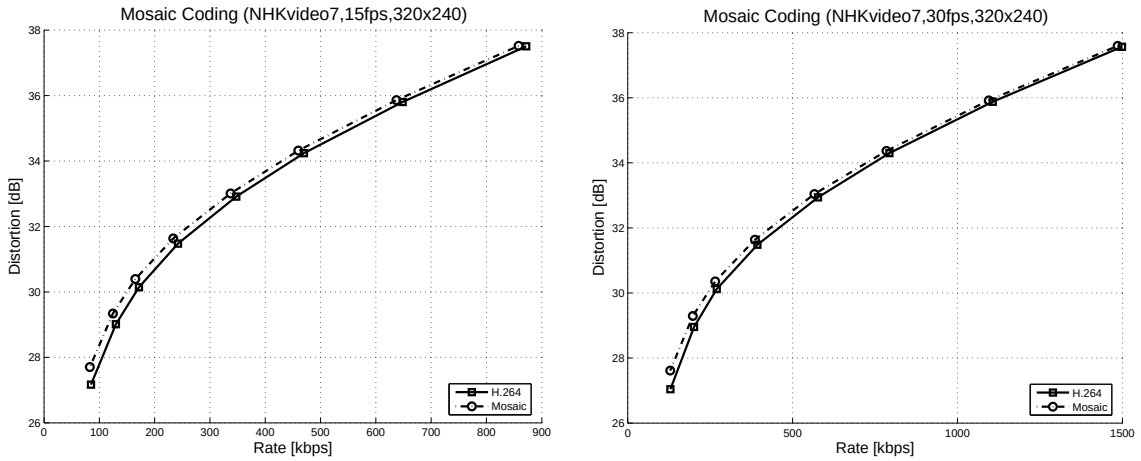


Figure 8.16: Rate-distortion curve of the H.264 and the mosaic extended H.264 for 280 frames of the sequence *Coastguard* (CIF resolution and 30.0 fps).

in Figure 8.13. As expected, the efficiency of using the mosaic in the H.264 video codec is lower than in the MPEG-4 codec. In this case, gains of 0.5 dB are achieved for low bitrates.



a) 15 fps b) 30 fps

Figure 8.17: Rate-distortion curve of the H.264 and the mosaic extended H.264 for 100 frames of the sequence *NHKvideo7*.

As a conclusion, results in this section have shown that if the mosaic is available at both encoder and decoder, it can be used for both MPEG-4 and H.264 encoders to increase the coding efficiency of the compression representation. Results have shown that, as the same mosaic can be used as a reference at all bitrates, the efficiency gains are concentrated on low bitrates.

8.2.2 Mosaic not available in the Decoder

In the case where the indexing representation is not available to the decoder, it must be streamed together with the content. As only the mosaic and the warping parameters are needed in the decoder, the key-regions do not need to be streamed to the decoder. In order to transmit the mosaic, several encoding strategies may be used. In the results of this section, two different encoding strategies are used to encode the mosaic. The first one is using the static mosaic coding provided by MPEG-4 where the mosaic is coded as an $I-VOP$ frame with an alpha map. The second uses the more recent standard JPEG2000 with ROI coding [31, 98]. The ROI is needed as, usually, the mosaic is not perfectly squared. The warping parameters are encoded using the trajectory encoding strategy of the MPEG-4 static mosaic coding scheme. In the strategy, motion vectors of the corner positions of the mosaic are differentially encoded and transmitted [68].

Table 8.2 shows the needed bits for coding the mosaics and warping parameters of the previous test sequences. As the mosaic is the same at all bitrates and only sent at the beginning of the sequence, the final rate depends on the total number of frames of the sequence. Longer sequences are able to use the mosaic as a reference during more frames and, therefore, the total bitrate can be further reduced.

Sequence	Resolution	MPEG-4 (bits)	JPEG2000 (bits)	Parameters (bits/frame)
Stefan	QCIF	165432	152288	24.02
Coastguard	QCIF	51904	48592	21.72
Coastguard	CIF	153008	143376	23.39
NHKvideo7	320x240	176152	151952	30.31

Table 8.2: Extra encoding bits needed for mosaic and warping parameters.

The Figures 8.18, 8.19 and 8.20 compare the rate-distortion curves for the baseline MPEG-4 video codec and the static mosaic coding scheme of MPEG-4. This time, however, it is not assumed that the mosaic is available to the decoder, therefore, it is streamed together with the content. The mosaic is encoded using the MPEG-4 static mosaic coding (the needed bits for each test sequence are indicated in the third column of Table 8.2).

As the same mosaic and warping parameters are encoded and transmitted at low and high bitrates, the number of bits for the streamed mosaic is the same for all points of the rate-distortion curve. This can be seen as a translation to the right of the rate-distortion curve as more rate is needed to achieve the same distortion level. The test sequences analyzed show that, for the MPEG-4 codec, even if the mosaic needs to be streamed, the resulting rate-distortion efficiency of using the mosaic image is still higher than the baseline encoding.

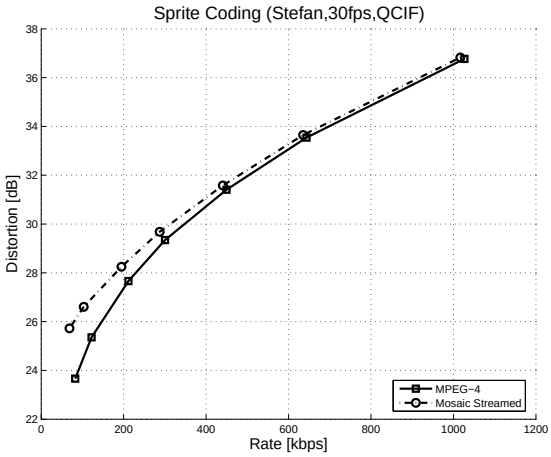
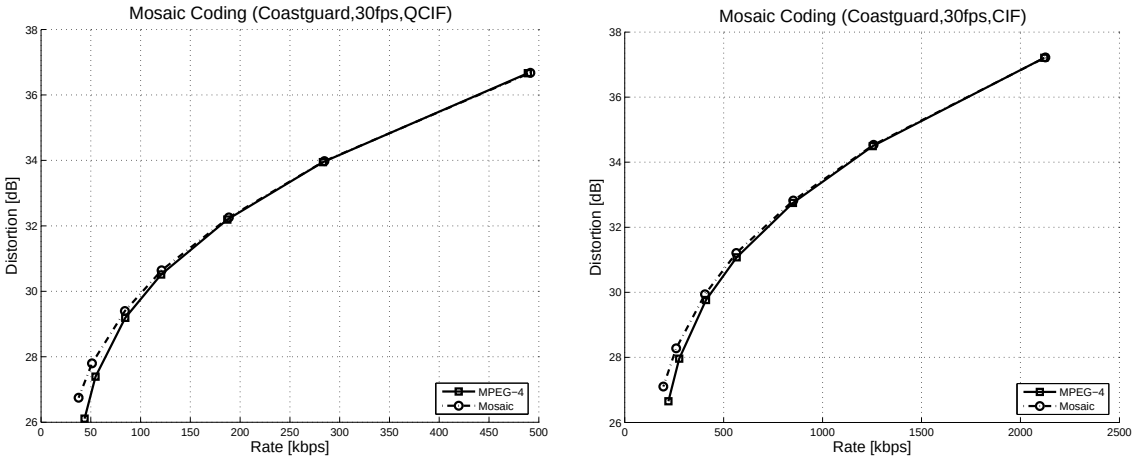


Figure 8.18: Rate-distortion curve of the baseline MPEG-4 and the static mosaic coding (streamed in the bitstream) for 240 frames of the sequence *Stefan* at QCIF resolution and 30 fps.



a) QCIF b) CIF

Figure 8.19: Rate-distortion curve of the baseline MPEG-4 and the static mosaic coding (streamed in the bitstream) for 280 frames of the sequence *Coastguard* at QCIF and CIF resolution and 30 fps.

For instance, Figure 8.18 shows that the rate-distortion curve of the streamed mosaic is still able to obtained a higher efficiency that the baseline encoding.

However, for the test sequence *NHKvideo7*, streaming the mosaic image with the content obtains a very similar rate-distortion efficiency to not using the mosaic. The rate-distortion curves on Figure 8.19 show that, especially at high bitrates, the efficiency of coding the test sequence using, and streaming, the mosaic is basically the same as the baseline encoding.

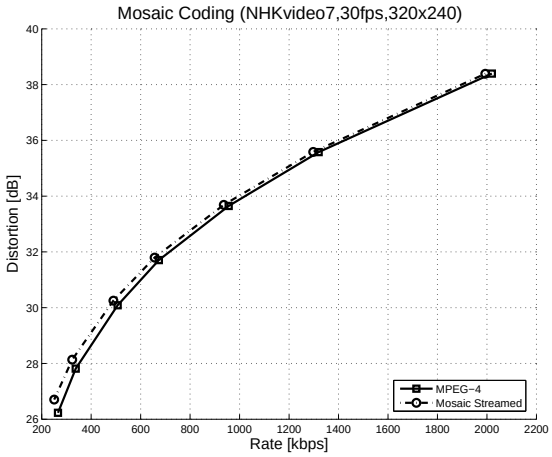
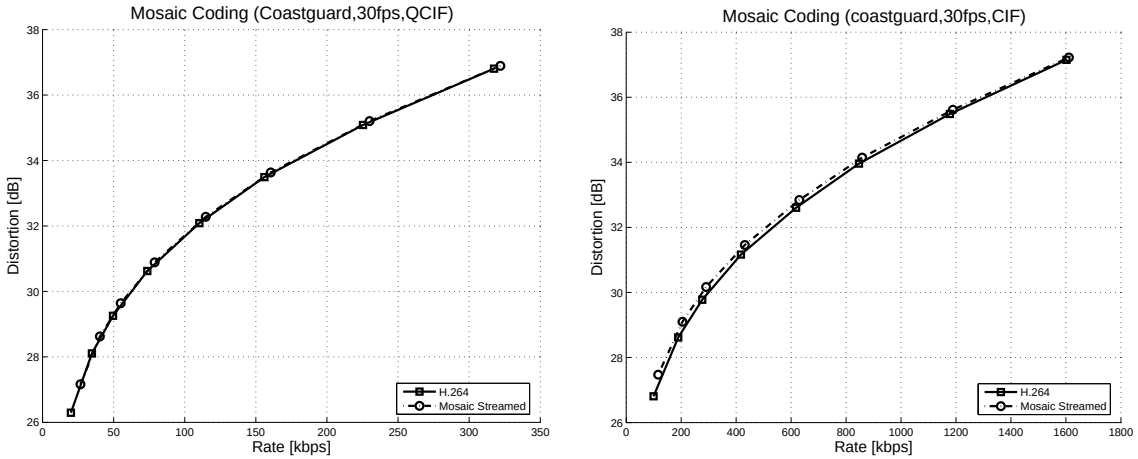


Figure 8.20: Rate-distortion curve of the baseline MPEG-4 and the static mosaic coding (streamed in the bitstream) for 100 frames of the sequence *NHKvideo7*.

The same application scenario is studied for the H.264 codec. Figures 8.21, 8.22 and 8.23 compare the rate-distortion curves of the standard H.264 codec versus the mosaic extended H.264. Again, the scenario where no indexing representation is available to the decoder is assumed, hence, the mosaic and warping parameters are streamed to the decoder. In this scenario, the mosaic is encoded using a JPEG2000 with ROI coding scheme while the warping parameters are encoded following a trajectory coding scheme (the same as the static mosaic coding scheme in MPEG-4). The number of bits needed for each test sequence is detailed in the fourth and fifth columns of Table 8.2.



a) QCIF b) CIF

Figure 8.21: Rate-distortion curve of the H.264 and the mosaic extended H.264 (streamed in the bitstream) for 280 frames of the sequence *Coastguard* at QCIF and CIF resolution and 30 fps.

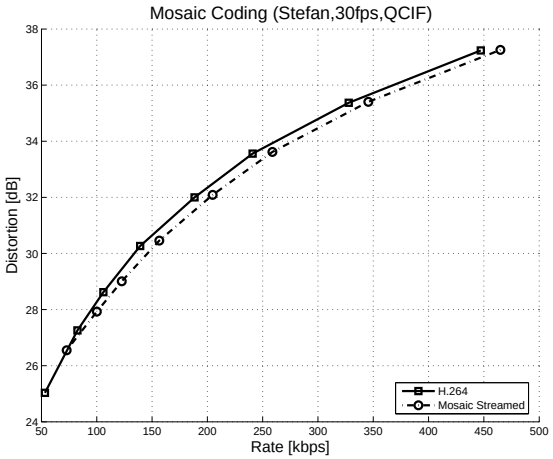


Figure 8.22: Rate-distortion curve of the H.264 and the static mosaic coding enabled H.264 (streamed in the bitstream) for 240 frames of the sequence *Stefan* at QCIF resolution and 30 fps.

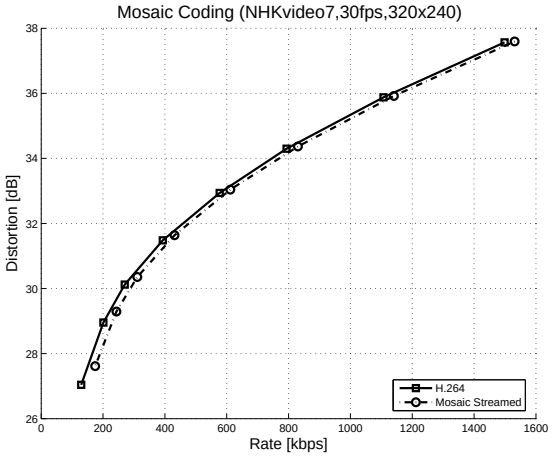


Figure 8.23: Rate-distortion curve of the H.264 and the mosaic extended H.264 (streamed in the bitstream) for 100 frames of the sequence *NHKvideo7*.

Figure 8.21 compares the rate-distortion curves for the standard H.264 codec and the mosaic extended H.264 for the sequence *Coastguard*. For this test sequence, even if the mosaic is streamed with the video content, it is still desirable to use the static mosaic coding scheme as it improves the final rate-distortion efficiency of the video codec.

However, for test sequences *Stefan* and *NHKvideo7*, in Figures 8.22 and 8.23 respectively, the standard H.264 codec outperforms the mosaic extended one. The H.264 is considerable more advanced than the MPEG-4 codec and therefore it is already very good exploiting the redundancy of the video sequence. In the case of the *Stefan* sequence, as it is composed of very high motion scenes, the associated mosaic is very large and sending it has a very high

cost in bits (as can be seen on the first row of Table 8.2). This results on a high penalty when the mosaic needs to be streamed with the content. For the *NHKvideo7* test sequence, as it is a short sequence composed of only 100 frames, the amount of bits needed to stream the mosaic turns to be very significant compared to the amount of bits needed to encode the original sequence.

As a conclusion, if the mosaic needs to be sent to the decoder, the final efficiency of the static mosaic coding is obviously reduced (especially for the H.264 video codec). However, using the static mosaic coding scheme can still be very attractive as it integrates the indexing representation into the compression one providing, not only compression functionalities, but also browsing and indexing ones.

Part III

Using Indexing Representations to Improve Compression Representations

Chapter 9

Motivation

9.1 Introduction

As the amount of digital video material grows, multimedia services need to consider the indexing and browsing aspects of the content in order to efficiently manage it. Therefore, it is expected that most of the multimedia material will be already indexed to facilitate functionalities of browsing, summarization, search, retrieval, etc. Due to the need of sharing these indexing representations between content providers, users, etc. several standards, such as MPEG-7 or SMPTE, have proposed a framework to describe and represent audio-visual material.

This part of the thesis is devoted to study the improvement of compression representations by using the already existing indexing representations. The novelty of this approach is to use indexing representations which has been extracted with other functionality in mind (indexing, browsing, etc.) in order to improve compression representations. Using any kind of extra information to improve coding has been used, and it is still being used, in coding for many years, for instance, the use of motion vectors. The fact that indexing representations are already used for other purposes, other than coding, and that they might be available at both encoder and decoder sides, makes them a nice candidate to exploit.

Among all possible indexing representations, only a small fraction of them may be useful for improve video coding. In this thesis, only the MPEG-7 and SMPTE standards have been considered. Chapter 10 makes a review of the MPEG-7 and the SMPTE standards commenting which indexing representations proposed by the standards may have a greater potential to be used by compression representations.

The rest of the chapters in this part of the thesis are devoted to show how indexing representations can be exploited to create indexing representation based video codecs. Four different schemes are presented to prove the ability of standard indexing representations to improve video coding. Chapter 11 studies the use of the MPEG-7 *Global Transition* DS

that describes the shot transition of a video sequence. The inter-frame prediction during the transition can exploit the modeling of the transition performed by the indexing representation. For instance, the video codec may use the knowledge that previous frames get darker (or brighter) in a fade-out (or fade-in).

Chapter 12 reviews how long term temporal prediction can be improved using indexing representations. Long term prediction introduced the idea of using N reference frames to predict the blocks of the current image instead of using only one reference (usually the previous one). It will be shown how a simple indexing representation, such as the MPEG-7 *Color Layout*, can make a pre-selection of N reference frames that better predict the current frame being coded.

Chapter 13 studies a high level descriptor, such as the MPEG-7 *Video Segment* DS, to create a new coding order for the entire video sequence that better exploits the temporal redundancy. The descriptor creates a hierarchical decomposition of the video sequence that is used to create a shuffled video sequence which can be compressed more efficiently.

Finally, Chapter 14 reviews a frame type selection scheme using indexing representations. For this scheme, the encoder control is able to select optimum GoP parameters based on information extracted from the indexing representations. The chapter will show how motion descriptors, such as the MPEG-7 *Motion Activity* D, can help in the decision of determining an optimum GoP structure for video segments.

All four indexing representation based video codecs are presented following the same structure. Each chapter is divided into 5 sections. In the first section, a motivation of the proposed scheme is described. In the second section, a description of the proposed coding scheme which exploits indexing representations is detailed. The third section is devoted to presenting the indexing representations selected in the coding scheme. The fourth and fifth sections show the experimental results of the proposed schemes for different test sequences. The fourth section focuses on the scenario where the indexing representations are available for both the encoder and decoder while the fifth section focuses on the scenario where the indexing representations are not available at the decoder side and must be streamed together with the content. All results are based on the H.264 video codec (reference software version JM-6.0a [42]). In the experimental tests, the reference software has been extended to be able to read and exploit indexing representations.

Chapter 10

Useful Indexing Representations for Coding

10.1 Introduction

Within the large amount of descriptors defined by the MPEG-7 and SMPTE standards, only a fraction of them may have the potential to improve the coding efficiency of current compression representations. Among all the MPEG-7 and SMPTE descriptors, the MPEG-7 Content Description and the SMPTE Class 3, 5 and 7 are the subsets that more clearly present a potential for video coding. These subsets contain indexing representations describing the video signal characteristics. This information about the signal (such as color, motion, structure, etc.) can be exploited in the coding process.

The MPEG-7 and SMPTE descriptors can be distributed in 8 different groups depending on the functionality provided. In this chapter, each group is analyzed and its potential for video coding is reviewed.

10.2 MPEG-7 and SMPTE Potential for Video Coding

- **MPEG-7 Basic Elements**

  Basic Elements descriptors define building blocks (such as data types, arrays, matrices, etc.) that are used for other Ds and DSs. These blocks are not used to describe the content itself but are used by other descriptors. Therefore, they are not appropriate for improving coding efficiency.

- **MPEG-7 Content Management and SMPTE Classes 1, 2 and 4**

  Descriptors defined in this group provide high-level information about the creation and usage of the content they describe. Some of them describe how the content has been

coded but this information cannot be used to modify the encoding or decoding processes.

- **MPEG-7 Navigation and Access**

  Descriptors in this group provide information for facilitating browsing and retrieval by defining summaries, views, and variations of the multimedia content. These DSs are very application dependent, hence, a common strategy for coding is difficult to create.

- **MPEG-7 User Interaction**

  These descriptors describe the user preferences. They do not describe any content and therefore are not useful for coding.

- **SMPTE Class 6**

  These descriptors define relations between other indexing representations. They are too high level in order to be used for improving indexing representation based video codecs.

- **SMPTE Classes 14, 15 and 16**

  Descriptors included in these classes define proprietary, undisclosed descriptors or information for future descriptors. They cannot be used to improve coding efficiency.

- **MPEG-7 Content Organization**

  These DSs describe the organization of multimedia content. This organization can help indexing representation based encoders in order to locate similar content on a large amount of multimedia material. In particular, the *Collection Structure* description scheme can be useful to detect the sequences where similar shots and frames to the one being coded can be found.

- **MPEG-7 Content Description and SMPTE Class 3, 5 and 7**

  This is where most of the useful Ds and DSs can be found. This part of the description is particularly useful for compression because it involves a large number of indexing representations that deal directly with the signal characteristics. The following list enumerates the different features described by the indexing representations included in these groups together with a possible use for encoding:

  1. **Color**
     - Descriptors: MPEG-7 *Color Space*, *Color Quantization*, *Dominant Color*, *Scalable Color*, *Color Layout*, *Color Structure*, SMPTE Video Color Type, Color Range Levels, etc.
     - These descriptors provide information about the color distribution of the pixels. Although the description gives an approximation of the pixel probability density function, it is not precise enough to allow any improvement of the

entropy coding. However, they can be used to search and retrieve images or video segments. Chapter 12 will show how these descriptors, in particular the MPEG-7 *Color Layout* D, can be used to select reference frames for hybrid video codecs [23, 24].

2. **Texture**
   - Descriptors: MPEG-7 *Homogeneous Texture*, *Texture Browsing*, *Edge Histogram*, etc.
   - These descriptors, in particular the MPEG-7 *Edge Histogram* D, can also be used to indicate the presence of textured areas. These areas can be omitted during the encoding process and synthesized at the receiver [70, 71]. These descriptors are also appropriate to tune the encoder parameters [23].
3. **Motion**
   - Descriptors: MPEG-7 *Camera Motion*, *Motion Trajectory*, *Parametric Motion*, *Motion Activity*, SMPTE Video Motion Type, Speed Change Effect, etc.
   - These descriptors can also be used to help compression. For instance, the *Parametric Motion* descriptor can be shared with the global motion descriptor computed to build mosaic images [90, 91] (Section 6). Motion information can help to efficiently create a bit assignment strategy. An example using the MPEG-7 *Motion Activity* descriptor will be given in Chapter 14 [23].
4. **Face**
   - Descriptors: MPEG-7 *Face Recognition*.
   - This descriptor defines a subspace appropriate for face identification and recognition. It specifies a transform that could be used for face encoding. The eigen-faces provided by the indexing representation generate an eigen-space that is used to project the face to be coded. The projection, together with a possible error image, is sent to the decoder which uses the same MPEG-7 eigenspace to recreate the original face. However, attempts using directly the MPEG-7 *Face Recognition* D for encoding have not been successful [76].
5. **Structural Aspects**
   - Descriptors: MPEG-7 *Segments*, SMPTE Data Relationships.
   - These descriptors define the structure of the content. This information can be used to improve the compression efficiency, as shown in Chapter 13, where similarity between video segments or frames is extracted from the structural description of the content to reorganize the sequence to be coded [23, 24].
6. **Shot Transition**
   - Descriptors: MPEG-7 *Analytic Transition*, SMPTE Default Fade Duration.

– The MPEG-7 includes a specific tool to describe the sequence structure at the level of shot transition. This information will be used in Chapter 11 to improve the compression of frames belonging to a transition.

In this thesis, only the MPEG-7 descriptors have been employed. Even though both SMPTE and MPEG-7 address similar problems and often share some of the same characteristics with the same goal in mind, the MPEG-7 standard was chosen over SMPTE for various reasons. First of all, the implication of the author and the research group in some of the MPEG-7 proposal (Section 4.2) made this standard more accessible and easier to follow. Secondly, MPEG-7 is more extended and more focused on low level descriptions that deal with the audiovisual characteristics of the signal and, therefore, they present more potential for the goals of this thesis. Finally, MPEG-7 also supplies an extensive explanation of the extraction and exploitation of the standard Ds and DSs which, even being non normative, provides a very helpful starting point to understand and implement the involved descriptors.

Chapter 11

Transition Coding

This chapter describes the use of indexing representations to encode video transitions. The MPEG-7 *Global Transition* DS that describes the shot transition of a video sequence is exploited to enhance the inter-frame prediction during the coding of the transition.

11.1 Motivation

Transitions between video scenes are common in standard video sequences [5]. The coding of such transitions is a very challenging problem for current video coding methods. This is because the traditional block based motion model cannot efficiently represent the transition motion. Normally, video shot transitions are categorized into two types: static shot transitions and dynamic transitions. In static shot transitions the video signal keeps its original size and orientation while the pixel amplitude changes. Therefore, no additional motion is introduced in the static transition. Typical static transition are dissolves, fades, wipes, etc. In dynamic shot transitions, the original video signal is modified, hence, its orientation, size or rotation may be different during the transition. Typical transitions in this category include geometric transformations, page curling, etc.

Figure 11.1: Example of a static dissolve transition between two shots. The fade between the two shots can be modeled using a linear (solid line) or non-linear (dotted line) function.

Figure 11.1 shows an example of a dissolve transition between two different shots. The

fade between shots can be modeled using linear or non-linear functions that represent the weights of forward and backward frame interpolation. Recent indexing representation standards have proposed some descriptors to describe the evolution of the transition between different shots. For instance the *Analytic Transition* DS and the *Global Transition* DS of the MPEG-7 standard store information about the start and end of the transition, the associated weights of the transition functions, etc.

These indexing representations can be used to improve the coding within transitions. For instance, the video codec may use the knowledge that frames get darker (or brighter) in a fade-out (or fade-in) transition. The motion estimation of the encoder can also be tuned, thus, it follows the transition model defined by the indexing representation. In the case of dynamic transitions geometric transformations can be applied to the video frames to better model the transition motion.

In this section, the standard hybrid encoding technique is improved to make use of existing MPEG-7 indexing representations in order to improve the efficiency when coding transitions. Several modules of standard hybrid codecs need to be modified to exploit the information already present in the indexing representations. A diagram of a hybrid video codec is shown in Figure 11.2. The modules modified by the proposed indexing representation enhanced scheme are represented in dark gray.

The coder control module is responsible for the GoP structure decision and high level decision on the coding of video shots. In the case of transition coding, the standard GoP structure can be modified to fit the transition. The motion estimation and compensation modules are also modified, hence, the standard interpolative mode of B frames is changed to be able to use different weights for previous and past references (instead of the fixed 0.5 weights for each reference). The term $\overline{B}$ will be used as the modified version of B frames able to deal with different weights for previous and past references and a joint motion estimation.

As the motion estimation and compensation blocks are modified, the resulting bitstream of an indexing representation based H.264 codec is not standard anymore and the decoder must also be notified to cope with the improved motion compensation module. Recent versions of the H.264 video codec (from version 7.0 [43]) implement a weighted motion estimation where different weights can be used in the motion estimation step [6]. If that version is used, the motion compensation module does not need to be modified and different weights can be used in the interpolative mode of B frames, and, therefore, no $\overline{B}$ frames are needed.

11.2 Indexing Representation Enhanced Coding Scheme

The proposed indexing representation enhanced codec for transition coding is composed of two basic steps. The first step (in the coder control module) modifies the GoP structure to better code the shot transition. In the second step, the motion estimation and compensation

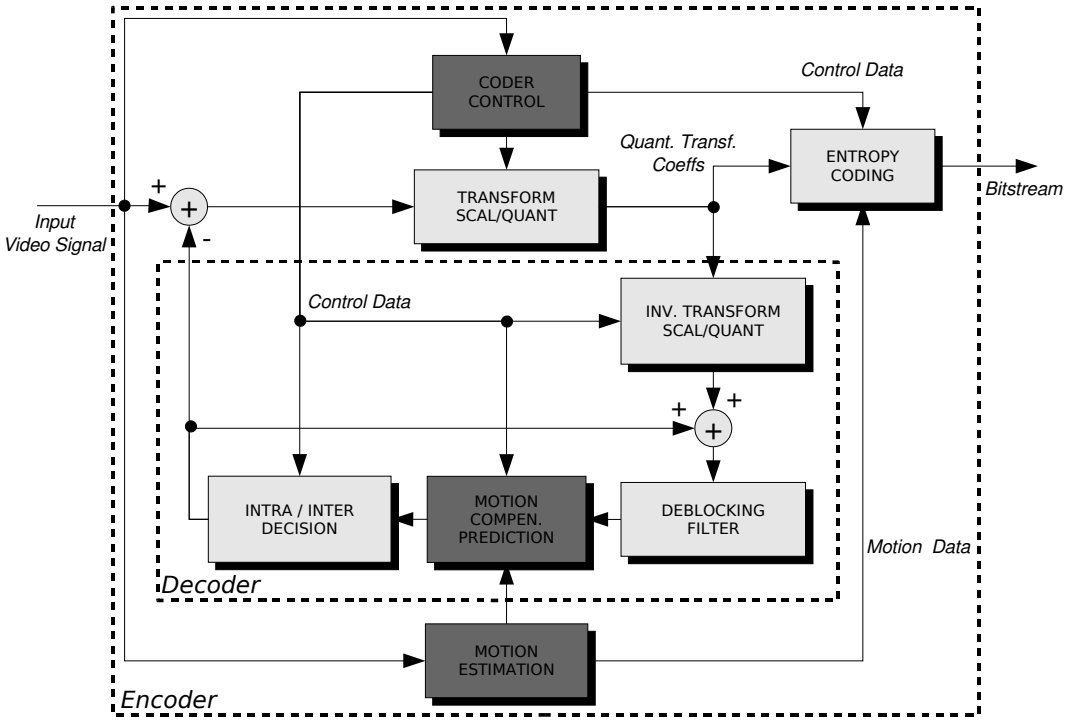


Figure 11.2: Indexing representation enhanced H.264 coding scheme. In dark gray the modules that are modified in order to exploit the indexing representation when coding transitions.

modules are changed, hence, the weights can be used in a joint motion estimation mode.

If it is assumed that the visual information during a transition usually depends on both previous and past shots then, the B frame type can be exploited to improve the coding efficiency within the transition. For that reason, the last frame of the transition is forced to be encoded as an I frame, while the frames within the transition are encoded as B frames. The start and end of the transition is known as it is described in the indexing representation. Using only B mode for frames in the transition has the advantage that current frames can be predicted using two references both at the beginning and at the end of the transition. In the second step, a new weighted interpolative mode of B frames is introduced in the video codec.

However, using only the weights indicated by the indexing representation is not enough. In order to precisely exploit the indexing representation weights, a new motion estimation scheme is needed. In standards such as MPEG-4 and H.264, the interpolative mode is performed using the motion vectors that have been estimated on the previous and past references separately. When using different weights for the previous and the past references, if the estimation is done separately, it turns out to be rather inefficient, as for instance, when the

weight associated to a reference is low, the motion estimation is extremely noisy. To solve this problem, the interpolative mode of $\overline{B}$ frames within the transition estimates both the forward and backward motion vectors in a joint motion estimation scheme taking into account the indexing representation weights.

Let us denote by $\mathbf{B}$ a block in the current frame (inside the transition) which is to be estimated using both reference images at the start, represented by $\mathbf{R}^b$, and at the end of the transition, represented by $\mathbf{R}^f$ and the indexing representation weights, denoted by w_b and w_f for the backward and forward reference frames respectively. In order to estimate the motion vectors, the criterion that needs to be minimized is, usually, the sum of absolute differences (SAD) between the current block $\mathbf{B}$ and the predicted block $\mathbf{P}$:

$$min\left(\sum_{\forall(x,y)} |\mathbf{B}(x,y) - \mathbf{P}(x,y)|\right) = min\left(SAD(\mathbf{B},\mathbf{P})\right) \tag{11.1}$$

If the motion vectors are estimated separately, two different SAD are computed. First, the backward motion vector $\mathbf{mv}^b$ is obtained by minimizing the $SAD(\mathbf{B},\hat{\mathbf{R}}^b)$ where $\hat{\mathbf{R}}^b = \mathbf{R}^b(\mathbf{mv}^b)$ is the predicted block in the reference image $\mathbf{R}^b$ using the motion vector $\mathbf{mv}^b$. Second, the forward motion vector $\mathbf{mv}^f$ is estimated by minimizing $SAD(\mathbf{B},\hat{\mathbf{R}}^f)$ where, this time, the predicted block is obtained from the forward reference frames $\hat{\mathbf{R}}^f = \mathbf{R}^b(\mathbf{mv}^b)$. The final motion vectors that are used in the interpolative mode are $\mathbf{mv}^b$ and $\mathbf{mv}^f$ computed separately. However, in the joint motion estimation, the motion vectors are found by minimizing a unique $SAD(\mathbf{B},\hat{\mathbf{P}})$ where, this time, the predicted block $\hat{\mathbf{P}}$ is directly computed using both backward and forward motion vectors and the indexing representation weights, $\hat{\mathbf{P}} = w_b \cdot \mathbf{R}^b(\mathbf{mv}^b) + w_f \cdot \mathbf{R}^f(\mathbf{mv}^f)$.

As computing a pair of motion vectors using a joint full search approach is computationally very expensive, an intermediate approach using a 2D-log search algorithm [36, 45] is used in the proposed indexing representation scheme. The 2D-log search algorithm is introduced in the interpolative motion estimation of $\overline{B}$ frames in two steps. In the first step, a combination of a full search in the backward reference frame $\mathbf{R}^b$ and a 2D-log search on the forward reference frame $\mathbf{R}^f$ is performed. This combination finds the motion vectors $(\mathbf{mv}^b_{min1}, \mathbf{mv}^f_{min1})$ with the minimum prediction error. In the second step, a 2D-log search on the backward reference and a full search on the forward reference are used to obtain the motion vectors $(\mathbf{mv}^b_{min2}, \mathbf{mv}^f_{min2})$. The final motion vectors selected for the interpolative mode of a block $\overline{\mathbf{B}}_i$ of the $\overline{B}$ frame $(\mathbf{mv}^b_i, \mathbf{mv}^f_i)$, are the ones with minimum prediction error between the previous two steps. Figure 11.3 shows the pseudo-code algorithm followed to compute the interpolative motion vectors $(\mathbf{mv}^b_i, \mathbf{mv}^f_i)$ for each block $\overline{\mathbf{B}}_i$ of the current frame. In the algorithm, w_b and w_f denote the indexing representation weights of the transition.

The number of candidate blocks to search is reduced using a 2D-log motion estimation. If sr denotes the motion estimation search range in pixels, the number of candidates goes from

foreach $\overline{\mathbf{B}}_i$ in current frame do
$e_{min1} = e_{min2} = MAXERROR$

foreach mv_1^b to R^b (full search) do
foreach mv_1^f to R^f (2D-log search) do
Estimate block with previous motion vectors:
$\hat{\mathbf{P}}_i = w_b \cdot \mathbf{R}^b(\mathbf{mv}_1^b) + w_f \cdot \mathbf{R}^f(\mathbf{mv}_1^f)$
Compute prediction error:
$e = SAD(\overline{\mathbf{B}}_i, \hat{\mathbf{P}}_i)$
Store motion vector with minimum error:
if $e < e_{min1}$ then
$e_{min1} = e$
$\mathbf{mv}_{min1}^b = \mathbf{mv}_1^b$
$\mathbf{mv}_{min1}^f = \mathbf{mv}_1^f$

foreach mv_2^f to R^f (full search) do
foreach mv_2^b to R^b (2D-log search) do
Estimate block with previous motion vectors:
$\hat{\mathbf{P}}_i = w_b \cdot \mathbf{R}^b(\mathbf{mv}_2^b) + w_f \cdot \mathbf{R}^f(\mathbf{mv}_2^f)$
Compute prediction error:
$e = SAD(\overline{\mathbf{B}}_i, \hat{\mathbf{P}}_i)$
Store motion vector with minimum error:
if $e < e_{min2}$ then
$e_{min2} = e$
$\mathbf{mv}_{min2}^b = \mathbf{mv}_2^b$
$\mathbf{mv}_{min2}^f = \mathbf{mv}_2^f$

if $e_{min1} < e_{min2}$
$\mathbf{mv}_i^b = \mathbf{mv}_{min1}^b$
$\mathbf{mv}_i^f = \mathbf{mv}_{min1}^f$
else
$\mathbf{mv}_i^b = \mathbf{mv}_{min2}^b$
$\mathbf{mv}_i^f = \mathbf{mv}_{min2}^f$

Figure 11.3: Pseudo-code procedure for the joint motion estimation of interpolative motion vectors using indexing representation weights in transitions and a combination of full and 2D-log search.

$(2 \cdot sr + 1)^2$ using a full search algorithm to a maximum of $21 + 2 \cdot \log_2(sr/4)$ using the 2D-log

algorithm. A combined full search estimation (searching all possible motion vectors of the backward and the forward references) would need a search of $(2 \cdot sr + 1)^4$ block candidates. In the proposed combination of 2D-log and full search, the maximum number of possible candidates is reduced to $2 \cdot (2 \cdot sr + 1)^2 \cdot (21 + 2 \cdot \log_2(sr/4))$.

Table 11.1 shows a comparison of the number of block candidates needed to be searched in a full search or 2D-log search for $sr = 8$ and $sr = 16$. The combination of full search and 2D-log is a good trade-off between the complexity of full search and the low candidate search of the 2D-log estimation. For instance, using a standard search range of $sr = 16$, the motion complexity of the joint motion estimation using a combination of full and 2D-log search for $\overline{B}$ frames is increased by a factor of 25 compared to the separate motion estimation. However, if a joint motion estimation using only a full search approach is used, the factor increases to 544.

	Separate Full Search	**Joint Full Search**	**Joint Full/2D-log search**
sr = 8	578	83521	13294
sr = 16	2178	1185921	54450

Table 11.1: Number of candidate blocks comparison of interpolative motion estimation using a separated full search, combined full search and a combined full/2D-log search. The search range is fixed to $sr = 8$ and $sr = 16$ pixels.

Figure 11.4 shows an example of the proposed indexing representation enhanced coding scheme for a short transition of 4 frames. The indexing representation enhanced encoder knows (by the information in the indexing representation) that there is a transition in those specific frames and it is able to change its encoding strategy. In this case, the encoder module changes the initial GoP structure (in the example with $B2$ or two B frames between P frames), hence, it forces the following P frame to be at the end of the transition (thus, no P or I frames are allowed in the transition). Generally, from the two shots that compose a video transition, the second shot does not have any relation with the first one, therefore, the frame type is changed to a I frame instead of a P frame. After the GoP structure is changed, B frames within the transition are also changed to $\overline{B}$ frames, hence, weights extracted from the indexing representation are used to compute motion vectors using the proposed joint motion estimation.

Results such as those reported in Section 11.4 show promising gains when coding transitions of short duration using indexing representations. In the case of long gradual transition with high internal motion, the performance of this method decreases. The term internal motion refers to the motion of the shot itself, to differentiate it from the motion added by the transition when, for instance, dissolving two shots. The use of references only at the

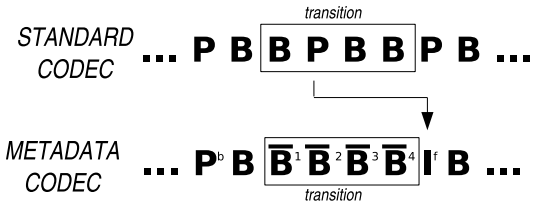


Figure 11.4: New GoP structure and interpolative mode of $\overline{B}$ frames when coding short transitions using indexing representations. The arrow represents that no P or I frame types are allowed in the transition.

beginning and the end of the transition, which may be several seconds long, is not enough to predict the frame in the middle of the transition. To overcome this problem, new references, called $\overline{B}_r$, can be introduced in the transition. With these new references, internal $\overline{B}$ frames can be predicted using references of $\overline{B}_r$ type which are closer in time and, therefore, may perform better as predictors. Basically, $\overline{B}_r$ frame types are the same as $\overline{B}$ frames with the exception that they can act as references for intermediate $\overline{B}$ frames. In that sense, they are similar to the hierarchical B frames defined by H.264 [87].

When $\overline{B}_r$ frames are used, new weights, called prediction weights, may need to be computed. Prediction weights are used in $\overline{B}$ frames in the transition and they refer, not to the start and end of the transition such as the indexing representation weights, but to the $\overline{B}_r$ frames in the transition. Consider for instance, that two references, named $\overline{\mathbf{B}}_r^b$ and $\overline{\mathbf{B}}_r^f$, are included in a dissolve. These references use the first and last frames of the transition as references ($\mathbf{P}^b$ and $\mathbf{I}^f$ respectively). Frames between these two references should use $\overline{\mathbf{B}}_r^b$ and $\overline{\mathbf{B}}_r^f$ instead of $\mathbf{P}^b$ and $\mathbf{I}^f$. The weights used by the indexing representation enhanced encoder should then be: $\overline{\mathbf{B}} = w_b \cdot \overline{\mathbf{B}}_r^b + w_f \cdot \overline{\mathbf{B}}_r^f$. On the other hand, we have:

$$\begin{aligned} \overline{\mathbf{B}} &= w_1 \cdot \mathbf{P}^b + w_2 \cdot \mathbf{I}^f \\ \overline{\mathbf{B}}_r^f &= w_3 \cdot \mathbf{P}^b + w_4 \cdot \mathbf{I}^f \\ \overline{\mathbf{B}}_r^l &= w_5 \cdot \mathbf{P}^b + w_6 \cdot \mathbf{I}^f \end{aligned} \tag{11.2}$$

where the weights $w_1, w_2, w_3, w_4, w_5, w_6$ are obtained directly from the indexing representation. Hence, the final weights w_b and w_f that refer to $\overline{\mathbf{B}}_r^b$ and $\overline{\mathbf{B}}_r^f$ can be computed as:

$$w_f = \frac{w_1 \cdot w_6 - w_2 \cdot w_5}{w_3 \cdot w_6 - w_4 \cdot w_5} \qquad w_b = \frac{w_1 \cdot w_4 - w_2 \cdot w_3}{w_4 \cdot w_5 - w_3 \cdot w_6} \tag{11.3}$$

In the case of long transition, such as the one in Figure 11.5, the modification of the GoP structure is more complex. As stated before, using only $\overline{B}$ frames in the transition is not appropriate because the start and end of the transition are too far apart in time. $\overline{B}_r$ frames are introduced in the transition, thus, $\overline{B}$ frames can use closer references in time. For instance

in Figure 11.5, the adaptation of the GoP to follow the transition is accomplished in three steps. Firstly, the first P reference in the transition is moved to a I reference at the end (frame $\mathbf{I}^f$). Secondly, $\overline{B}_r$ frames are included in the transition (in the example, $\overline{\mathbf{B}}_r^5$ and $\overline{\mathbf{B}}_r^{10}$). Finally, all the remaining frames of the transition are coded as $\overline{B}$ frames.

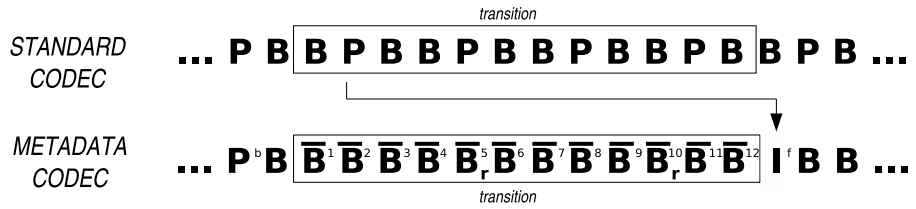


Figure 11.5: New GoP structure and interpolative mode of $\overline{B}$ and $\overline{B}_r$ frames when coding long transitions using indexing representations. The arrow represents that no P or I frame types are allowed in the transition.

Consequently, $\overline{B}_r$ frames use the closer P or I frames in time as references. In the example, video frame $\overline{\mathbf{B}}_r^5$ and $\overline{\mathbf{B}}_r^{10}$ have frames $\mathbf{P}^b$ and $\mathbf{I}^f$ as references for the interpolative motion estimation process. However, $\overline{B}$ frames use the closest $\overline{B}_r$, P or I frames in time as references. For instance, in Figure 11.5, the first group of $\overline{B}$ frames ($\overline{\mathbf{B}}^1$ to $\overline{\mathbf{B}}^4$) use references $\mathbf{P}^b$ and $\overline{\mathbf{B}}_r^5$. The second group composed of frames $\overline{\mathbf{B}}^6$ to $\overline{\mathbf{B}}^9$ use frames $\overline{\mathbf{B}}_r^5$ and $\overline{\mathbf{B}}_r^{10}$ backward and forward references. Finally, as the example transition is composed of 12 frames, frames $\overline{\mathbf{B}}^{11}$ and $\overline{\mathbf{B}}^{12}$ use references $\overline{\mathbf{B}}_r^{10}$ and $\mathbf{I}^f$.

The number of $\overline{B}$ frames between $\overline{B}_r$ frames varies for different sequences and, basically, it depends on the length of the transition and on the internal motion of the two shots composing the transition. Although this number does not greatly influence the final bitrate savings, experimental results have shown that values between 2 and 5 are a good decision for the number of $\overline{B}$ frames between $\overline{B}_r$ frames. For the experimental results, a number of 4 $\overline{B}$ frames between $\overline{B}_r$ frames has been used for all sequences. For shot transition smaller than 4 or 5 frames, $\overline{B}_r$ frames are not needed and only $\overline{B}$ frames are employed.

11.3 Selected Indexing Representation

An indexing representation that can be selected to improve the coding of transition corresponds to the MPEG-7 *Video Editing Segment Description Tools*. The MPEG-7 subclause *Video Editing Segment Description Tools* specifies tools for describing segments of visual content created by a video editing work, including attributes and structural decompositions of these segments. Among the descriptors in this subclass, the *Analytic Transition* DS describes a transition between two shots. The term analytic means that the description is made a posteriori automatically or manually based on the final video content. The *Analytic Transi-*

tion DS describes the evolution type and reliability of the analytic transition. In MPEG-7, analytic transitions are classified in three types:

- Global transitions which involve the entire video frame. The indexing representation *Global Transition* DS describes global transitions and classifies them into "fade-in", "fade-out", "dissolves", etc. For the indexing representation enhanced coding scheme proposed in this chapter, only global transitions, described by the *Global Transition* DS, are considered.

- Composition transitions which are caused by edition areas appearing on, or disappearing from the video frame. The indexing representation *Composition Transition* DS describes composition transitions in MPEG-7.

- Internal transitions have all the characteristics of a global transition but they occur only in a sub-region of the video frame. The indexing representation *Internal Transition* DS describes internal transitions.

The XML schema syntax that defines the *Analityc Transition* DS is:

```
<!-- ################################################ -->
<!--  Definition of AnalyticTransition DS             -->
<!-- ################################################ -->
<complexType name="AnalyticTransitionType" abstract="true">
  <complexContent>
    <extension base="mpeg7:AnalyticEditedVideoSegmentType">
      <sequence>
        <element name="EvolutionType" type="mpeg7:TermUseType" minOccurs="0"/>
        <element name="EditedMovingRegionRef" type="mpeg7:ReferenceType"
          minOccurs="0"/>
      </sequence>
      <attribute name="evolutionReliability" type="mpeg7:zeroToOneType"
        use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

The semantics of the *AnalitycTransitionType* are:

AnalyticTransitionType describes an analytic transition between two shots.

EvolutionType describes the type of the evolution of the analytic transition. Examples of evolution types are "cut", "analog cut" and "gradual transitions". For gradual transitions, examples of evolution types are "fade-in", "fade-out" and "dissolves", among others. (optional)

EditedMovingRegionRef describes a reference to the description of the edited moving region (edition area) that is involved, introduced or deleted in the analytic transition. (optional)

EvolutionReliability indicates the reliability of the evolution information of the analytic transition. (optional)

11.4 Experimental Results

Several experiments are performed to assess the interest of indexing representations to improve the coding of transitions. The JM-6.0a version of the H.264 video codec [42] is used for all experiments. Also, the same settings are set for the different experiments. Table 11.2 shows the conditions and settings for the H.264 video codec used in the experiments of this section.

Setting	Status
Intra period	0
Hadamard transform	on
Arithmetic coding (CABAC)	on
Error resilience	off
RD mode decision	on
MV search range	±16
Variable size motion modes	all
Motion pixel accuracy	1/8

Table 11.2: Basic H.264 settings for the experimental results of the indexing representation enhanced encoder.

Name	Transition	Frames	Resolution	fps	Motion
Linear short A	linear dissolve	4	QCIF,CIF	30	medium
Linear short B	linear dissolve	5	QCIF,CIF,TV	25	low
Linear short C	linear dissolve	5	QCIF,CIF,TV	25	high
Linear long A	linear dissolve	40	QCIF,CIF	30	medium
Linear long B	linear dissolve	30	QCIF,CIF,TV	25	low
Linear long C	linear dissolve	30	QCIF,CIF,TV	25	high
Non-linear A	non-linear dissolve	30	QCIF,CIF,TV	25	low
Non-linear B	non-linear dissolve	25	QCIF,CIF	30	high
Fade A	fade	21	QCIF,CIF	30	low
Fade B	fade	31	QCIF,CIF,TV	25	high

Table 11.3: Sequences used for the experimental tests of the indexing representation enhanced scheme for transition coding

Different transitions are evaluated, in particular, results are shown for dissolve transitions and for fade-in and fade-out transitions. In the case of dissolve transitions, two different categories are analyzed. Linear dissolves correspond to dissolve transitions where a linear temporal evolution of the weights is used, on the other hand, non-linear dissolves correspond to dissolve transitions where a non-linear temporal evolution of the weights is used. Figure 11.6

shows an example of the weight values used to create the linear and non-linear dissolves of duration 30 frames.

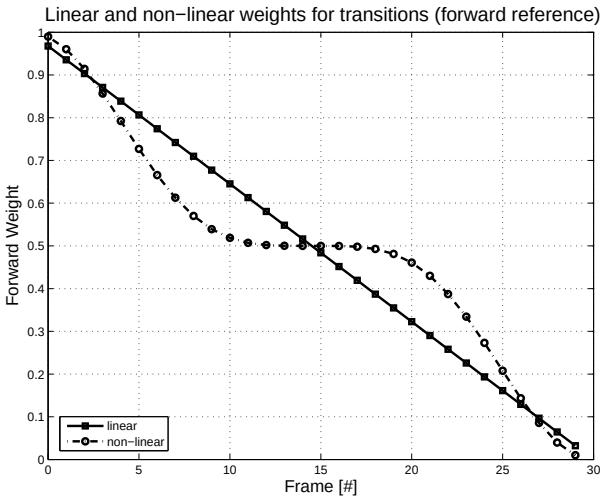


Figure 11.6: Linear and non-linear weights (for forward reference frame) used to create dissolve transitions for the test sequences. Weights for the forward reference w_f of a dissolve transition of 30 frames are represented. In the case of the backward reference, the weights are computed as $w_b = 1 - w_f$.

Table 11.3 lists all test sequences created to evaluate the performance of the transition coding. The first test sequence, *Linear short A*, involves a linear dissolve of 4 frames at QCIF and CIF resolution between sequences *Foreman* and *Akiyo*, several frames of the test sequence can be seen on Figure B.6 in the appendix B. The sequences *Linear short B* and *Linear short C* correspond to two different linear dissolve transitions of 5 frames at QCIF, CIF and TV resolutions. The test sequence *Linear short B* (key-frames presented in Figure B.7), is created from shots of sequences *Àgora* and *La nit al dia* [1] and the shots were selected so the internal motion of the test sequences was small (The original *Àgora* sequence is composed of a debate of 14450 frames while the *La nit al dia* sequence is composed of 14450 frames of a news sequence). Test sequence *Linear short C* (key-frames presented in Figure B.8), involves a dissolve transition of 5 frames between shots from sequences *Formula1* and *La nit al dia*. In this case, shots with high internal motion are selected to create the transition.

In order to test long linear transitions sequences *Linear long A*, *Linear long B* and *Linear long C* (represented in Figures B.9, B.10 and B.11) were created involving linear transitions of 40, 30 and 30 frames respectively. The shots selected to create the transitions are the same as the ones used for shorter transitions described previously. Again, shots with slow, medium and high motion are used respectively for each test. Test sequences *Non-linear A*

[1] The author would like to thank TV3 (National Television of Catalunya) for the sequences *Àgora*, *Formula1* and *La nit al dia*.

and *Non-linear B* (represented in Figures B.12 and B.13) are used to study the performance of the proposed scheme with non-linear dissolves. Test sequence *Non-linear A* involves a non-linear dissolve of 30 frames between sequences *Àgora* and *La nit al dia* at QCIF, CIF and TV resolutions and low internal motion in the shots. Test sequence *Non-linear B* involves a non-linear dissolve of 25 frames between sequences *Coastguard* and *Foreman* with high internal motion in the shots.

Finally, fade transitions are created for tests. Test sequence *Fade A* is composed of a fade-out between sequence *Akiyo* to a black sequence of 10 frames, followed by a completely black frame, and a fade-in from black to sequence *Mother and daughter* of 10 frames, resulting in 21 frames for the transition. The fade-in and fade-out are created using a linear temporal evolution of the weights. To test the behavior of the proposed scheme in fades in high motion sequences, a similar fade of 31 frames is created using two shots from sequences *Formula1* and *La nit al dia* in *Fade B*. Several key-frames from both test sequences are presented in Figures B.14 and B.15.

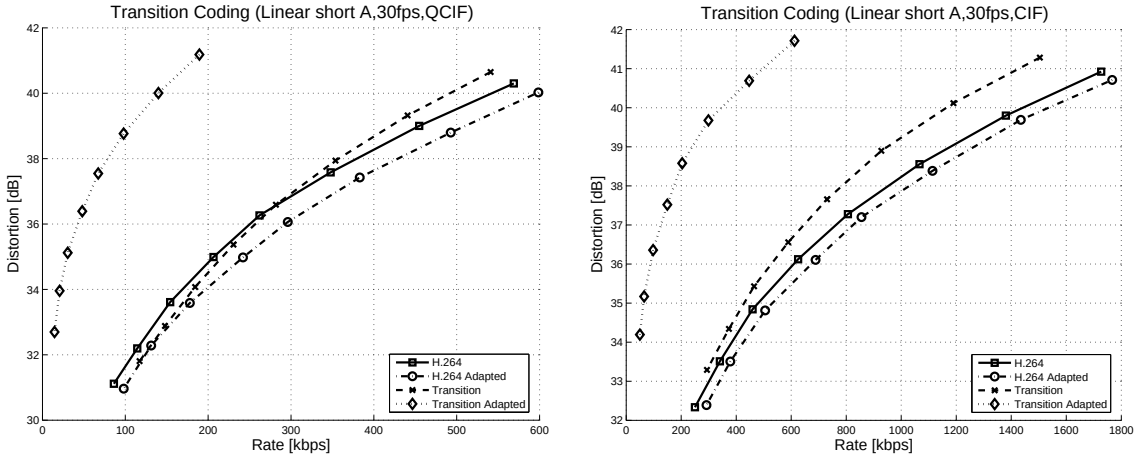
All the above described sequences are used as test sequences for the indexing representation enhanced codec. As the transitions have been manually generated, the weights used in the generation of the transitions are known and can be written with a MPEG-7 *Global Transition* DS. The indexing representation enhanced codec has access to the descriptor and has a prior knowledge of the duration of the transition, the start and end frames and the prediction weights employed in the transition to dissolve or fade the shots. Only frames that compose the transition are considered to create all rate-distortion curves and Figures in the experimental results.

The experimental results are created by comparing the standard H.264 codec with the modified indexing representation enhanced codec when coding transitions. In order to create a fair comparison, 4 different analyses are performed. The first analysis is created using the standard H.264 with a fixed GoP structure (the number of B frames between P frames of the GoP is empirically chosen as the optimum GoP structure, in terms of rate-distortion, for a particular test sequence and transition). The second analysis is performed on the same standard H.264 codec but adapting the GoP structure to the transition (forcing to use reference frames at the start and end of the transition and using only B frames within the transition). In this second analysis, no $\overline{B}$ frames are used, hence, no prediction weights are included in the codec. The third analysis uses the prediction weights in the motion estimation step but without adapting GoP structure. Finally, the fourth analysis uses a GoP structure adapted to the transition and the prediction weights computed from the indexing representation. Therefore, in this last analysis, both $\overline{B}$ and $\overline{B}_r$ frames are used inside the transition.

The first analyzed transition involves a linear dissolve of 4 frames. Table 11.4 summarizes the 4 analyses used with the test sequence *Linear short A*. In this specific sequence, the

Analysis	GoP structure	Legend
GoP not adapted to transition Prediction weights not used	$\cdots BBPB\|BPBB\|PBBP\cdots$	*H.264*
GoP adapted to transition Prediction weights not used	$\cdots BBPB\|BBBB\|IBBP\cdots$	*H.264 Adapted*
GoP not adapted to transition Prediction weights used in $\overline{B}$ frames	$\cdots BBPB\|\overline{B}P\overline{BB}\|PBBP\cdots$	*Transition*
GoP adapted to transition Prediction weights used in $\overline{B}$ frames	$\cdots BBPB\|\overline{BBBB}\|IBBP\cdots$	*Transition Adapted*

Table 11.4: GoP structure and rate-distortion curve legends for the 4 analyses of the indexing representation enhanced codec when coding short dissolves.



a) QCIF resolution b) CIF resolution

Figure 11.7: Rate-distortion curves within the transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a linear dissolve of 4 frames between sequences *Foreman* and *Akiyo* at 30 fps.

fixed GoP structure is set to $B2$ (two B frames for each P frame) and, for the indexing representation enhanced codec, the entire transition (4 frames) is encoded using $\overline{B}$ frames, thus, in this example, no $\overline{B}_r$ frames are used.

Figure 11.7 shows the rate-distortion curves when coding the linear transition for the test sequence *Linear short A* at QCIF and CIF resolution and 30 fps. As expected, the best result is achieved when adapting the GoP structure to the transition and when using prediction weights in $\overline{B}$ frames. It can be seen that only adapting the GoP structure to the transition (rate-distortion curve with legend *H.264 Adapted*) does not achieve better results than the standard H.264 codec (rate-distortion curve with legend *H.264*). Also, using only $\overline{B}$ frames without adapting the GoP (rate-distortion curve with legend *Transition*) does not improve the final results. In order to fully exploit the transition information presented in the indexing

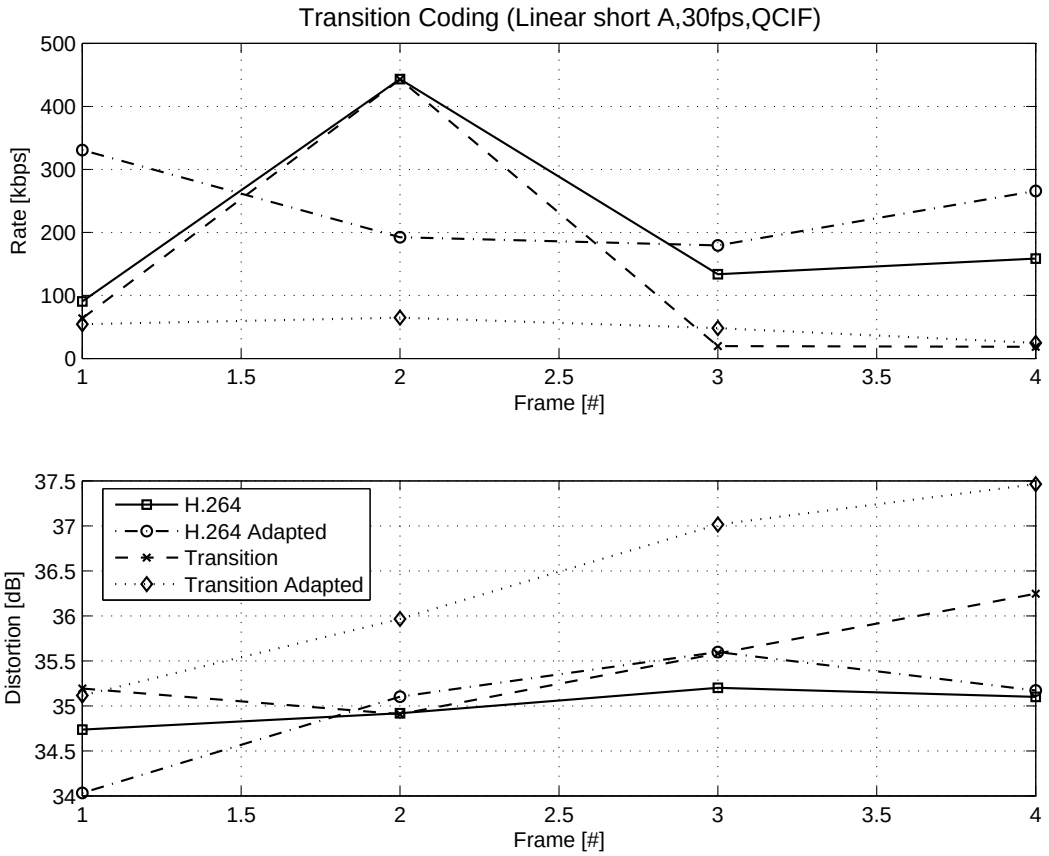
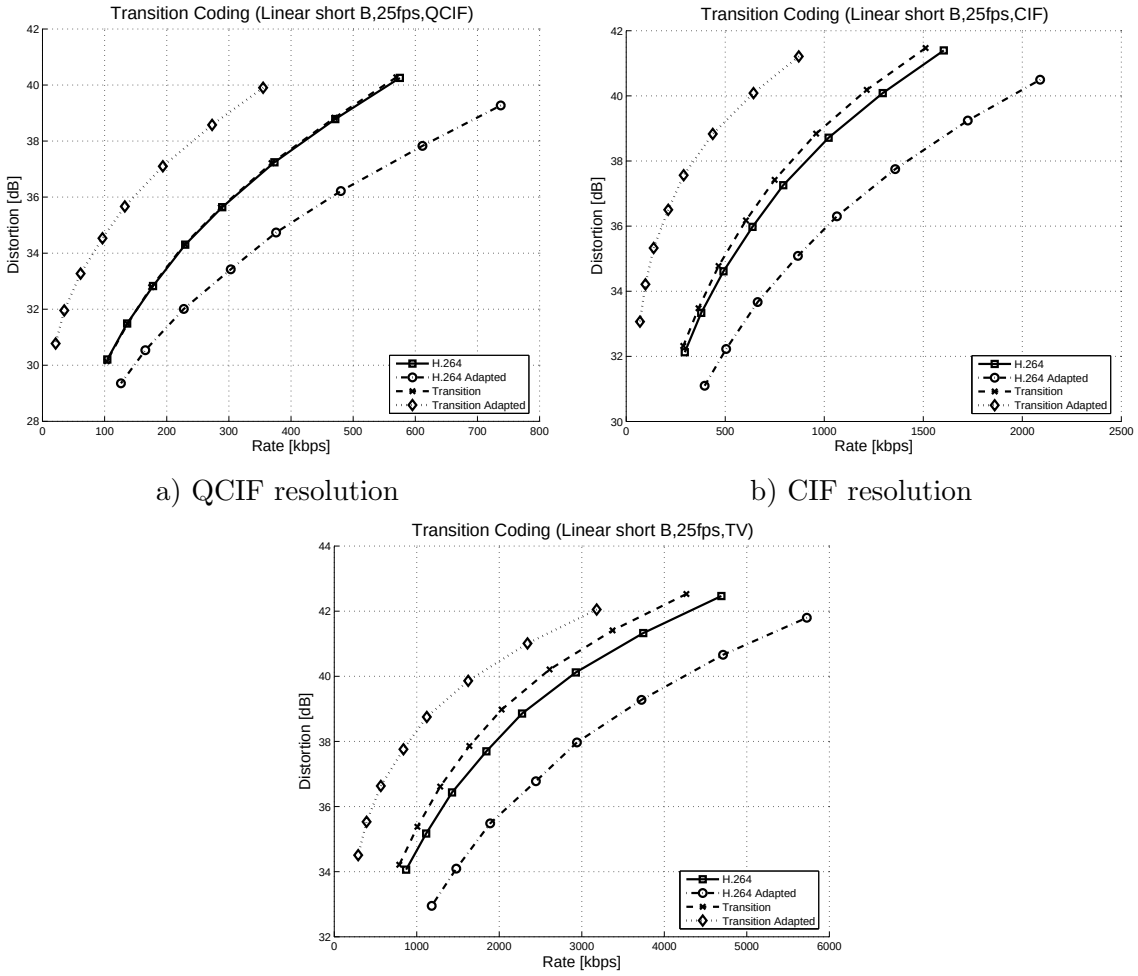


Figure 11.8: Bitrate and PSNR evolution within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec on a frame by frame basis. The transition involves a linear dissolve of 4 frames between sequences *Foreman* and *Akiyo* at 30 fps, QCIF resolution and quality factor $Q = 17$.

representation, both techniques must be introduced in the video codec.

Figure 11.8 is devoted to study, on a frame by frame basis, the encoding of the previous test sequence. A fixed quality factor of $Q = 17$ and QCIF resolution is used in the analysis. The top graph shows the bitrate in kbps needed to encode each frame of the transition. The bottom graph plots the PSNR for the individual frames of the transition. As expected, for a fixed quality factor, using $\overline{B}$ frames within the transition greatly improves the bitrate (and collaterally also improves the PSNR) needed to encode the sequence. Rate-distortion curves in Figure 11.8 show that, using prediction weights without adapting the transition (curves with legend *Transition*) improves the bitrate only on standard B frames (which are changed to $\overline{B}$ frames) as the interpolative mode is not used on P frames. Adapting only the transition without using prediction weights (curves with legend *H.264 Adapted*) improves the bitrate coding compared to previously coded standard P frames but not standard B frames.

Figure 11.9 and 11.10 show results for the test transitions *Linear short B* and *Linear short*



a) QCIF resolution

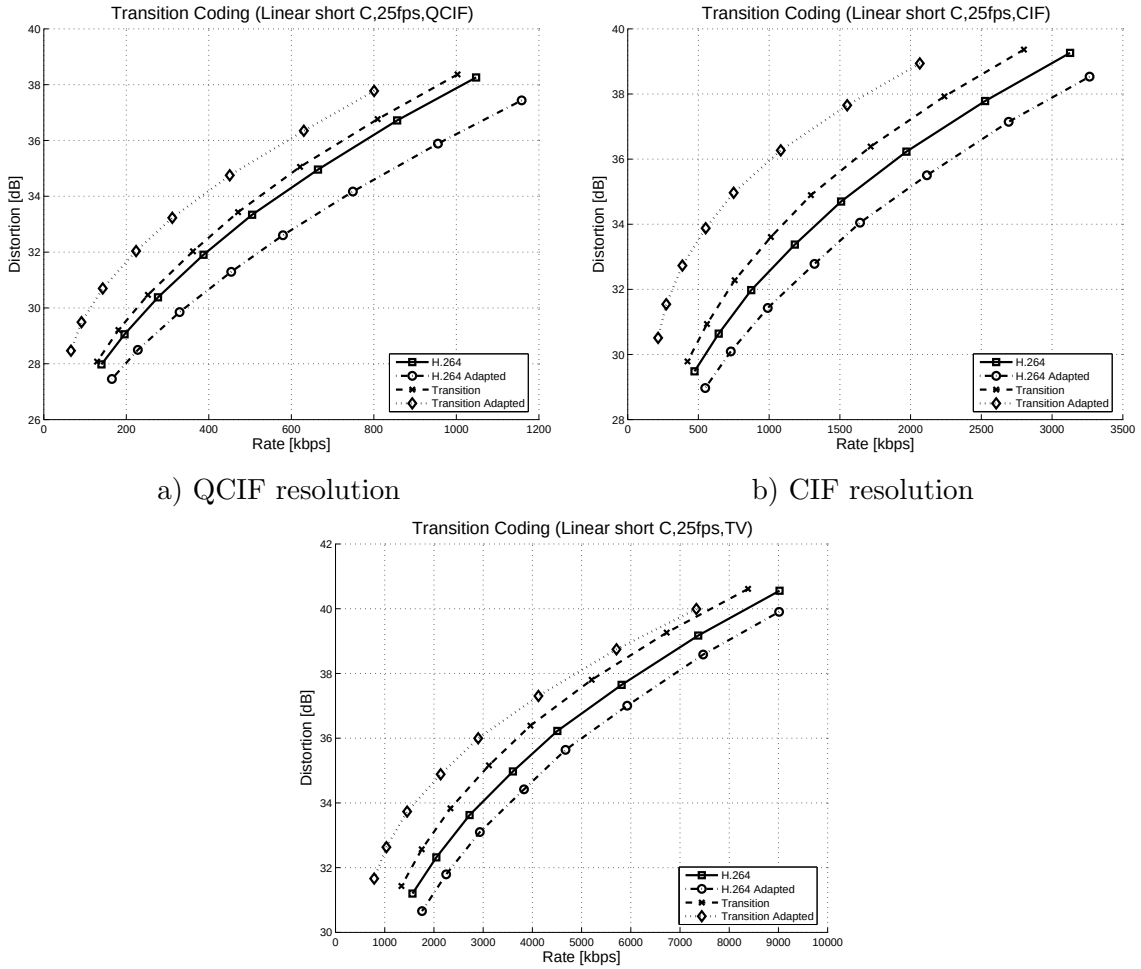
b) CIF resolution

c) TV resolution

Figure 11.9: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a linear dissolve of 5 frames between sequences *Àgora* and *La nit al dia* at 25 fps.

C at QCIF,CIF and TV resolutions. The same 4 analysis detailed in Table 11.4 are used, but this time, adapting the GoP structure inserts 5 $\overline{B}$ frames within the transition (the test transition is 5 frames long). Also, the fixed GoP structure is set to be $B1$, 1 B frames between P frames which is empirically chosen to be the optimum for both test sequences. In both cases, the best analyses results when using the indexing representation enhanced H.264 codec adapting the GoP transition and using prediction weights in $\overline{B}$ frames (rate-distortion curve with legend *Transition Adapted*). As expected, the efficiency of the indexing representation enhanced codec decreases in transitions with high internal motion (such as *Linear short C*). In any case, the use of the indexing representation enhanced codec improves the final results.

To study the effects of the proposed indexing representation enhanced codec in longer



a) QCIF resolution

b) CIF resolution

c) TV resolution

Figure 11.10: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a linear dissolve of 5 frames between sequences *Formula1* and *La nit al dia* at 25 fps.

transitions, similar tests are performed with linear dissolves of duration 30 and 40 frames. In the case of long dissolve transitions (more that 4 or 5 frames), using only $\overline{B}$ frames in the transition is not efficient as reference frames at the start and the end of the transition are too far in time to be useful. In the case of long transition, $\overline{B}_r$ frames are introduced in the transition they can act as references to $\overline{B}$ frames. Empirical results have shown that, for long transitions, the number of $\overline{B}$ frames between $\overline{B}_r$ frames should be between 2 and 5. Figure 11.11 shows the rate-distortion curves for different number (between 1 and 8) of $\overline{B}$ frames between $\overline{B}_r$ frames for the test sequence *Linear long A*. It can be seen how using values between 2 to 5 does not greatly affect the final encoding results. Different tests for other type of sequences, with high and low motion, have shown similar characteristics. Therefore, for all

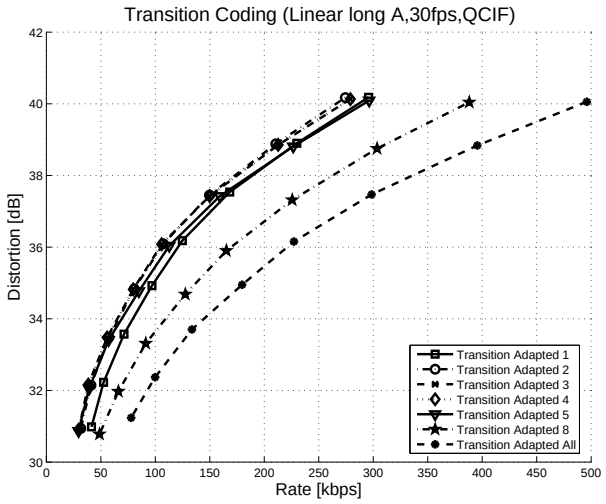


Figure 11.11: Rate-distortion curves for different number of $\overline{B}$ frames within $\overline{B}_r$ frames for test sequence *Linear long A*. The number n in the legends denotes n $\overline{B}$ frames used between $\overline{B}_r$ frames. *All* denotes no $\overline{B}_r$ frames used (all frames within transition are $\overline{B}$ frames)

.

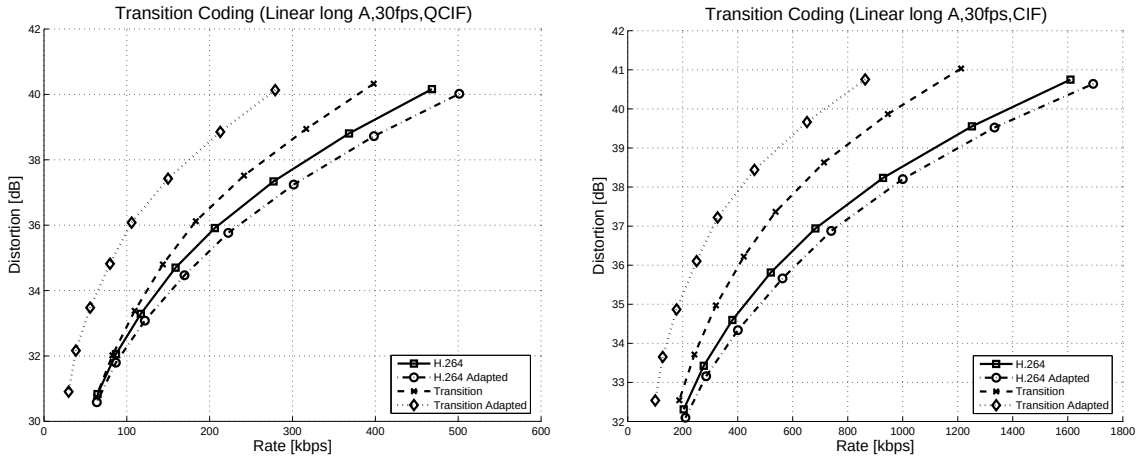
remaining analyses, a fixed number of 4 $\overline{B}$ frames between $\overline{B}_r$ frames have been selected.

Using $\overline{B}_r$ frames within the transition improves considerably the coding efficiency. However, in the case of long transitions, not having any P or I frame type in the transition itself (only at the end) may increase the memory requirements of the decoder. In cases were the memory size of the decoder is limited, P frames and not $\overline{B}_r$ frames should be used. This situation will relax the memory requirements of the decoder but, unfortunately, will not fully exploit the potential of using the indexing representation when coding transitions.

Analysis	GoP structure	Legend
GoP not adapted to transition Prediction weights not used	$\cdots PB\|BPBBPBBP\cdots PBBPBB\|PB\cdots$	*H.264*
GoP adapted to transition Prediction weights not used	$\cdots PB\|BBBB_rBBBBB_r\cdots B_rBBBBB_rBB\|IB\cdots$	*H.264* *Adapted*
GoP not adapted to transition Prediction weights used	$\cdots PB\|\overline{B}P\overline{BB}P\overline{BB}P\cdots P\overline{BB}P\overline{BB}\|PB\cdots$	*Transition*
GoP adapted to transition Prediction weights used	$\cdots PB\|\overline{BBBB}_r\overline{BBBBB}_r\cdots\overline{BB}_r\overline{BBBBB}_r\overline{BB}\|IB\cdots$	*Transition* *Adapted*

Table 11.5: GoP structure and rate-distortion curve legends for the 4 different analyses of the indexing representation enhanced codec when coding long dissolves.

Long dissolve transitions are studied in Figures 11.12, 11.13, 11.14 and 11.15. The first example (Figure 11.12) shows the rate-distortion curves for the test sequence *Linear long A*. As for short transitions, 4 different analyses are run. Table 11.5 resumes the four analyses applied



a) QCIF resolution b) CIF resolution

Figure 11.12: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a linear dissolve of 40 frames between sequences *Foreman* and *Akiyo* at 30 fps.

for this test sequence. In this case, the standard H.264 codec with fixed GoP structure is set to $B2$ (rate-distortion curve with legend *H.264*). In the indexing representation enhanced codec, the GoP is adapted to the transition and 4 $\overline{B}$ frames between $\overline{B}_r$ frames are used within the transition itself (rate-distortion curve with legend *Transition Adapted*). For completeness, two more tests without adapting the GoP but using prediction weights (rate-distortion curve with legend *Transition*) and adapting the GoP without prediction weights (rate-distortion curve with legend *H.264 Adapted*) are computed.

Two different points can be highlighted from the results on Figure 11.12. The first one is that, as expected, the total efficiency of the transition coding exploiting the indexing representation decreases in the presence of long transitions. For instance, a maximum of 3 dB in PSNR are obtained while up to 8 dB where achieved when coding short transitions. This is due to the fact that reference frames are far in time and that the internal motion of the shots prevents the motion estimation step to find good references for $\overline{B}$ or $\overline{B}_r$ frames in the transition. The second point is to notice that using prediction weights without adapting the transition (rate-distortion curve with legend *Transition*) also improves the coding efficiency. This is useful when memory size of the decoder is limited and a single I reference at the end of the long dissolve cannot be used.

Figure 11.13 shows the encoding of the *Linear long A* test sequence on a frame by frame basis for CIF resolution and a fixed quality factor of $Q = 17$ for all 40 frames of the transition. The top graph shows the bitrate in kbps needed to encode each frame of the transition. The bottom graph plots the PSNR for the same transition frames. It can be seen that the bitrate improvement when using prediction weights without adapting the GoP (curves with legend

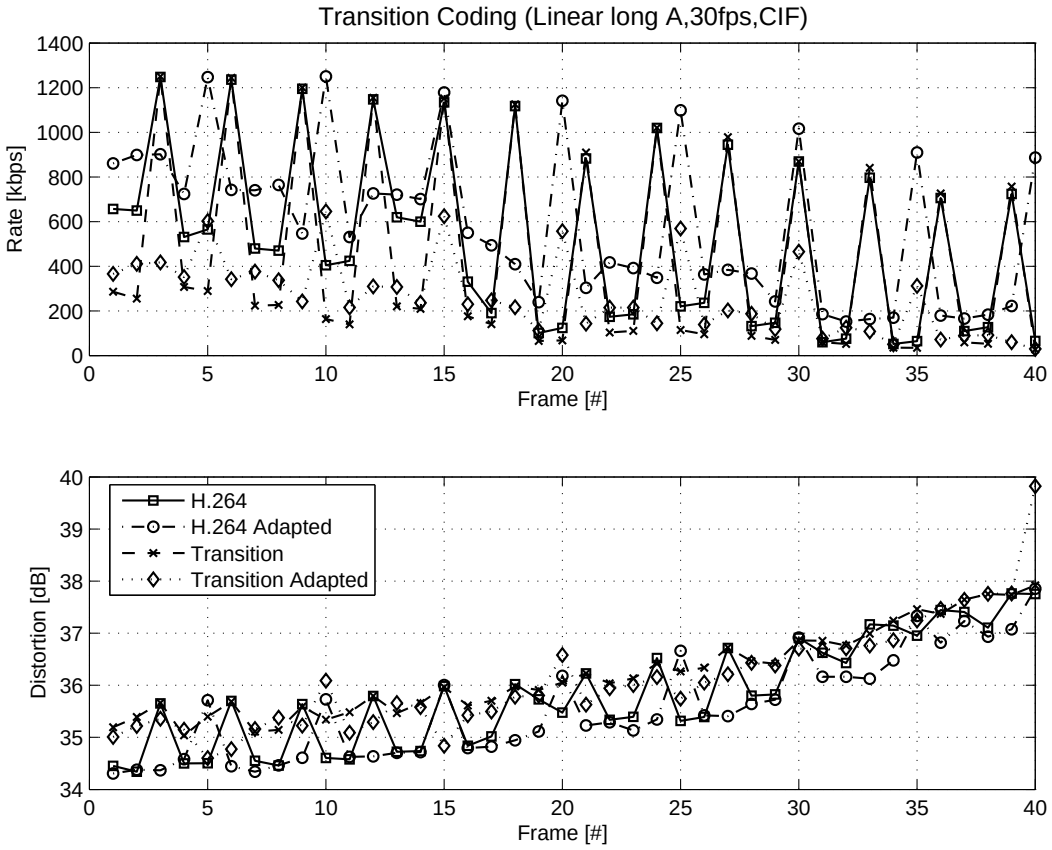
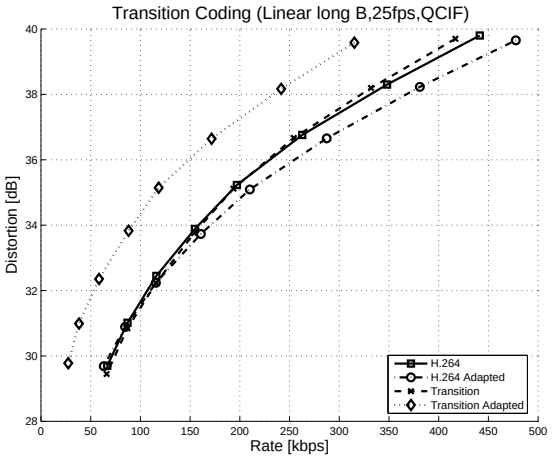


Figure 11.13: Bitrate and PSNR evolution within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec on a frame by frame basis. The transition involves a linear dissolve of 40 frames between sequences *Foreman* and *Akiyo* at 30 fps, CIF resolution and quality factor $Q = 17$.

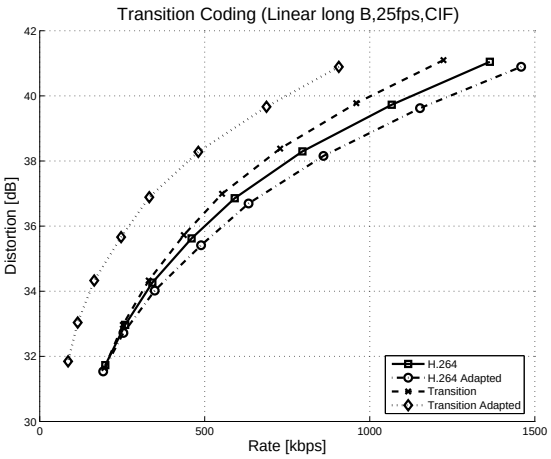
Transition) is concentrated on the B frames as there is no interpolative mode in P frames. Adapting the GoP transition without using prediction weights (curves with legend *H.264 Adapted*) does not improve the final efficiency of the encoder. Combining both adaptation of GoP and prediction weights (curves with legend *Transition Adapted*) obtains the best results in terms of bitrate. In the case of PSNR all curves present similar distortion measures.

The same 4 analyses detailed in Table 11.5 are run on sequence *Linear long B* to test the indexing representation enhanced scheme on long transitions with low internal motion. Figure 11.14 shows the rate-distortion curves at QCIF, CIF and TV resolutions. Again, the best rate-distortion curves are obtained adapting the GoP to the transition and using prediction weights. In this particular test sequence, the improvement of using an indexing representation enhanced scheme for transition coding could get up to 3 dB on PSNR improvements (rate-distortion curve with legend *Transition Adapted* in Figure 11.14.b).

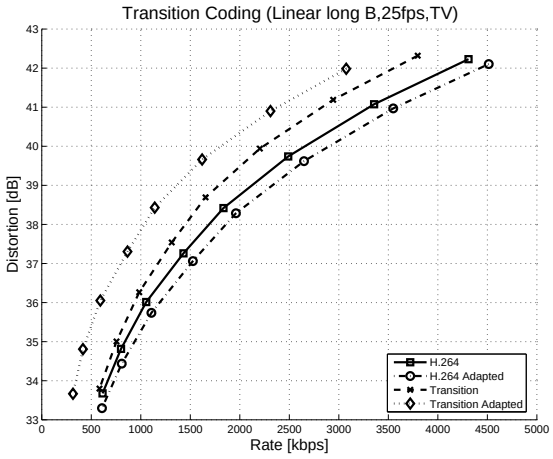
Rate-distortion curve with legend *H.264 Adapted* in Figure 11.14.b shows that only adapt-



a) QCIF resolution



b) CIF resolution

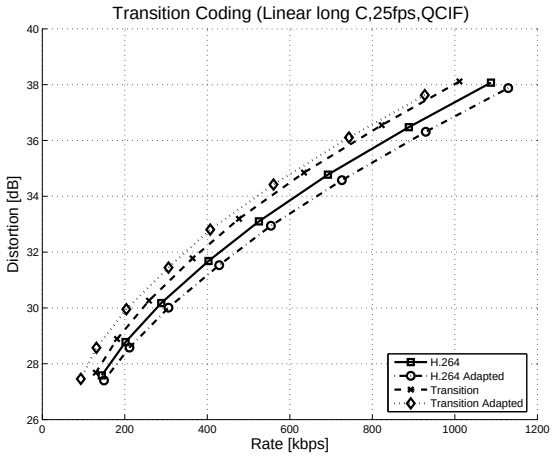


c) TV resolution

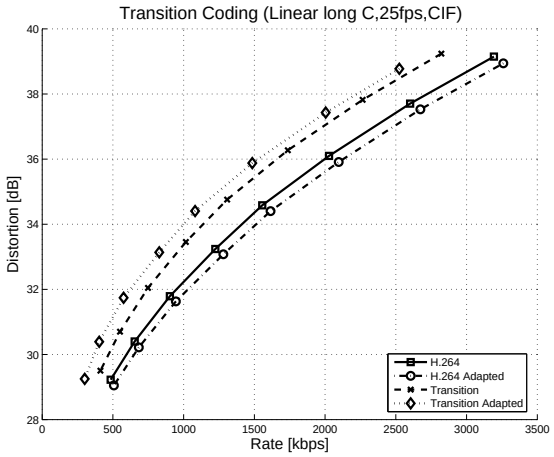
Figure 11.14: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a linear dissolve of 30 frames between sequences *Àgora* and *La nit al dia* at 25 fps.

ing the GoP structure results on a loss of performance compared to standard H.264. However, the rate-distortion curve with legend *Transition* shows how using only prediction weights improves the standard H.264 results. This improvement also increases as the test sequence resolution increases (Figure 11.14.a shows no gains at QCIF resolution while Figure 11.14.c shows significant gains when using prediction weights only). The increase is due to the fact that at higher resolutions the internal motion is less noticeable (as blocks in the image contains less details) and the dissolve transition is the dominant motion of the block. Therefore, the proposed interpolation method of $\overline{B}$ frames is better suited to predict the change in the transition.

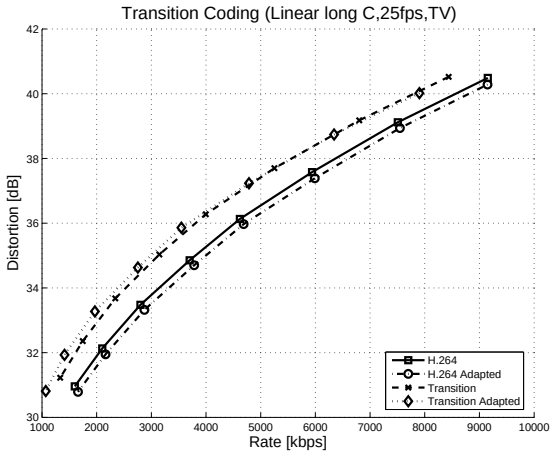
If transitions are composed of high motion shots, the final efficiency of the indexing rep-



a) QCIF resolution



b) CIF resolution



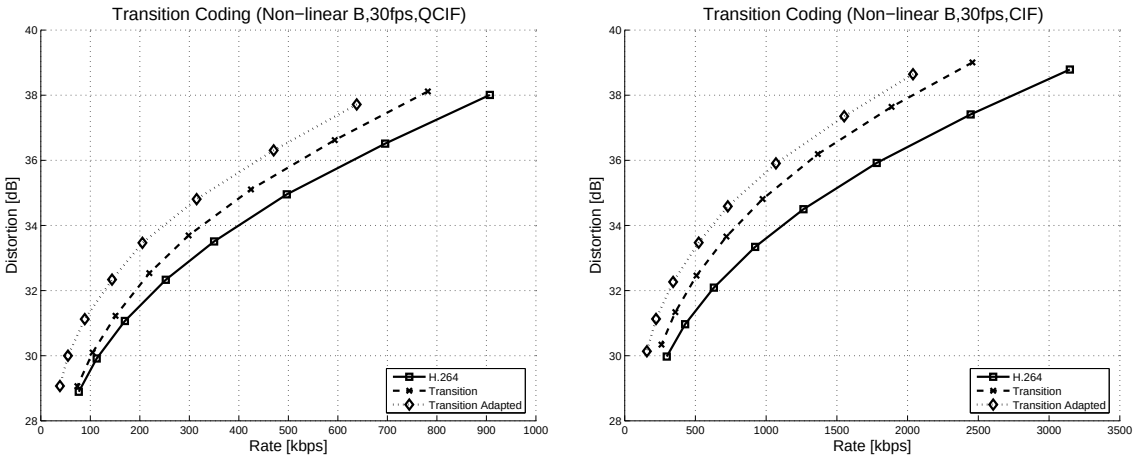
c) TV resolution

Figure 11.15: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a linear dissolve of 30 frames between sequences *Formula1* and *la nit al dia* at 25 fps.

resentation enhanced scheme decreases compared to that obtained for low motion shots. In Figure 11.15, test sequence *Linear long C* is used to study the efficiency of the proposed scheme when coding transition composed of shots with high internal motion. In these sequences, the total gains obtained by using GoP adaptation and prediction weights are usually lower than that observed with low motion sequences. For instance, with this particular test sequence, the maximum gain obtained is 1.8 dB (for CIF resolution and mid-bitrate) which is significantly less than previous test sequences with lower internal motion.

A good consequence of coding transitions composed of high motion shots is that using prediction weights without adapting the GoP structure (rate-distortion curve with legend *Transition*) is also very efficient. This effect can be easily seen in Figure 11.15.c. In this

case, high motion shots are not suited to have frame references too far and therefore, using P references within the dissolve transition helps to obtain greater improvements when using $\overline{B}$ frames instead of B frames in the transition.



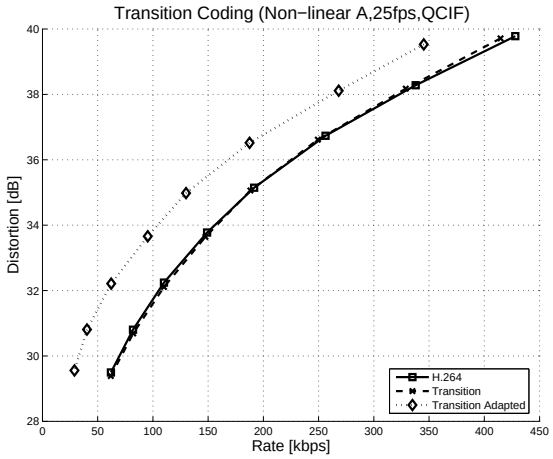
a) QCIF resolution b) CIF resolution

Figure 11.16: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a non-linear dissolve of 25 frames between sequences *Coastguard* and *Foreman* at 30 fps.

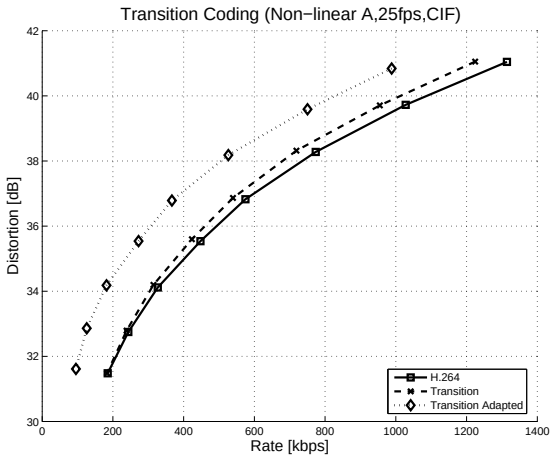
In the case of non-linear dissolves, similar results were obtained. Test sequences *Non-linear A* and *Non-linear B* are created using non-linear weight functions. In these sequences, as the weight factors are known and described in the indexing representation, the final results are similar to those of linear dissolve transitions. Figure 11.16 shows the results for the test sequence *Non-linear B* at QCIF and CIF resolutions. For clarity, the analysis with adapted GoP and no prediction weights (legend *H.264 Adapted*) has been removed from the remaining Figures since they did not improve the efficiency of the video encoder. Again, the indexing representation enhanced scheme is able to code more efficiently the non-linear dissolve transition than the standard H.264 codec. In this particular test sequence, it is able to get up to 2.5 dB for high-rates and CIF resolution when adapting the GoP structure and using prediction weights.

Figure 11.17 shows the test results for a non-linear transition composed of low motion shots. Similar results as these reported on Figure 11.14 for linear transitions are obtained. Again, for this non-linear transition, the indexing representation enhanced scheme adapting the GoP structure and using prediction weights (rate-distortion curve with legend *Transition Adapted*) increases the coding efficiency of the standard H.264 codec (rate-distortion curve with legend *H.264*). The indexing representation enhanced codec is able to obtain up to 2.3 dB gains at QCIF resolution.

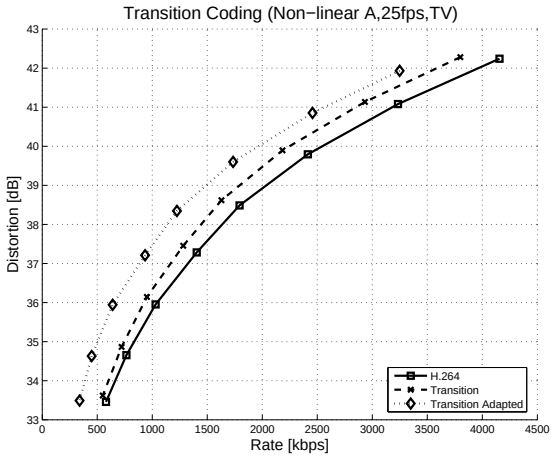
The final results of this section study transition fades. Fades can be seen as normal dis-



a) QCIF resolution



b) CIF resolution



c) TV resolution

Figure 11.17: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a non-linear dissolve of 30 frames between sequences *Àgora* and *La nit al dia* at 25 fps.

solve transitions where one of the shot is completely black. Therefore, it is expected that test results of the indexing representation enhanced scheme applied to fade-in or fade-out yield similar results. Figure 11.18 analyzes the rate-distortion curves of the indexing representation enhanced codec when coding a fade composed of low motion shots. The Figure shows significant gains at both QCIF and CIF resolutions. Gains for fades are greater than those for normal dissolve transitions. In these particular tests, up to 7 dB gains at QCIF resolution and low bitrate are achieved. This is due to the fact that one of the shots is completely black and static, therefore, the dominant motion of the transition is not the internal motion of the shots that composed the transition but the fade to black itself. Fortunately, the fading to black can be predicted very well by the joint motion estimation algorithm proposed for $\overline{B}$

and $\overline{B}_r$ frames.

Analysis	GoP structure	Legend
GoP not adapted Prediction weights not used	$\cdots PB\|BPBPBPBPBP\|P\|BPBPBPBPBP\|BP\cdots$	*H.264*
GoP not adapted Prediction weights used	$\cdots PB\|\overline{B}P\overline{B}P\overline{B}P\overline{B}P\overline{B}P\|P\|\overline{B}P\overline{B}P\overline{B}P\overline{B}P\overline{B}P\|BP\cdots$	*Transition*
GoP adapted Prediction weights used	$\cdots PB\|\overline{B}\overline{B}\overline{B}\overline{B}_r\overline{B}\overline{B}\overline{B}\overline{B}\overline{B}_r\overline{B}\|I\|\overline{B}\overline{B}\overline{B}\overline{B}\overline{B}_r\overline{B}\overline{B}\overline{B}\overline{B}\overline{B}_r\|IBP\cdots$	*Transition* *Adapted*

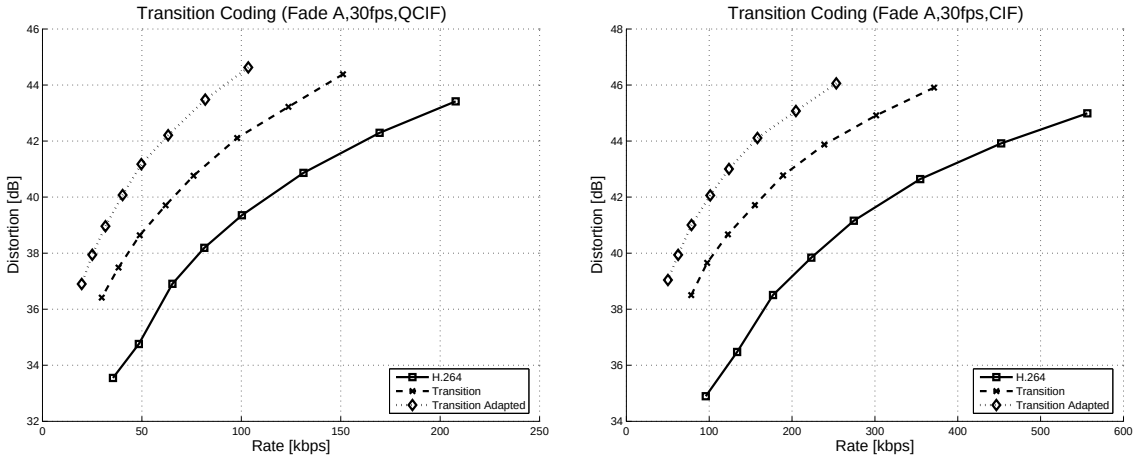
Table 11.6: GoP structure and rate-distortion curve legends for 3 different analyses of the indexing representation enhanced codec when coding fades.

Figure 11.18 shows the rate-distortion curves for the three analyses proposed. Rate-distortion curve with legend *H.264* shows the analysis when coding the test sequence *Fade A* using the standard H.264 codec with a fixed GoP structure of $B1$ (empirically found to be the optimum number of B frames for this specific sequence). The rate-distortion curve with legend *Transition* plots the results of the same analysis as before but replacing B frames for $\overline{B}$ frames so the interpolative mode using prediction weights is added (P frames are not modified in this analysis). Finally, the rate-distortion curve with legend *Transition Adapted* shows the indexing representation scheme where the GoP structure is adapted and the prediction weights are used. In this case, two fades are included in the sequence so one I reference is situated in the middle of the fades and another one at the end of the second fade. Within the fades, 4 $\overline{B}$ frames between $\overline{B}_r$ frames are used. Table 11.6 summarizes all three analysis settings. The second column lists the GoP structure of each analysis indicating the frame type that is selected for each particular frame of the transition.

To finalize this section on experimental results, Figure 11.19 shows the rate-distortion curves of sequence *Fade B* for the same three settings detailed in Table 11.6. The test sequence involves a fade-in to black and a fade-out from black between two shots with high internal motion. As expected, gains are smaller than in low motion sequences (such as the ones in Figure 11.18). However, due to the good behavior of the joint motion estimation in $\overline{B}$ and $\overline{B}_r$ frames during fades, gains are still very high for these sequences. Using the indexing representation enhanced codec when adapting the GoP structure and using prediction weights results in up to 3 dB gains for most bitrates and resolutions, and 2.5 dB if the GoP structure is not adapted, compared to the standard H.264 codec.

11.5 Indexing Representation not Available at the Decoder

This section studies the scenario where the indexing representation is not available at the decoder side. All results in Section 11.4 assume that the decoder has access to the information



a) QCIF resolution b) CIF resolution

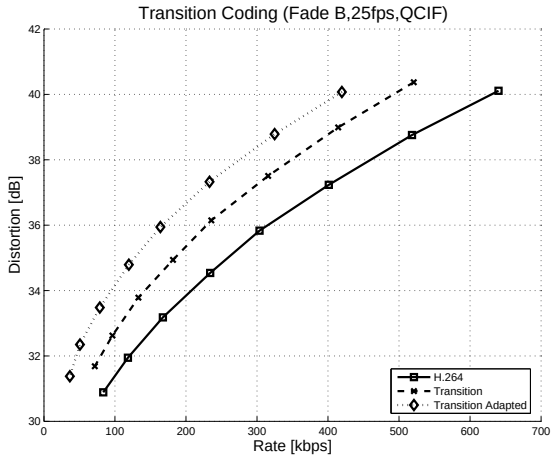
Figure 11.18: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a fade to black of 10 frames with sequence *Akiyo* followed by a fade from black of 10 frames with sequence *Mother and daughter* at 30 fps.

described by the indexing representation (in the case of transition coding, to an MPEG-7 *Global Transition* DS). If the indexing representation is not available at the decoder side, it must be streamed together with the video (as it is needed in the decoder). In such a case, only a few elements of the MPEG-7 descriptor are necessary. In particular, only the transition location in the video sequence and the temporal evolution of the weights are needed. A possible encoding of this information proposed in this thesis is shown in Table 11.7. In order to efficiently encode the weights associated with the transition, two different cases can be studied. In the case of linear transitions, the temporal evolution of weights can be determined from the transition length and no other information needs to be transmitted. For non-linear transitions, the weights need to be supplied for each frame of the transition.

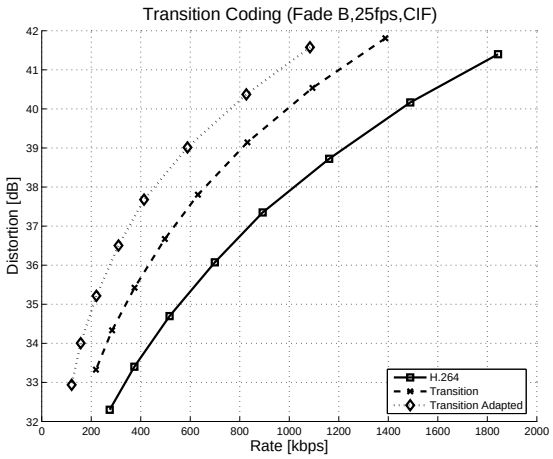
The number of bits necessary to encode the transition information is:

- For linear transitions, the transition information can be encoded using 25 bits per transition. This can be used in linear dissolve transitions where weights can be determined from the transition length. This procedure of encoding the transition information will be called implicit transition coding.

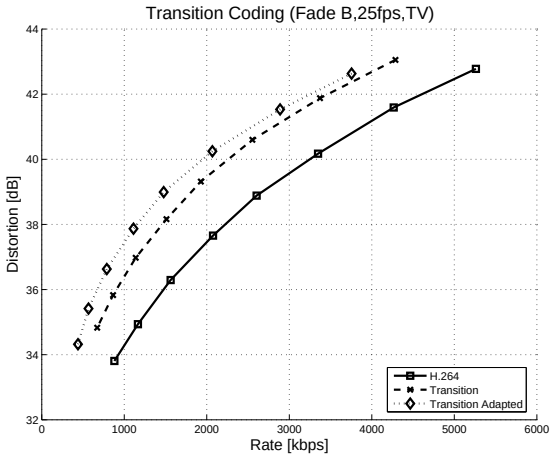
- For non-linear transitions, the transition information can be encoded in $25 + L \cdot 8$ bits/transition (L denotes the transition duration in frames). This is used in non-linear transitions where weights are supplied for each frame in a fixed point format with a precision of 1/256. This procedure of encoding the transition information will be called explicit transition coding.



a) QCIF resolution



b) CIF resolution



c) TV resolution

Figure 11.19: Rate-distortion curves within a transition using the standard H.264 codec and the indexing representation enhanced H.264 codec. The transition involves a fade to black of 15 frames with a sample shot of sequence *Formula1* followed by a fade from black of 15 frames with a sample shot of sequence *La nit al dia* at 25 fps.

In the case of indexing representations which describe a transition, the description is independent of the frame rate and size. Therefore, the amount of bits needed to encode transition information is the same whether the sequence is QCIF, TV or the bitrate used for the encoding. In order to simplify the testing scenario, only QCIF resolution is used on the experimental results of this section. In addition, the methodology of presenting the experimental results is changed in this section. No rate-distortion curves comparing the transition coding are used but, instead, a table comparing the needed bitrate to send (or not) the indexing representation is shown. This is because, as it will be seen later, the penalty of sending the indexing representation together with the content is usually very low and, both

Transition {	**bits**
InitialFrame	16
TransitionLength	8
ExplicitFlag	1
if (ExplicitFlag==1) {	
for (k=1; k < TransitionLength; k++) {	
Weight	8
}	
}	
}	

Table 11.7: Proposed binary representation syntax for the indexing representation of transitions.

rate-distortion curves (between coding the test sequence and sending or not the indexing representation) overlap each other.

Table 11.8 shows the bitrate needed to encode the test sequences listed in Table 11.3 at QCIF resolution. For each test sequence, one example at low bitrates and one at high bitrates are presented. At each bitrate, the following information is displayed in the table:

1. The first column represents the data bitrates (in kbps) when coding the transition using the standard H.264 codec at QCIF for low and high bitrates.

2. The second column details the total savings (in kbps) when using the indexing representation enhanced H.264 codec compared to the standard H.264 codec. A higher value in this column implies a more efficient indexing representation enhanced codec. The value in brackets shows the percentage with respect to the standard data bitrate of the first column. Therefore, a value of 0% implies no savings at all while a 50% value represents that the indexing representation enhanced codec is able to code the test sequence using half the bitrate of the standard codec. In this column, the indexing representation enhanced codec adapts the GoP structure and uses only $\overline{B}$ and $\overline{B}_r$ frame types in the transition (analysis with legend *Transition Adapted*).

3. The third column prints the distortion difference between the indexing representation enhanced codec and standard H.264 with fixed GoP structure. Positive values imply that the indexing representation enhanced codec improved the PSNR of the codec transition.

4. The fourth column shows the bitrate needed to send the indexing representation (in kbps). In the case of linear dissolve transitions and fades, the implicit transition coding is used so a fixed number of 25 bits per transition is needed. For non-linear transitions the explicit transition coding is used resulting on $25 + L \cdot 8$ bits per transition (with L the length of the transition in frames). In brackets, the percentage of the indexing

representation bitrate with respect to the data bitrate of standard H.264 (first column) is shown.

If the indexing representation is transmitted with the content, the bitrate increase is very low (only 0.265 kbps for the explicit transition coding) while the bitrate savings obtained by the indexing representation enhanced codec are significantly higher. For instance, for test sequence *Non-linear A*, sending the indexing representation suppose the 0.43% of the data bitrate, while savings represent the 51.3% of the total bitrate. If the indexing representation must be streamed with the content, final savings are still 49.9% compared to the standard H.264 codec.

For sequence *Linear long C*, savings are much lower, around 14.7% of the total bitrate. This test sequence is composed of high motion shots. As it has been seen on previous sections, total gains are lower for high motion sequences. However, as data bitrates are also higher for high motion sequences, the streaming of the indexing representation does not affect the results significantly. Hence, for the *Linear long C* sequence, sending the indexing representation together with the content represents less than the 0.0001% of the total bitrate.

Test	Data H.264 [kbps]	Savings [kbps]	Distortion [dB]	Indexing Representation [kbps]
Linear short A	93.42	87.00 (93.1%)	−0.05	0.188 (0.20%)
	554.46	364.98 (65.8%)	0.56	0.188 (0.03%)
Linear short B	104.40	83.16 (79.7%)	0.57	0.125 (0.12%)
	575.08	219.96 (38.2%)	−0.35	0.125 (0.02%)
Linear short C	139.88	73.88 (52.8%)	0.49	0.125 (0.09%)
	1047.80	247.04 (23.6%)	−0.48	0.125 (0.01%)
Linear long A	64.69	34.87 (53.9%)	0.07	0.019 (0.03%)
	468.28	189.21 (40.4%)	−0.03	0.019 (0.00%)
Linear long B	66.73	39.25 (58.8%)	0.08	0.021 (0.03%)
	441.39	126.19 (28.6%)	−0.22	0.021 (0.00%)
Linear long C	144.20	50.80 (35.2%)	−0.13	0.021 (0.01%)
	1087.41	160.22 (14.7%)	−0.45	0.021 (0.00%)
Non-linear A	61.99	31.80 (51.3%)	0.06	0.265 (0.43%)
	428.09	98.83 (23.1%)	−0.24	0.265 (0.06%)
Non-linear B	76.76	38.04 (49.6%)	0.18	0.225 (0.29%)
	907.05	268.96 (29.7%)	−0.29	0.225 (0.02%)
Fade A	65.38	45.61 (69.8%)	−0.01	0.075 (0.11%)
	207.80	125.94 (60.6%)	0.06	0.075 (0.04%)
Fade B	100.26	56.46 (56.3%)	0.49	0.050 (0.05%)
	768.09	264.57 (34.4%)	−0.04	0.050 (0.01%)

Table 11.8: Comparisons between the data bitrate and the bitrate needed to stream the indexing representations for transition coding.

It is important to remark that for newer version of the H.264 codec (such as JM-7.0 [43]),

prediction weights could be included in the weighted prediction scheme if needed and not be sent in the binary representation syntax proposed in Table 11.7. This would reduce the bitrate needed to stream the indexing representation. Nevertheless, as results show on Table 11.8, the needed bitrate to encode the transition information is very small compared to the achieved bitrate savings. The final bitrate savings for the proposed indexing representation enhanced codec is not influenced by having to stream the indexing representation together with the content.

In conclusion, even if the indexing representation is not available at the decoder end, the great savings obtained by this indexing representation enhanced scheme motivates the use of the MPEG-7 transition descriptor for video coding. Another advantage of the proposed scheme is that, in some cases, the bitrate needed to encode the transitions is higher than the bitrate needed to encode the shots immediately before and after the transition (if the quality is fixed). This results on peaks in the bitrate that may saturate the memory buffer of the decoder. Using the proposed scheme, these peaks can be eliminated or, at least, reduced. On the other hand, one of the main drawbacks of the scheme is that, as the gains are concentrated in the transitions, the overall gain in standard video sequences may be low if there are not many transitions in the sequence. The overall gain of this scheme in standard sequences will be extremely dependent on the number of transitions in the sequence and it is difficult to quantize them for all kind of sequences. For example, in some documentary footage or film trailers, transitions may get up to 20% or even 30% of the total time of the video scene. In those cases, the proposed scheme may be more attractive than on other sequences with practically no transitions.

Chapter 12

Long Term Reference Frame Selection

This chapter is devoted to the study of indexing representations to improve the long term selection of reference frames for standard hybrid codecs. The selection is based on a low-level descriptor which selects, among a large number of previous frames, the best possible reference frames for the one being currently encoded.

12.1 Motivation

The motivation of this approach is based on the high degree of temporal redundancy in real video sequences. Standard hybrid coders exploit this fact by using past (or future) frames as reference frames when coding the current frame. Once a good candidate has been selected as a reference frame, it is used to make a prediction of the current frame and only the difference between the prediction and the current frame is encoded and written in the bitstream. A simple block based translational motion model is also used to cope with the internal motion of the video sequence. Therefore, for each block to be coded in the current frame, the final encoder must send the motion vector information plus the difference between the current block and the predicted one.

It is natural to think that the closest frame in time will be the most similar to the frame being coded. For that reason, most hybrid codecs use the closest P or I frame in time as the reference frame when coding the current frame. For each block being coded in the current frame, a motion estimation algorithm finds the best reference block only in the closest reference frame. In this case, all blocks of the current frame being coded share the same reference frame.

Several studies have shown that using more than one reference frame for encoding can increase the coding efficiency [103, 104]. The idea that the most similar frame to the one

being coded is the closest in time is generally true for images but not for blocks in the image and the best possible reference frame may be situated several frames in the past. In that case, all encoded blocks in a single frame do not share the same reference frame. For each block, information about the frame that has been used as reference frame has to be added. This implies an increment of the bitrate. However, results show that the gain in prediction error is greater than the bitrate needed to encode the reference frame and hence, the final rate-distortion efficiency increases.

Standards such as H.263 [35] and H.264 [1] have adopted these ideas into a long term temporal prediction or multi-frame prediction scheme. In this scheme, a group of N reference frames is created for each frame to be coded and stored in a long term prediction buffer. For each block of the current frame, the motion estimation selects the best reference block among the N possible reference frames in the long term prediction buffer. In the same sense as before, the N possible reference frames are the N closest P or I frames in the video sequence. In practice, the number N of possible reference frames is limited due to two factors: Firstly, the computational complexity of performing the motion estimation and secondly, the bitrate increase needed to add the information about the reference frame.

Indexing representations can be introduced in the long term temporal prediction scheme to improve the overall coding efficiency. The indexing representations can be used to perform a pre-selection of possible N candidates reference frames to be included in the long term prediction buffer. As indexing representations have been designed for search and retrieval capabilities, the search space of possible reference frames can be increased without a severe penalty in the computational cost.

Moreover, indexing representations can further help the searching of possible reference frames. For example, shot descriptors can be used so that reference frames are only searched in shots that have similar content. The computational complexity of the motion estimation step can be reduced if a lower number of reference frames is used. As these reference frames are selected by the indexing representations, reference frames can be more appropriate in the sense that the reference frames can act as better predictors. Also, existing indexing representations about collections of audio-visual documents can point the encoder to similar content in other sequences. For example, a user may watch and store the news everyday. Indexing representations can inform the encoder that several bitstreams corresponding to past days are already stored and available to the encoder. Although not tested in the results of this thesis, it is theoretically possible that the codec makes use of those streams as possible reference for coding the current sequence.

The dark gray module in Figure 12.1 shows the coding module modified by the proposed indexing representation enhanced scheme. The only module that has to be modified is the motion compensation and prediction module. In the standard H.264 codec, the module is responsible for creating the long term prediction buffer and selects the best reference frames

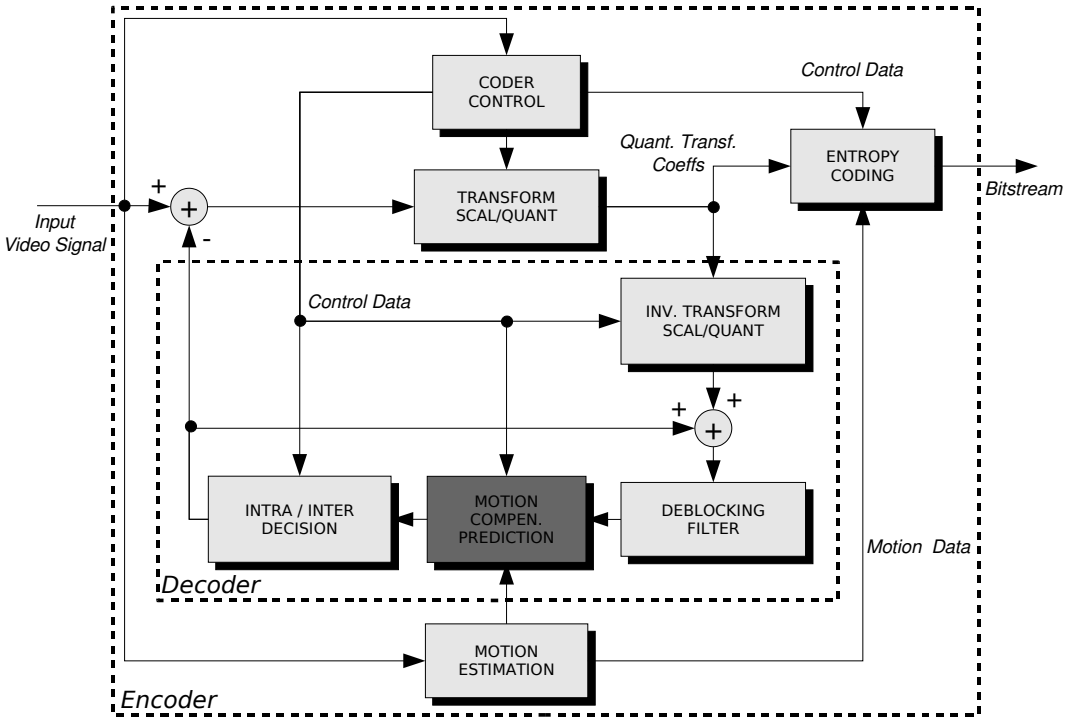


Figure 12.1: Indexing representation enhanced H.264 coding scheme. In dark gray the modules that are modified in order to exploit the indexing representation when selecting long term reference frames.

for each block being coded. Typically, the previous 5 encoded frames are used as possible reference frames in the long term prediction buffer. If the indexing representation is made available for both the encoding and decoding process then this number can be increased. In the experimental results of Section 12.4, values from 100 up to 5000 frames are used with a very small penalty on computational complexity.

Generally, increasing the number of frames to pre-select reference frames in the long term prediction buffer also increases the memory requirements of the decoder. Frames in the long term prediction buffer are needed in the decoder to compensate the motion estimation. Therefore, the decoder needs to have access to all reference frames that were used in the encoding process. If memory size in the decoder is limited, then the number of possible candidates M needs to be reduced. Another possibility for memory limited decoders is to add a mark in the bitstream. The mark can signal, for each coded frame, the reference frames that the encoder has selected using the provided indexing representation. The decoder could inspect those marks and, therefore, only the needed reference frames are stored in memory. Unfortunately, this last option would require a two pass encoding strategy and thus, only

useful for non-real time applications.

12.2 Indexing Representation Enhanced Coding Scheme

The proposed indexing representation enhanced scheme basically exploits the long term prediction buffer of current standard hybrid codecs. The basic strategy is to use an indexing representation to create a more efficient long term prediction buffer. Standard codecs, such as H.264, fill the buffer with the previous coded frames. In our scheme, reference frames in the long term prediction buffer are selected by their similarity with the frame being coded. This similarity is computed using the indexing representation and, therefore, the selection of reference frames is fast and efficient. Experimental results have shown that the search and retrieval step of similar frames using the indexing representations is less that 1% of the computational time needed for the motion estimation itself.

Having a long term prediction buffer where the contents of reference frames are similar to the frame being coded ensures a better prediction. Consider, for instance, when the start of a new shot in the sequence is about to be coded. In that case, indexing representations can indicate to the encoder to use reference frames from similar shots in the past instead of using just the previous frames from a previous shot where the image content is completely different. In a similar manner, when objects reappear in the scene, this scheme can also search for references where the same object was present before.

The proposed indexing representation scheme can be summarized as follows. First, let us assume that the long term prediction buffer is N frames long. In the scheme, instead of filling the long term prediction buffer with the N previous coded frames, the current frame to be coded is compared to M previous frames with usually $M >> N$. The comparison is made using an indexing representation for speed and reliability. The indexing representation, denoted by $\mathcal{M}_{\mathbf{I}(t)}$, and extracted from the current frame $\mathbf{I}(t)$ is compared against the indexing representations of previous frames.

Indexing representations of previous frames (at time instant t') can be extracted from the original non-coded version $\mathbf{I}(t')$ or from the coded frames $\hat{\mathbf{I}}(t')$. If the original images are used, the indexing representations $\mathcal{M}_{\mathbf{I}(t')}$ can be previously computed and there is no need to extract new descriptions. However, if the indexing representations are computed from the coded version of previous frames $\mathcal{M}_{\hat{\mathbf{I}}(t')}$, the comparison between indexing representations can be computed against frames that will actually be used in the decoder itself. This second approach has the advantage that reference frames in the long term prediction buffer are selected taking into account the coded versions. For instance, at low bitrates, where coded frames differ significantly from the original, computing the indexing representations from coded images selects better reference frames. If the chosen indexing representation is easy and fast to compute (as it will be seen on Section 12.3), it is desirable to compute it from

already coded frames.

Once the indexing representation has been computed for past coded frames, it is compared with the indexing representation $\mathcal{M}_{\mathbf{I}(t)}$ of the current frame. The distance measure between indexing representations is employed to sort all previous coded frames. Then, only the $N-1$ frames with lowest distance will be used as reference frames in the long term prediction buffer. $N-1$ frames are used because 1 frame of the long term prediction buffer is always reserved for the previous coded frame in time. This ensures that the last coded frame $\hat{\mathbf{I}}(t-1)$ is always used as a reference frame. Experimental results have shown that forcing to use $\hat{\mathbf{I}}(t-1)$ as reference frame improves the overall efficiency of the proposed scheme.

---

foreach frame $\mathbf{I}(t)$ to be encoded do:
- Obtain $\mathcal{M}_{\mathbf{I}(t)}$
- Compute and store $\mathcal{M}_{\hat{\mathbf{I}}(t-1)}$
- reset the long term prediction buffer (of N frames):
  - $\mathbf{LT}(1) = \mathbf{LT}(2) = \cdots = \mathbf{LT}(N) = 0$
- fill first reference with previous coded frame:
  - $\mathbf{LT}(1) = \hat{\mathbf{I}}(t-1)$
- for $t' = t-2$ to $t' = t-1-M$ (M previous coded frames) do:
  - Calculate distance $\mathcal{D}(t')$ between $\mathcal{M}_{\mathbf{I}(t)}$ and $\mathcal{M}_{\hat{\mathbf{I}}(t')}$
- sort all distances (from lowest, s_1, to highest, s_M)
- fill rest of $L-1$ references from the sorted distances:
  - $\mathbf{LT}(2) = \hat{\mathbf{I}}(s_1), \ldots, \mathbf{LT}(N) = \hat{\mathbf{I}}(s_{N-1})$

Figure 12.2: Pseudo-code procedure for the filling of the long term prediction buffer using indexing representations.

Figure 12.2 shows a summary of the pseudo-code algorithm used to fill the long term prediction buffer using indexing representations. In this pseudo-code procedure, $\mathbf{I}(t)$ represents frames at time instance t while $\hat{\mathbf{I}}(t)$ indicates the coded frame at the same time instant. $\mathbf{LT}(i)$ represents frame i in the long term prediction buffer. $\mathcal{D}(t')$ denotes the distance value between the indexing representation $\mathcal{M}_{\mathbf{I}(t)}$, of the frame at the current time instant and $\mathcal{M}_{\hat{\mathbf{I}}(t')}$, of the coded frame at time t'. Distances are sorted, hence, s_1 denotes the time instant of the coded frame that minimizes the distance between the corresponding indexing representations, s_2 denotes the time instant of the coded frame with the second smaller sorted distance, etc. At the end of the process, $\mathbf{LT}(i)$ is filled with a subset of M previous coded frames. This subset corresponds to the coded frames with lower distance to the current frame $\mathbf{I}(t)$. Also, the previous coded frame $\hat{\mathbf{I}}(t-1)$ is always used in the first position of the long term prediction buffer.

12.3 Selected Indexing Representation

An indexing representation useful for this proposed scheme is the MPEG-7 *Color Layout* D. This descriptor specifies a spatial distribution of colors for high-speed retrieval and browsing. The descriptor is simple, easy to compute and a distance criterion can be used to perform similarity matches. With the *Color Layout* D, frames with similar content can be recognized and included in the long term prediction buffer of reference frames.

The process to extract a *Color Layout* D is described in [46] and it can be summarized as follows. The input image is partitioned into 64 blocks. The $(i,j)^{th}$ block is a set of pixels whose size is approximately $\frac{W}{8} \times \frac{H}{8}$ (with W and H the width and height of the image). A thumbnail image of 8×8 pixels is created by extraction of a dominant color for each block. In our case, a simple mean for all pixels in the block has been used to extract the dominant color. The thumbnail image is transformed by a classical DCT. The final DCT coefficients (for the luminance and chrominance images) are quantized and stored. A simple zig-zag scan of the DCT coefficients is performed to create three vectors with luminance $\mathbf{y}$ and chrominance $\mathbf{c_b}$, $\mathbf{c_r}$ quantized coefficients. If needed, the quantized luminance and chrominance coefficients of high frequency can be truncated and discarded. The final coefficients for the luminance and chrominance images are stored as the indexing representation, in this case a *Color Layout*, of the entire image.

The XML schema syntax that defines the *ColorLayout* D is:

```
<!-- ################################################ -->
<!--  Definition of ColorLayout D                     -->
<!-- ################################################ -->
<complexType name="ColorLayoutType" final="#all">
  <complexContent>
    <extension base="mpeg7:VisualDType">
      <sequence>
        <element name="YDCCoeff" type="mpeg7:unsigned6"/>
        <element name="CbDCCoeff" type="mpeg7:unsigned6"/>
        <element name="CrDCCoeff" type="mpeg7:unsigned6"/>
        <choice>
          <element name="YACCoeff2">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned5"/>
                </simpleType>
                <length value="2"/>
              </restriction>
            </simpleType>
          </element>
          <element name="YACCoeff5">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned5"/>
                </simpleType>
                <length value="5"/>
              </restriction>
            </simpleType>
          </element>
          <element name="YACCoeff9">
            <simpleType>
```

```
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="9"/>
        </restriction>
      </simpleType>
    </element>
    <element name="YACCoeff14">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="14"/>
        </restriction>
      </simpleType>
    </element>
    <element name="YACCoeff20">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="20"/>
        </restriction>
      </simpleType>
    </element>
    <element name="YACCoeff27">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="27"/>
        </restriction>
      </simpleType>
    </element>
    <element name="YACCoeff63">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned5"/>
          </simpleType>
          <length value="63"/>
        </restriction>
      </simpleType>
    </element>
  </choice>
  <choice>
    <sequence>
      <element name="CbACCoeff2">
        <simpleType>
          <restriction>
            <simpleType>
              <list itemType="mpeg7:unsigned5"/>
            </simpleType>
            <length value="2"/>
          </restriction>
        </simpleType>
      </element>
      <element name="CrACCoeff2">
        <simpleType>
          <restriction>
            <simpleType>
              <list itemType="mpeg7:unsigned5"/>
            </simpleType>
            <length value="2"/>
          </restriction>
```

```
    </simpleType>
  </element>
</sequence>
<sequence>
  <element name="CbACCoeff5">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="5"/>
      </restriction>
    </simpleType>
  </element>
  <element name="CrACCoeff5">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="5"/>
      </restriction>
    </simpleType>
  </element>
</sequence>
<sequence>
  <element name="CbACCoeff9">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="9"/>
      </restriction>
    </simpleType>
  </element>
  <element name="CrACCoeff9">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="9"/>
      </restriction>
    </simpleType>
  </element>
</sequence>
<sequence>
  <element name="CbACCoeff14">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="14"/>
      </restriction>
    </simpleType>
  </element>
  <element name="CrACCoeff14">
    <simpleType>
      <restriction>
        <simpleType>
          <list itemType="mpeg7:unsigned5"/>
        </simpleType>
        <length value="14"/>
      </restriction>
    </simpleType>
  </element>
</sequence>
```

```
      <sequence>
        <element name="CbACCoeff20">
          <simpleType>
            <restriction>
              <simpleType>
                <list itemType="mpeg7:unsigned5"/>
              </simpleType>
              <length value="20"/>
            </restriction>
          </simpleType>
        </element>
        <element name="CrACCoeff20">
          <simpleType>
            <restriction>
              <simpleType>
                <list itemType="mpeg7:unsigned5"/>
              </simpleType>
              <length value="20"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
      <sequence>
        <element name="CbACCoeff27">
          <simpleType>
            <restriction>
              <simpleType>
                <list itemType="mpeg7:unsigned5"/>
              </simpleType>
              <length value="27"/>
            </restriction>
          </simpleType>
        </element>
        <element name="CrACCoeff27">
          <simpleType>
            <restriction>
              <simpleType>
                <list itemType="mpeg7:unsigned5"/>
              </simpleType>
              <length value="27"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
      <sequence>
        <element name="CbACCoeff63">
          <simpleType>
            <restriction>
              <simpleType>
                <list itemType="mpeg7:unsigned5"/>
              </simpleType>
              <length value="63"/>
            </restriction>
          </simpleType>
        </element>
        <element name="CrACCoeff63">
          <simpleType>
            <restriction>
              <simpleType>
                <list itemType="mpeg7:unsigned5"/>
              </simpleType>
              <length value="63"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
    </choice>
  </sequence>
</extension>
```

```
  </complexContent>
</complexType>
```

The semantics of the *ColorLayoutType* are:

YDCCoeff The first quantized DCT coefficient of the Y component.

CbDCCoeff The first quantized DCT coefficient of the Cb component.

CrDCCoeff The first quantized DCT coefficient of the Cr component.

YACCoeff The second and successive quantized DCT coefficients of the Y component. Coefficients are zig-zag scanned.

CbACCoeff The second and successive quantized DCT coefficients of the Cb component. Coefficients are zig-zag scanned.

CrACCoeff The second and successive quantized DCT coefficients of the Cr component. Coefficients are zig-zag scanned.

The MPEG-7 standard recommends a distance measure so similarity between *Color Layout* Ds, and therefore between images represented by the descriptor, can be computed. The distance is computed by measuring the weighted Euclidean distance between DCT coefficients. Let $\mathbf{y}$, $\mathbf{c_b}$, $\mathbf{c_r}$ and $\mathbf{y}'$, $\mathbf{c'_b}$, $\mathbf{c'_r}$ be the DCT coefficients of the first and second descriptors respectively and N_y, N_c be the number of luminance and chrominance coefficients for the two descriptors. The distance between *Color Layout* Ds can be computed as:

$$\mathcal{D} = \sqrt{\sum_{i=0}^{N_y-1} \lambda_{\mathbf{Y}}(i) \left(\mathbf{y}(i) - \mathbf{y}'(i)\right)^2} + \sqrt{\sum_{i=0}^{N_c-1} \lambda_{\mathbf{Cb}}(i) \left(\mathbf{c_b}(i) - \mathbf{c'_b}(i)\right)^2} + \\ + \sqrt{\sum_{i=0}^{N_c-1} \lambda_{\mathbf{Cr}}(i) \left(\mathbf{c_r}(i) - \mathbf{c'_r}(i)\right)^2} \tag{12.1}$$

where λ values denote weights for each *Color Layout* coefficients. They should decrease according to the zig-zag scan line order. Therefore, larger weights are given to low frequency coefficients to increase the importance of low frequencies of the image in the distance measure. In the experiments, the following λ weights have been used:

$$\begin{aligned} \lambda_{\mathbf{Y}} &= (2, 2, 2, 1, 1, 1, 1, 1, \ldots, 1, 1) \\ \lambda_{\mathbf{Cb}} &= (2, 1, 1, 1, 1, 1, 1, 1, \ldots, 1, 1) \\ \lambda_{\mathbf{Cr}} &= (4, 2, 2, 1, 1, 1, 1, 1, \ldots, 1, 1) \end{aligned} \tag{12.2}$$

Even though the indexing representation allows storing the 64 coefficients for each Y, Cb and Cr components, the quantized coefficients are truncated and only the 6 corresponding to the lowest frequencies are stored in the representation, $N_y = N_c = 6$. This ensures that the *Color Layout* D is kept simple (only 18 coefficients in total are stored per image) and that the distance criterion can be computed very fast without any significant penalty on similarity accuracy. Also, as it will be seen on Section 12.5, this results on a smaller, in terms of size,

indexing representation which is very useful in scenarios where the indexing representation is not available at the decoder side and needs to be streamed with the content.

The resulting distance, $\mathcal{D}$, measures the similarity between two *Color Layout* Ds. For instance, if $\mathcal{D} = 0$ then both *Color Layout* Ds are identical and the frames represented by these indexing representations are expected to be very similar. However, greater values of $\mathcal{D}$ represent very different indexing representations and, therefore, images with no similar content between them.

12.4 Experimental Results

Experimental tests are performed to compare the use of the *Color Layout* D to improve the coding of video sequences. As the indexing representation enhanced scheme exploits the temporal redundancy between frames of the video, the proposed scheme in this section should perform better on sequences with repeating shots or sequences where the same objects disappear and reappear in the scene. Table 12.1 lists the sequences used for the experimental tests on this section.

Name	Frames	Resolution	fps	Motion	Type of content
Geri	364	176×92	6,30	high	video clip
Jornal da noite	3000	QCIF	5,25	low,medium	news
Telediario	3000	QCIF	5,25	low,high	news
Formula1	2500	QCIF,CIF	5,25	high	sports
Fr3	5000	QCIF,CIF	5,25	medium	documentary
Àgora	14000	QCIF,CIF	5,25	low	interview

Table 12.1: Sequences used for the experimental tests of the indexing representation enhanced scheme for the long term reference frame selection.

The test base is composed of 6 sequences at QCIF and CIF resolutions. The sequence *Geri* is composed of 12 seconds of RAW uncompressed frames at a resolution of 176×92 pixels. The *Geri* sequence (several key-frames of the sequence can be seen in Figure B.16) corresponds to a video clip with high internal motion composed of short and repeating shots. Sequences *Jornal da noite* and *Telediario* correspond to two different news programs from the MPEG-7 test set. The MPEG-7 test set is originally encoded in MPEG at CIF resolution, thus, both sequences are extracted from the MPEG bitstream and low-pass filtered and decimated to QCIF resolution in order to create test sequences with limited coding artifacts. By using the QCIF versions, coding artifacts are reduced and, as it will be seen later on the results, the final improvements obtained by the indexing representation enhanced codec are similar to other sequences. The *Jornal da noite* test sequence contains a mixture of low, medium and

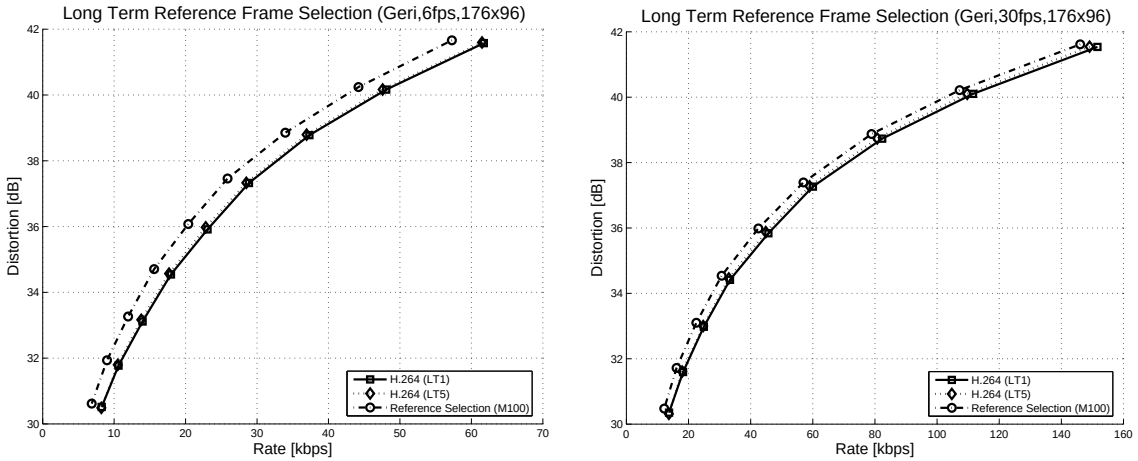
high motion shots and it has a duration of 3000 frames (Several key-frames of the sequence are shown in Figure B.18). The *Telediario* test sequence also contains a mixture of shots with different internal motion and it has a duration of 3000 frames (several key-frames of the sequence are shown in Figure B.17).

Sequence *Fr3* consists of 3 minutes and 20 seconds of a documentary program. The original sequence is RAW uncompressed video at CIF resolution and it is down-sampled to also create a QCIF version (key-frames are shown in Figure B.19). Sequences *Formula1*, and *Àgora* correspond to original RAW uncompressed sequences at CIF, also down-sampled to create QCIF versions. Sequence *Formula1* consists of 1 minute and 40 seconds of a sport sequence with high motion and short video shots (key-frames in Figure B.20). Sequence *Àgora* is composed of 9 minutes and 20 seconds of a debate with very low internal motion and long but repeated video shots (key-frames in Figure B.21).

All sequences are encoded with a standard H.264 codec and an indexing representation enhanced H.264 codec. The conditions and settings for the H.264 video codec are the same as in Section 11.4 and listed in Table 11.2. In addition, for all experiments, only P frame types are used. The indexing representations are created off-line and, in this section, they are supposed to be accessible for both encoder and decoder. The number of coefficients in the *Color Layout* are $N_Y = 6$ and $N_C = 6$. So a total of 18 DCT coefficients per frame are used for the distance measures. In the experiments, the size of the long term prediction buffer **LT** has been fixed to $L = 5$ in the H.264 encoder. Therefore, the indexing representation enhanced scheme pre-selects, among a variable number of previous frames M, the 4 frames with lowest distance to the frame being coded.

Figure 12.3 shows the comparison between the H.264 codec and the indexing representation enhanced H.264 codec for the sequence *Geri*. Two experiments are run for the standard H.264 codec. The first one uses only one reference frame in the long term prediction buffer (rate-distortion curve with legend *H.264 (LT1)*). The second one encodes the same sequence but using the previous 5 frames as reference frames in the long term prediction buffer (rate-distortion curve with legend *H.264 (LT5)*). The third experiment corresponds to the indexing representation enhanced H.264 codec (rate-distortion curve with legend *Reference Selection (M100)*). In this case, 4 frames of the long term prediction buffer are pre-selected, on a frame by frame basis, among the last 100 coded frames.

At 6 fps (Figure 12.3.a), the indexing representation enhanced codec is able to select better references, showing gains up to $1dB$ for low bitrates. At higher bitrates, the efficiency decreases to 0.5 dB compared to standard H.264 codec. At 30 fps, shown in Figure 12.3.b, the efficiency of the indexing representation enhanced codec decreases compared to the *Geri* sequence at 6 fps. This is expected as original frames are more similar between them at higher frames rates. However, using the indexing representation in the encoder still provides a gain. For this test sequence, the proposed scheme obtains gains in PSNR of, approximately, 0.7



a) 6 fps b) 30 fps

Figure 12.3: Rate-distortion curves using the standard H.264 codec and the indexing representation enhanced H.264 codec for the long term reference frame selection of test sequence *Geri*.

dB at low bitrates and 0.3 dB at higher bitrates. These gains are basically concentrated at the beginning of shots of the video sequence. In these parts of the sequence, the previous frames are not related to the current shots and cannot be used as proper reference frames. However, the indexing representation enhanced encoder is able to search further back in time and locates other reference frames in previous shots which may be similar to current frames.

Figure 12.4 shows the bitrate gains on a frame by frame basis for a segment of the sequence *Geri* at 6 fps. The solid black line at the bottom of both graphs shows the different shots that compose the video segment. As it can be seen, the efficiency of the standard H.264 codec at the start of shots decreases due to the fact that there are no previous reference frames which may be useful for encoding. However, the indexing representation enhanced codec (rate-distortion curve with legend *Reference Selection (M100)*) is able to locate proper reference frames. An example is, for instance, frame number 38, where the indexing representation enhanced codec can greatly reduce the bitrate associated with the frame.

Results on Figures 12.5 show the rate-distortion curves for the sequence *Jornal da noite* from the MPEG-7 test set. For the indexing representation enhanced codec, the *Color Layout* Ds are used to pre-select 4 possible reference frames against the last 100 frames in the case of 5 fps and 500 at 25 fps. Experimental results show gains in bitrate and PSNR quality for all bitrates. At 5 fps, savings up to 0.5 dB in PSNR are achieved against the standard H.264 codec with 5 reference frames in the long term prediction buffer and 1 dB against standard H.264 with 1 reference frame (Figure 12.5.a). In the case of 25 fps, the maximum gains are obtained at low bitrates with up to 0.4 dB in PSNR quality compared to H.264 with 5 reference frames (Figure 12.5.b).

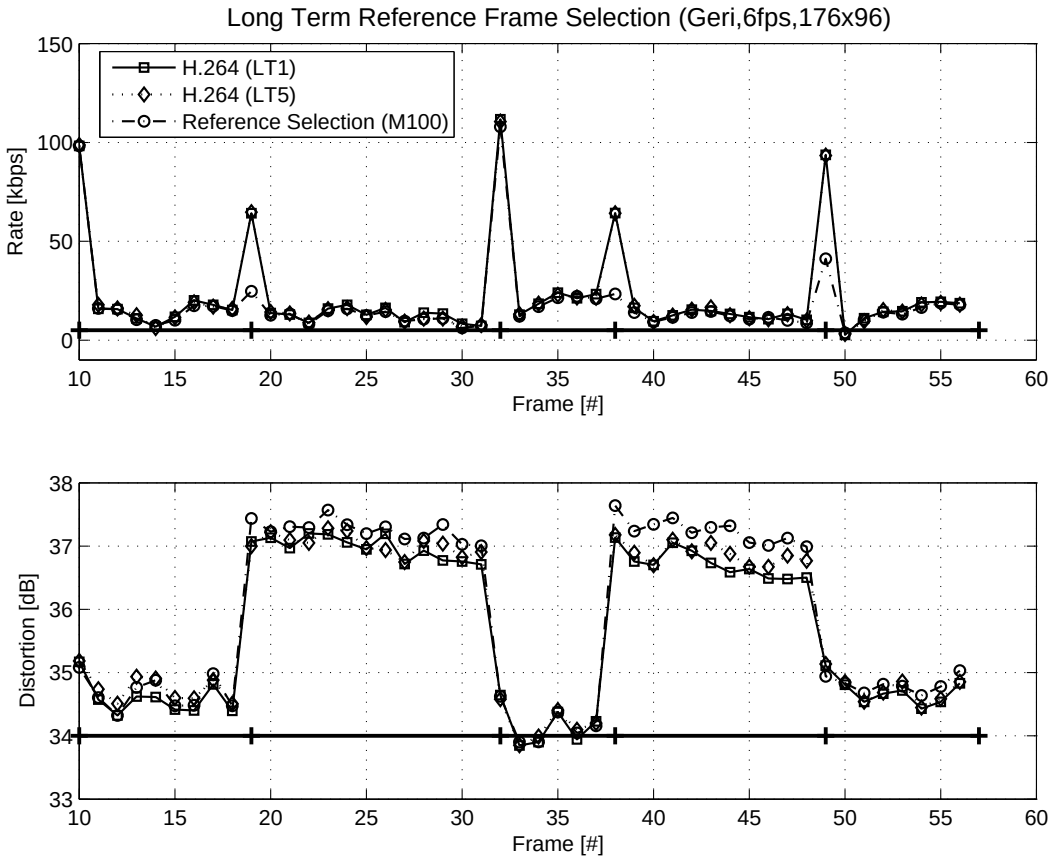
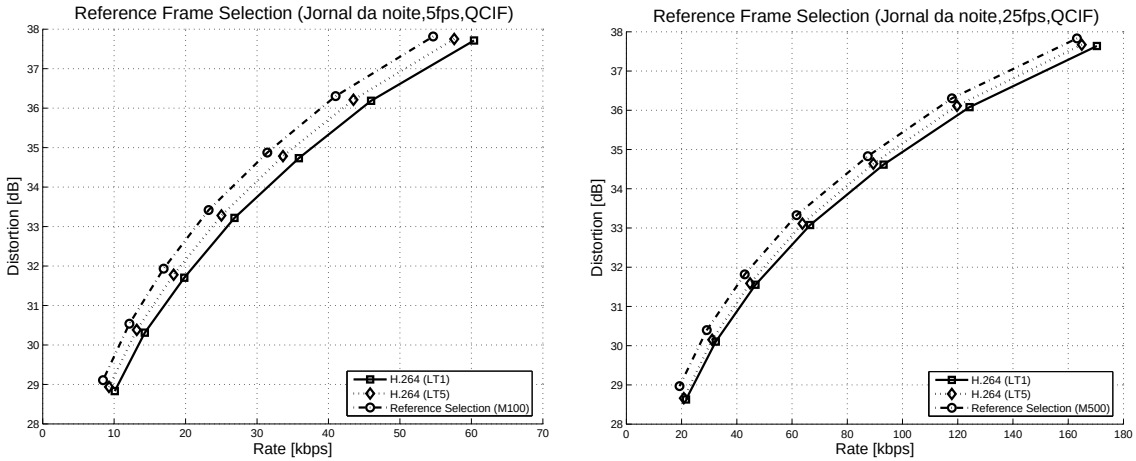


Figure 12.4: Bitrate and PSNR evolution using the standard H.264 codec and the indexing representation enhanced H.264 codec on a frame by frame basis for sequence *Geri*.

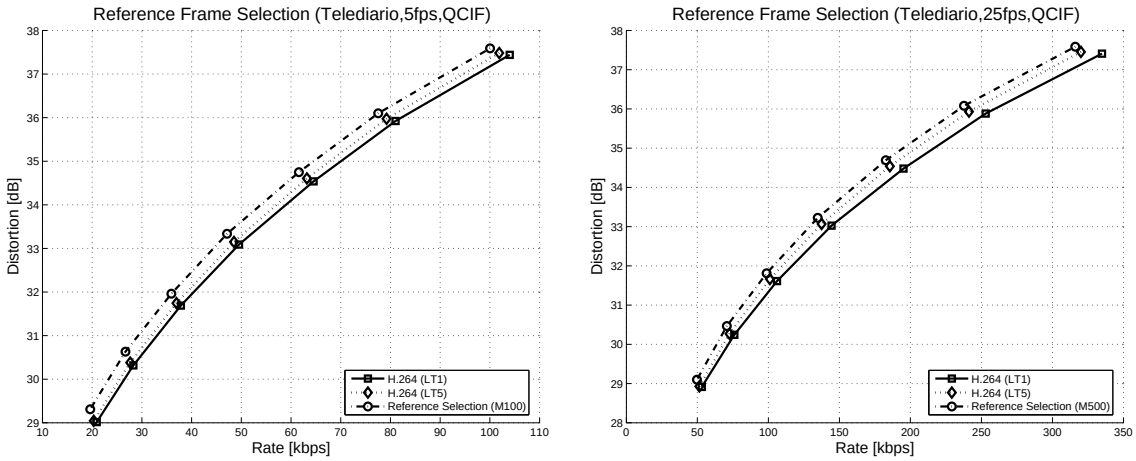
Figures 12.6.a and 12.6.b show the same experiment for 3000 frames of the *Telediario* test sequence. Rate-distortion curves show gains up to 0.4 dB in PSNR quality at the same bitrate for the indexing representation enhanced codec compared to the standard H.264 with 5 reference frames in the long term prediction buffer. The *Telediario* sequence is composed of long shots with low motion and almost no repetition. Therefore, the efficiency of the indexing representation enhanced codec decreases a little compared to previous test sequences.

Rate-distortion curves in Figure 12.7 show the experimental results for the sequence *Formula1* at QCIF and CIF resolutions at 5 and 25 fps. To simplify the Figures, only the experiments with the H.264 codec using 5 reference frames in the long term prediction buffer is compared to the indexing representation enhanced codec at CIF resolution. The *Formula1* sequence is a sport sequence with high internal motion and short shots. Experimental results show gains up to 0.6 dB at 5 fps for QCIF (Figure 12.7.a) and around 0.5 dB in the case of the CIF version (Figure 12.7.c). If the sequence is encoded at 25 fps, experimental results show similar gains than results at 5 fps. As the internal motion of the *Formula1* sequence is



a) 5 fps b) 25 fps

Figure 12.5: Rate-distortion curves using the standard H.264 codec and the indexing representation enhanced H.264 codec for the long term reference frame selection of test sequence *Jornal da noite*.

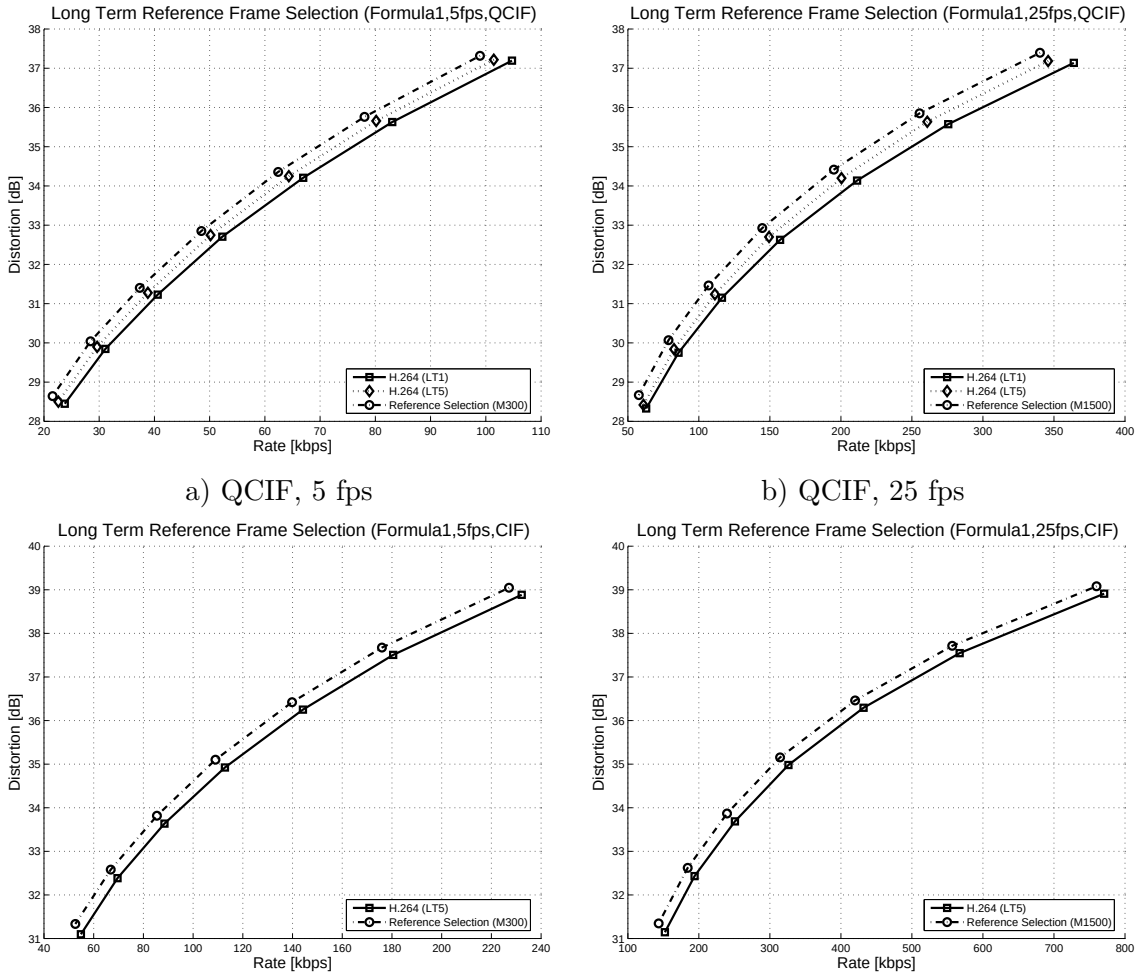


a) 5 fps b) 25 fps

Figure 12.6: Rate-distortion curves using the standard H.264 codec and the indexing representation enhanced H.264 codec for the long term reference frame selection of test sequence *Telediario*.

already very high, both versions, at 5 and 25 fps, still preserve a high internal motion and the indexing representation enhanced codec is able to select better reference frames than the standard codec.

Experimental results of the long term reference frame selection using indexing representations for the sequence *Fr3* do not show any coding gains. Rate-distortion curves in Figure 12.8



a) QCIF, 5 fps

b) QCIF, 25 fps

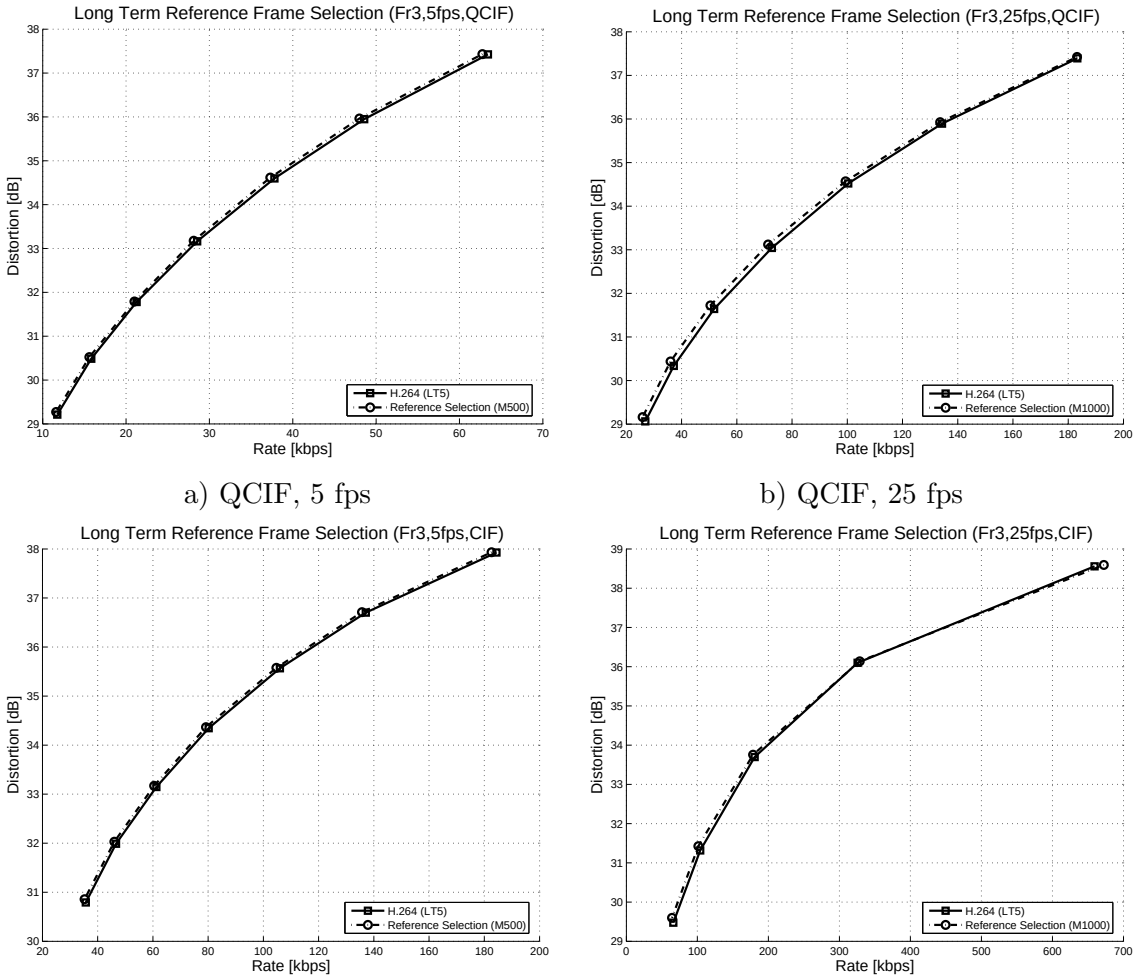
c) CIF, 5 fps

d) CIF, 25 fps

Figure 12.7: Rate-distortion curves using the standard H.264 codec and the indexing representation enhanced H.264 codec for the long term reference frame selection of test sequence *Formula1*.

show similar efficiency for both the standard H.264 codec and the indexing representation enhanced codec. This result is due to the fact that the *Fr3* sequence is composed of long and non-repeated shots and, therefore, the reference frames extracted by the indexing representations are not better than the 5 most recent encoded frames.

On the other hand, results for test sequence *Àgora* show remarkable gains with respect to the standard H.264 codec. The *Àgora* sequence corresponds to a low internal motion interview sequence. As shots are similar and repeated through the entire sequence, the selected indexing representation is able to locate good reference frames. Figure 12.9 show the rate-distortion curves for the H.264 codec using 5 references frames and the indexing representation enhanced codec using 1000 (at 5 fps) and 5000 (at 25 fps) previous frames to pre-select 4 reference



a) QCIF, 5 fps

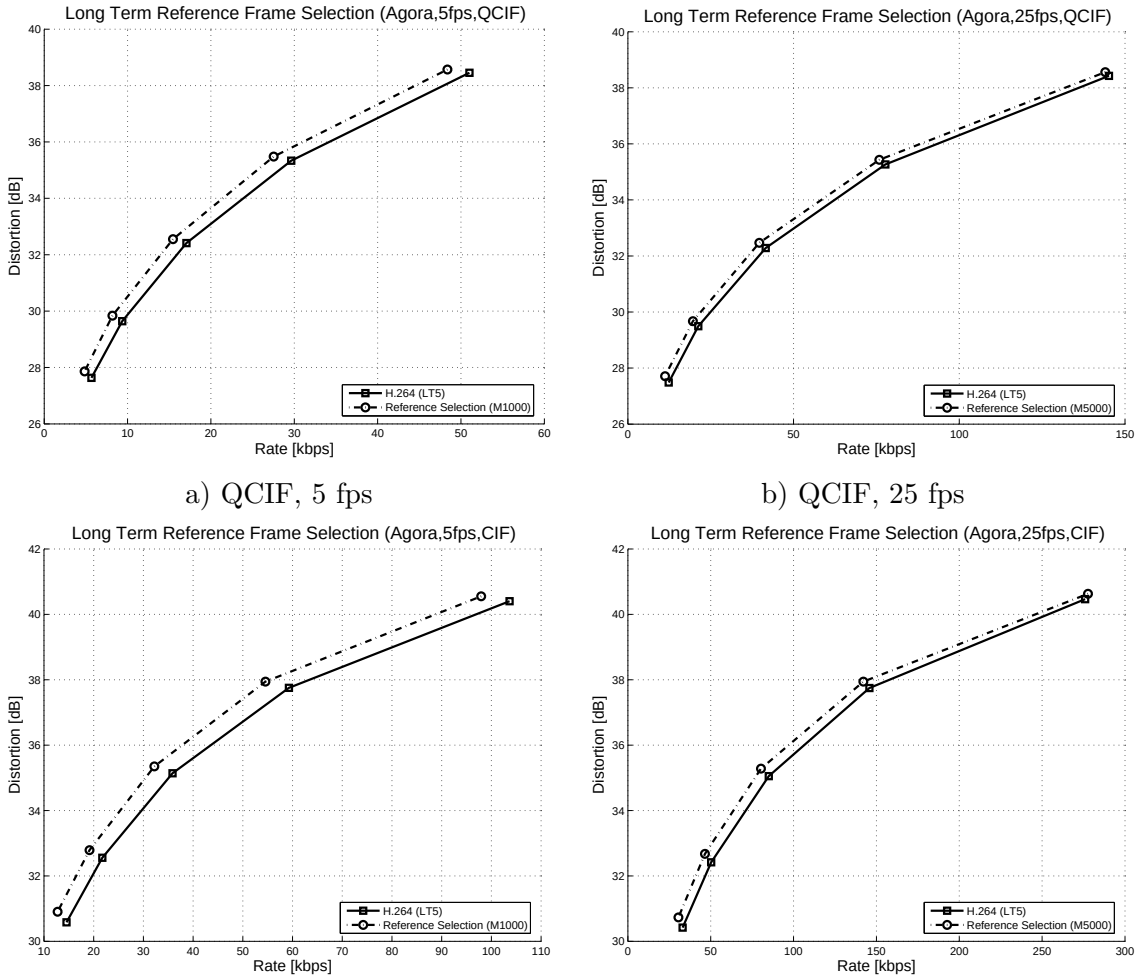
b) QCIF, 25 fps

c) CIF, 5 fps

d) CIF, 25 fps

Figure 12.8: Rate-distortion curves using the standard H.264 codec and the indexing representation enhanced H.264 codec for the long term reference frame selection of test sequence *Fr3*.

frames. In this sequence, gains are similar between QCIF and CIF resolutions but they decrease when going to 5 to 25 frames per second. When coding at 5 fps, greater gains are achieved for the *Àgora* sequence (up to 1 dB). At 5 fps, the internal motion of the sequence is greater and therefore the standard H.264 with the 5 previous references decreases in efficiency. The indexing representation enhanced codec, on the other hand, is still able to select good reference frames by inspecting several coded frames in the past.



a) QCIF, 5 fps

b) QCIF, 25 fps

c) CIF, 5 fps

d) CIF, 25 fps

Figure 12.9: Rate-distortion curves using the standard H.264 codec and the indexing representation enhanced H.264 codec for the long term reference frame selection of test sequence *Àgora*.

12.5 Indexing Representation not Available at the Decoder

The long term reference frame selection scheme proposed in this section needs to access the indexing representations at both the encoder and decoder sides. Even if the indexing representation is computed from the coded sequence (as seen in Section 12.2), the indexing representation of the current frame $\mathcal{M}_{\mathbf{I}(t)}$ is still needed in order to compute the distances with the previous indexing representations. In the scenario where the indexing representations are not already available at the decoder side, they must be streamed together with the content. The encoding of the *Color Layout* D is performed using the MPEG-7 binary coding. Table 12.2 represents the binary encoding proposed by the MPEG-7 standard.

ColorLayout {	**bits**
CoeffPattern	1-2
if (CoeffPattern==11) {	
NumOfYCoeffIndex	3
NumOfCCoeffIndex	3
}	
YDCCoeff	6
CbDCCoeff	6
CrDCCoeff	6
for (k=1; k < NumOfYCoeff; k++) {	
YACCoeff	5
}	
for (k=1; k < NumOfCCoeff; k++) {	
CbACCoeff	5
}	
for (k=1; k < NumOfCCoeff; k++) {	
CrACCoeff	5
}	
}	

Table 12.2: MPEG-7 Binary representation syntax for the *Color Layout* D.

For the experimental results, one *Color Layout* D is extracted for each frame using $N_y = 6$ coefficients for the Y component and $N_c = 6$ coefficients for the Cb and Cr components. The *Color Layout* D is independent of the frame size of the input image, consequently, the resulting bitrate needed to send the indexing representation is 95 bits per frame for all kind of sequences and resolutions. As *Color Layout* D is very similar across frames in a shot, it is compressed using a standard Lempel-Ziv algorithm [112] and sent to the decoder before the transmission of the content. After compression, the indexing representation bitrates result on approximately 15 bits per frame.

Tables 12.3 and 12.4 show the comparison of data and indexing representation bitrates. Both tables represent the same information showed in Table 11.8 but, this time, for the long term reference frame selection scheme. To simplify the results, only the QCIF resolution is analyzed. As the *Color Layout* D is independent of frame resolution, the penalty (in bitrate) of having to send the indexing representation together with the content is greater for low resolutions where data bitrate is lower.

Table 12.3 lists bitrates for all previous test sequences (detailed in Table 12.1) at 5 fps (except *Geri* at 6 fps). For each test sequence, results are shown for QCIF resolution at both low and high bitrates. Table 12.4 lists the same bitrate comparisons for the 25 fps version of the test sequences (sequence *Geri* at 30 fps). As detailed in Table 11.8, the first column

lists the data bitrate for the standard H.264 codec using 5 reference frames. The second column details the total savings when using indexing representations to pre-select reference frames (in brackets the percentage with respect to the first column). The third column prints the distortion difference between the indexing representation enhanced codec and the H.264 codec. Finally, the last column shows the bitrate needed to stream the *Color Layout* D (in brackets the percentage with respect to data rate in first column).

Test	**Data H.264 [kbps]**	**Savings [kbps]**	**Distortion [dB]**	**Indexing Representation [kbps]**
Geri	8.20	1.34 (16.3%)	0.13	0.096 (1.17%)
	36.94	2.98 (8.1%)	0.06	0.096 (0.26%)
Jornal da noite	9.29	0.84 (9.0%)	0.17	0.070 (0.75%)
	57.60	2.96 (5.1%)	0.06	0.070 (0.12%)
Telediario	20.32	0.70 (3.5%)	0.27	0.070 (0.34%)
	101.90	1.86 (1.8%)	0.11	0.070 (0.07%)
Formula1	22.61	1.04 (4.6%)	0.14	0.085 (0.38%)
	101.45	2.50 (2.5%)	0.10	0.085 (0.08%)
Fr3	11.75	0.14 (1.2%)	0.06	0.070 (0.60%)
	63.41	0.66 (1.0%)	0.01	0.070 (0.11%)
Àgora	7.85	1.02 (13.0%)	0.20	0.065 (0.83%)
	44.52	2.56 (5.8%)	0.12	0.065 (0.15%)

Table 12.3: Comparisons between the data bitrate and the bitrate needed to stream the indexing representations for the long term reference frame selection. All test sequences are analyzed at 5 fps except *Geri* which is analyzed at 6 fps.

Test	**Data H.264 [kbps]**	**Savings [kbps]**	**Distortion [dB]**	**Indexing Representation [kbps]**
Geri	13.68	1.49 (10.9%)	0.16	0.480 (3.51%)
	80.76	1.86 (2.3%)	0.13	0.480 (0.59%)
Jornal da noite	20.82	1.55 (7.4%)	0.31	0.350 (1.68%)
	164.90	1.74 (1.1%)	0.16	0.350 (0.21%)
Telediario	51.43	1.70 (3.3%)	0.17	0.350 (0.68%)
	320.06	3.99 (1.2%)	0.13	0.350 (0.11%)
Formula1	61.03	3.30 (5.4%)	0.25	0.425 (0.70%)
	345.93	5.80 (1.7%)	0.21	0.425 (0.12%)
Fr3	26.86	0.96 (3.6%)	0.10	0.350 (1.30%)
	183.30	0.12 (0.1%)	0.03	0.350 (0.19%)
Àgora	21.24	1.63 (7.7%)	0.18	0.325 (1.53%)
	145.18	1.26 (0.9%)	0.13	0.325 (0.22%)

Table 12.4: Comparisons between the data rate and the rate needed to stream the indexing representations for the long term reference frame selection. All test sequences are analyzed at 25 fps except *Geri* which is analyzed at 30 fps.

For the indexing representation enhanced codec, it would be desirable that, if the indexing representation needs to be streamed, the total bitrate needed to send the representation would be lower than the gains obtained by using it. Table 12.3 and 12.4 show that this is true for all sequences in the experimental tests except for sequence *Fr3*. For instance, sequence *Geri* at 5 fps presents bitrate savings of 16.3% (comparing the standard H.264 codec with the indexing representation enhanced codec at the same quality factor) while sending the indexing representation suppose only the 1.17% of the total rate. In this case, if the indexing representation is attached to the data, final bitrate savings are still 15.13% compared to standard H.264. In the case of sequences at 25 fps, the indexing representation bitrates are higher, as the indexing representation is extracted for each frame of the sequence, but still lower than the savings obtained by using the representation by the encoder. As the *Color Layout* D is independent of frame size, the penalty for sending the indexing representation together with the content on higher resolution sequences, such as CIF or TV, is even lower.

As conclusions, the experimental results have shown that the use of indexing representations can improve the efficiency of the long term temporal prediction of standard codecs and, specially, the H.264 codec. Also, even if the indexing representation has to be streamed with the content, it is still useful to improve the long term selection of reference frames based on the provided indexing representations. The key idea of the proposed scheme is to formulate the motion estimation problem of current standard codecs into a search and retrieval problem. In this thesis, the search and retrieval scheme has been used to select good reference frames in the long term prediction buffer. However, this scheme could be extrapolated to the entire motion estimation step and replace it by a pure search and retrieval step. For instance, a possible futuristic application would be the encoding of sequences with a high degree of repetition, such as mobile phone video conferences, where the encoder has to deal basically with similar sequences. In these sequences, the same person appears on the scene, sometimes wearing glasses or with the same clothing. For this application, the encoder may be able to use a learning based coding scenario, where a database is built with representative parts of the sequence that can be searched and retrieved later if similar content needs to be encoded.

Chapter 13

Video Segment Shuffling

This chapter is devoted to present the use of an indexing representation, that is extracted to provide browsing and summarization functionalities, to improve the coding of video sequences. The selected indexing representation describes the entire video sequence as a unit and it is used to organize, or shuffle, the order of the video frames of the sequence in order to help standard hybrid codecs to better exploit its temporal redundancy.

The video segment shuffling scheme, together with the long term reference frame selection scheme proposed in the previous chapter, exploit the same basic concept. Both schemes use indexing representations in order to group similar frames together to exploit the temporal redundancy of video sequences. However, the two schemes focus on the problem from different starting points. The long term reference frame selection scheme uses a low level descriptor in order to select, in the compression stage, the best possible reference frames for the frame currently being encoded in a frame by frame basis. The video segment shuffling scheme, however, uses a high level descriptor to create a new coding order for the entire video sequence. In this case, the temporal redundancy of the sequence is exploited globally and before any compression takes place.

13.1 Motivation

The motivation of this scheme relies on exploiting the temporal redundancy of video sequences in a global manner. Standard hybrid codecs exploit the temporal redundancy by performing a motion estimation between the current image being coded and a previous reference frame (or various previous reference frames such as H.263 or H.264 codecs). With the use of a B frame type, these reference frames can also be situated in the future. B frames have proved to generally increase the coding efficiency, either because of the linear combination of two motion compensated reference frames or because of the exploitation of future reference frames where regions may not be occluded [72].

The use of a B frame type with previous and past reference frames implies that the coding order, the order in which frames are encoded, and the display order, the order in which frames are originally set in the sequence, need to be different to effectively increase the coding efficiency. Generally, the coding order is changed only by a small amount of frames (the distance between P frames in hybrid codecs). This distance is usually small for two reasons. First, as reference frames are further apart in time, the internal motion of the sequence makes them worse predictors for the current frame being coded. Second, for real time applications, the delay introduced in the system cannot be very high.

If non-real time applications are considered, and the entire video is available before the encoding, a global shuffling of the video sequence could be performed. If no other information is used, searching for the optimum coding order (in terms of coding efficiency) could lead to very difficult problems as, for instance, the computational complexity would be too high due to the enormous number of possible combinations. However, indexing representations can be exploited to create a coding order that reduces the amount of combinations while effectively increasing the final coding efficiency. This coding order will be able to exploit the entire temporal redundancy of the video instead of only the local temporal redundancy. If the indexing representations have been created to perform browsing functionalities, they are able to summarize the entire video sequence by grouping video segments, group of frames of the sequence with variable lengths, with similar colors, content or motion information. The relationships between video segments defined by the indexing representation can be used by the proposed scheme to create the new coding order.

One of the advantages of this scheme is that no coding module of standard hybrid codecs must be modified. The shuffling is performed before the actual encoding takes place. Therefore, both encoder and decoder modules remain unmodified and the resulting video bitstream is not modified, only the coding order of frames in the video sequence. However, as this coding order is changed, the decoder also needs to access the same indexing representation to reproduce the original visual order.

13.2 Indexing Representation Enhanced Coding Scheme

The basic idea of the proposed scheme is to use a high level indexing representation to create a shuffled video sequence where video segments are grouped together based on similarities. Some indexing representations can describe a sequence by segmenting it into video segments where each video segment can be of variable length from 1 frame to an entire shot. Take for instance the example of Figure 13.1 (composed of 1000 frames and 6 video segments from A to F). A high level indexing representation describes the sequence by creating, for instance, a hierarchical representation where video segments are grouped by similar characteristics, such as color and motion.

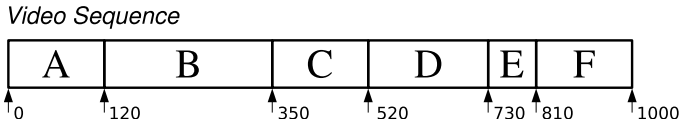


Figure 13.1: Initial video segmentation of a example sequence.

Figure 13.2 represents the indexing representation of the previous example sequence. Video segments are grouped into a hierarchical representation, a tree, and relationships between video segments are created. Section 13.3 reviews in detail the selected indexing representation used in this video segment shuffling scheme. For instance, in the example, video segments A and D share similar color and motion information and therefore are grouped in the indexing representation. In the Figure, nodes $\mathcal{A}$ and $\mathcal{D}$ are merged into $\mathcal{Z}$. Video segment F also shares similar color and motion information with video segments A and D so it is linked with both of them in node $\mathcal{X}$. The gray nodes in Figure 13.2 represent two or more grouped video segments. Hence, for instance, the node $\mathcal{W}$ represents the entire sequence. The indexing representation shows that video segments A and E are less similar, therefore, are linked together at the top of the hierarchical representation.

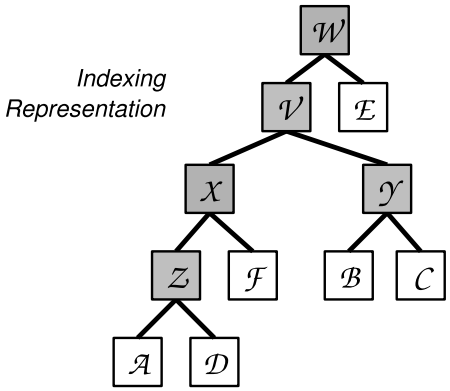


Figure 13.2: Indexing representation of the example sequence of Figure 13.1.

In order to create the shuffled sequence with the new coding order, video segments are grouped again in a time line sequence. The shuffled sequence groups together video segments with similar characteristics while the ones with less similarities are further apart in time. From the Figure 13.2, it can be seen that the leaves of the hierarchical representation describe the video segments of the sequence while the other nodes represent a merging of various video segments. Therefore, the shuffled sequence is created by visiting each node of the hierarchical representation, usually known as traversing a tree [47], but only considering the leaves of the hierarchical representation. The order in which the tree is traversed is not critical, the key point is to visit the nodes in such a way that sibling nodes are visited consecutively. For

instance, if the pre-order traversal is used [47], each node is visited before its children starting from the root node.

In the example of the Figure 13.2 the traversing of the indexing representation yields the sequence of nodes $\mathcal{W}, \mathcal{V}, \mathcal{X}, \mathcal{Z}, \mathcal{A}, \mathcal{D}, \mathcal{F}, \mathcal{Y}, \mathcal{B}, \mathcal{C}, \mathcal{E}$. To create the shuffled sequence, only the leaves are considered, which results in the sequence with video segments A, D, F, B, C, E. Figure 13.3 shows the final video segments that compose the time line of the shuffled sequence. As it can be seen, video segments with similar characteristics, such as A and D, are grouped together in the shuffled sequence whereas they were separated in the original sequence. However, if two video segments are not related to each other, such as A and B, they are separated in the final shuffled sequence. The proposed video segment shuffling scheme is expected to take profit of some of the new positions of the video segments in the shuffled video sequence, for instance the combination A and D.

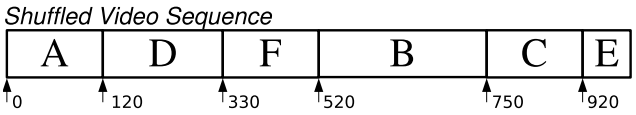


Figure 13.3: Final shuffled sequence using the indexing representation of Figure 13.2 to the initial video sequence in Figure 13.1.

13.3 Selected Indexing Representation

An indexing representation useful for this purpose is the MPEG-7 *Segment* DS. The *Segment DS* represents a spatial, temporal or spatio-temporal portion of the audiovisual content. The *Segment* DS can be organized into a hierarchical structure to produce a Table of Content for accessing or an Index for searching the multimedia content [82]. In that sense, the hierarchical representation is useful to provide functionalities of browsing and summarization of the video scene. Among other information, the *Segment* DS includes the following attributes:

- Location of the segment. In the case, for instance, of video segments the start time and the duration of the video segment is described.
- Media information of the segment. This includes information such as the type of content (for example if it is an audio or a video segment), the format of the segment, etc.
- Creation information of the segment.
- Semantic information to describe the scene that occurs in the segment.
- Structural relations of the segment between other segments. These relations provide the mechanism to create decompositions of segments into other sub-segments which

enables the creation of hierarchical representations to describe entire audio tracks or video sequences.

- Segments can be further described using MPEG-7 descriptors for color, texture, shape, motion, audio features, etc.

The XML schema syntax that defines the *Segment DS* is:

```
<!-- ################################################ -->
<!--  Definition of Segment DS                        -->
<!-- ################################################ -->
<complexType name="SegmentType" abstract="true">
  <complexContent>
    <extension base="mpeg7:DSType">
      <sequence>
        <choice minOccurs="0">
          <element name="MediaInformation" type="mpeg7:MediaInformationType"/>
          <element name="MediaInformationRef" type="mpeg7:ReferenceType"/>
          <element name="MediaLocator" type="mpeg7:MediaLocatorType"/>
        </choice>
        <element name="StructuralUnit" type="mpeg7:ControlledTermUseType" minOccurs="0"/>
        <choice minOccurs="0">
          <element name="CreationInformation" type="mpeg7:CreationInformationType"/>
          <element name="CreationInformationRef" type="mpeg7:ReferenceType"/>
        </choice>
        <choice minOccurs="0">
          <element name="UsageInformation" type="mpeg7:UsageInformationType"/>
          <element name="UsageInformationRef" type="mpeg7:ReferenceType"/>
        </choice>
        <element name="TextAnnotation" minOccurs="0" maxOccurs="unbounded">
          <complexType>
            <complexContent>
              <extension base="mpeg7:TextAnnotationType">
                <attribute name="type" use="optional">
                  <simpleType>
                    <union memberTypes="mpeg7:termAliasReferenceType
                      mpeg7:termURIReferenceType string"/>
                  </simpleType>
                </attribute>
              </extension>
            </complexContent>
          </complexType>
        </element>
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="Semantic" type="mpeg7:SemanticType"/>
          <element name="SemanticRef" type="mpeg7:ReferenceType"/>
        </choice>
        <element name="MatchingHint" type="mpeg7:MatchingHintType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="PointOfView" type="mpeg7:PointOfViewType"
          minOccurs="0" maxOccurs="unbounded"/>
        <element name="Relation" type="mpeg7:RelationType"
          minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The semantics of the *Segment DS* are:

SegmentType describes segments of audiovisual content.

MediaInformation describes the media information of the segment, this includes the type of content, format, etc. (optional)

MediaInformationRef describes a reference to the description of media information. (optional)

MediaLocator describes a locator of the segment. (optional)

StructuralUnit describes the role of the segment. Examples of values of *StructuralUnit* include "story", "scene", and "shot". The values may depend on the particular form of the multimedia content, e.g., movie or news report. (optional)

CreationInformation describes the creation information. (optional)

CreationInformationRef describes a reference to the description of creation information. (optional)

UsageInformation describes the usage information. (optional)

UsageInformationRef describes a reference to the description of usage information. (optional)

TextAnnotation describes text that annotates or describes the content of the segment. (optional)

type indicates the type or purpose of the textual annotation. Examples of values of type include "scene description", "transcription", and "color description". (optional)

Semantic describes the semantics of the scene depicted in the segment. (optional)

SemanticRef describes a reference to the description of semantic information. (optional)

MatchingHint describes the relative importance of descriptions in the *Segment DS* for matching segments. A segment description can include multiple *MatchingHint* Ds to describe multiple matching criteria or different combinations of descriptors. (optional)

PointOfView describes the relative importance of the segment given a specific point of view. (optional)

Relation describes a relation that the segment participates in. The relations include, for instance, structural relations. (optional)

In the case of the video segment shuffling scheme, a specific type of *Segment DS*, the *VideoSegment DS*, is used. The *VideoSegment* DS describes a video or a temporal segment of a video. In the case of digital video, a video segment can correspond to a single frame, an arbitrary group of frames, or the full video sequence. As the *Segment* DS, the video segment described by the *VideoSegment* DS does not need to be connected in time. A segment can also be decomposed into several segments creating hierarchical structures to describe the content. The XML schema syntax that defines the *VideoSegment* DS is:

```
<!-- ################################################ -->
<!--  Definition of VideoSegment DS                   -->
<!-- ################################################ -->
<complexType name="VideoSegmentType">
  <complexContent>
    <extension base="mpeg7:SegmentType">
      <sequence>
        <choice minOccurs="0">
          <element name="MediaTime" type="mpeg7:MediaTimeType"/>
          <element name="TemporalMask" type="mpeg7:TemporalMaskType"/>
        </choice>
```

```
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="VisualDescriptor" type="mpeg7:VisualDType"/>
          <element name="VisualDescriptionScheme" type="mpeg7:VisualDSType"/>
          <element name="VisualTimeSeriesDescriptor" type="mpeg7:VisualTimeSeriesType"/>
        </choice>
        <element name="MultipleView" type="mpeg7:MultipleViewType" minOccurs="0"/>
        <element name="Mosaic" type="mpeg7:MosaicType" minOccurs="0" maxOccurs="unbounded"/>
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="SpatialDecomposition"
            type="mpeg7:VideoSegmentSpatialDecompositionType"/>
          <element name="TemporalDecomposition"
            type="mpeg7:VideoSegmentTemporalDecompositionType"/>
          <element name="SpatioTemporalDecomposition"
            type="mpeg7:VideoSegmentSpatioTemporalDecompositionType"/>
          <element name="MediaSourceDecomposition"
            type="mpeg7:VideoSegmentMediaSourceDecompositionType"/>
        </choice>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

The semantics of the *VideoSegment DS* are:

VideoSegmentType describes a video or a temporal segment of a video. The temporal localization or composition of the video segment is optionally described using a choice of *MediaTime* or *TemporalMask*.

MediaTime describes the temporal localization of the video segment by specifying the start time and the duration of the video segment. If neither *MediaTime* nor *TemporalMask* is described, then the video segment refers to the entire video. (optional)

TemporalMask describes a temporal mask that defines the temporal composition of the video segment. If absent, the video segment is composed of the single connected interval defined by *MediaTime*. (optional)

VisualDescriptor describes visual features of the video segment using a visual descriptor. (optional)

VisualDescriptionScheme describes visual features of the video segment using a visual description scheme. (optional)

VisualTimeSeriesDescriptor describes a temporal sequence of visual features in the video segment. (optional)

MultipleView describes visual features of a 3D moving physical object depicted in the video segment as seen from one or more viewing positions or angles. (optional)

Mosaic describes a mosaic constructed from the video segment. (optional)

SpatialDecomposition describes a spatial decomposition of the video segment into one or more sub-segments. (optional)

TemporalDecomposition describes a temporal decomposition of the video segment into one or more sub-segments. (optional)

SpatioTemporalDecomposition describes a spatio-temporal decomposition of the video segment into one or more sub-segments. (optional)

MediaSourceDecomposition describes a media source decomposition of the video segment into one or more sub-segments. (optional)

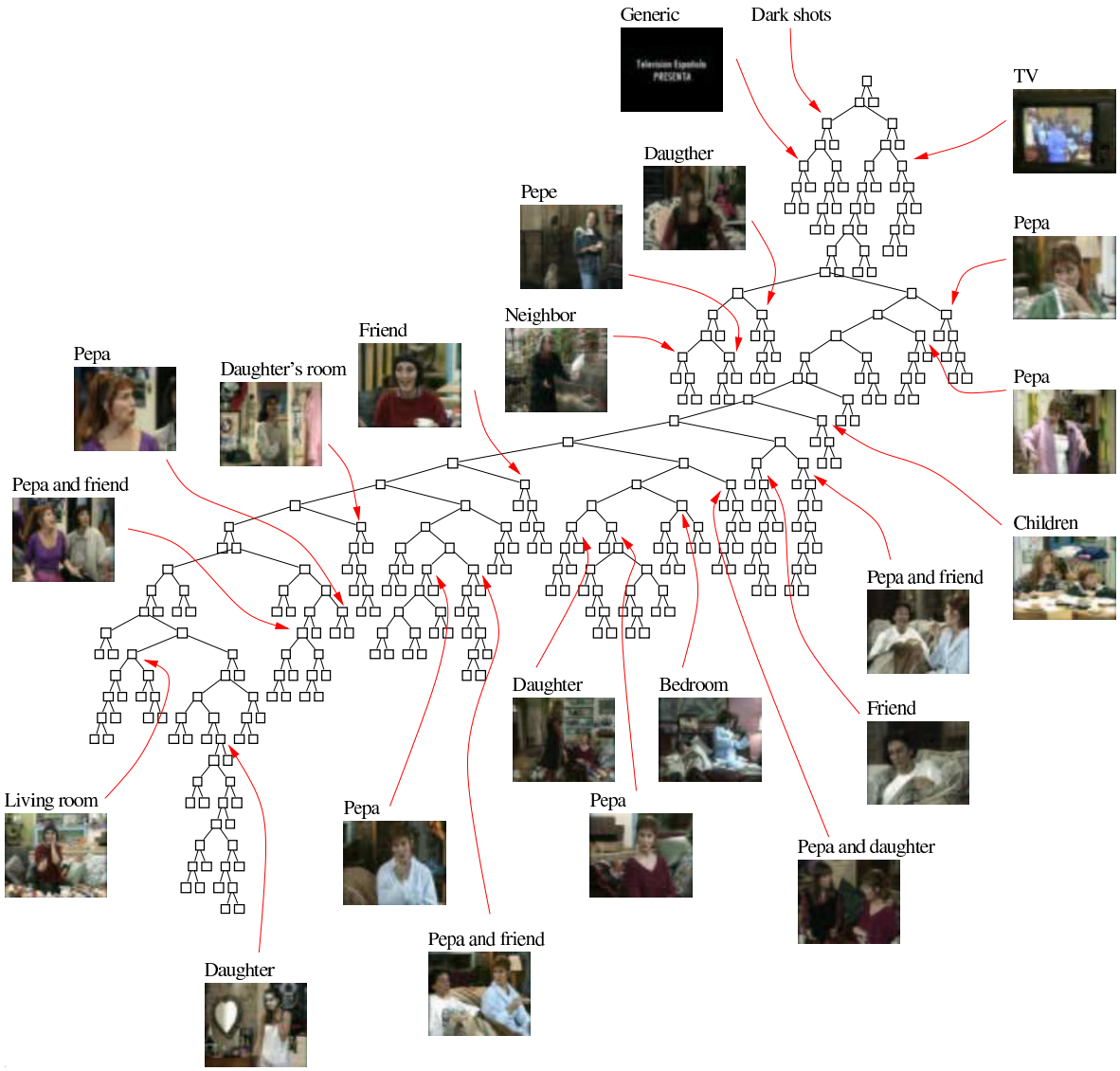


Figure 13.4: *VideoSegment* DS example of a video sequence. For some of the nodes of the indexing representation, the central frame of the described video segment is shown.

One possible indexing representation, which uses the *VideoSegment* DS, can be obtained by analyzing the entire video sequence and creating a hierarchical representation from an initial partition of the video sequence. The initial partition can be composed of single frames, of groups of L frames or even of video shots. Initially, all video segments belonging to the initial partition are considered and a merging criterion is used to group video segments that match a certain similarity criterion. Note that video segments do not have to be contiguous to be merged. One by one, all segments are merged with the most similar one (in terms of the chosen criterion). The hierarchical representation keeps information of all merging steps

that have been done until the entire video sequence is merged.

Figure 13.4 shows an example of the indexing representation of the video sequence *Drama* based on the *VideoSegment* DS. Several key-frames corresponding to some video segments of the original sequence can be seen in Figure 13.5. In the example, the initial partition is composed of all individual shots of the sequence, hence, each video segment corresponds to a shot. As the merging criterion only merges two different regions at a time, the resulting structure is a binary tree called BPT (Binary Partition Tree). The specific details that have been followed for the creation of the BPT are given in [55]. In the example of Figure 13.4, similar shots are grouped in the same branches of the hierarchical representation. For instance, all video segments composed of dark frames are grouped into the top left branch. If a new sequence is constructed by traversing the hierarchical representation from the root node, the resulting sequence contains the same number of frames as the initial one. In this case, however, similar frames, in terms of color and motion similarity, will be closer in time. Figure 13.6 shows the same video segments as Figure 13.5 for the shuffled sequence. It can be seen that, effectively, video segments with similar content are closer in time.

13.4 Experimental Results

Several experiments are performed to assess the efficiency of the video segment shuffling scheme. Table 13.1 lists the test sequences employed in the experimental tests. The test set is composed of the same sequences employed for the experimental results of the selection of reference frames in Section 12.4. The sequence *La nit al dia* is also included in the test set. The *La nit al dia* sequence is composed of 8000 frames of a news program with a mixture of low and medium motion shots (several key-frames of the test sequence are shown in Figure B.22).

Name	Frames	Resolution	fps	Motion	Type of content
Geri	364	176×92	30	high	video clip
Telediario	3000	QCIF	25	low,high	news
Jornal da noite	7000	QCIF	25	low,medium	news
Formula1	2500	QCIF,CIF	25	high	sports
Fr3	3500	QCIF,CIF	25	medium	documentary
Àgora	14000	QCIF,CIF	25	low	interview
La nit al dia	8000	QCIF,CIF	25	low,medium	news

Table 13.1: Sequences used for the experimental tests of the video segment shuffling scheme.

Comparisons are made by encoding the original test sequences and the shuffled versions with the standard H.264 codec. The conditions and settings for the H.264 video codec are the same as the ones in previous sections and detailed in Table 11.2. In all experiments,

VS #01 (0-277) VS #02 (278-583) VS #03 (584-884) VS #04 (885-927) VS #05 (928-1312)

VS #06 (1313-1366) VS #07 (1367-1729) VS #08 (1730-1884) VS #09 (1885-2055) VS #10 (2056-2300)

VS #11 (2301-2723) VS #12 (2724-3190) VS #13 (3191-3399) VS #14 (3400-3435) VS #15 (3436-3507)

VS #16 (3508-3568) VS #17 (3569-3617) VS #18 (3618-3786) VS #19 (3787-3870) VS #20 (3871-3972)

VS #21 (3973-4032) VS #22 (4033-4226) VS #23 (4227-4315) VS #24 (4316-4796) VS #25 (4797-5522)

VS #26 (5523-5587) VS #27 (5588-5866) VS #28 (5867-6019) VS #29 (6020-6064) VS #30 (6065-6279)

VS #31 (6280-6808) VS #32 (6809-6872) VS #33 (6873-7380) VS #34 (7381-8051) VS #35 (8052-8094)

Figure 13.5: Several video segments, corresponding to complete shots, of the *Drama* test sequence. Each key-frame represents the central frame of the corresponding video segment (VS #). The numbers in brackets show its starting and ending time instants. Video segments are ordered from left to right and from top to bottom as they appear in the original sequence.

Figure 13.6: Several video segments, corresponding to complete shots, of the shuffled version of the *Drama* test sequence. As in Figure 13.5, each key-frame represents the central frame of the corresponding video segment. Video segments are ordered from left to right and from top to bottom as they appear in the shuffled sequence.

the previous 5 reference frames are used to fill the long term prediction buffer **LT**. The shuffled versions of the sequences are created using the indexing representation as explained in Section 13.2. It seems obvious that the shuffling of video segments will be more efficient for test sequences with a high number of similar shots scattered through the overall sequence as the indexing representation is able to group them in the same branches of the tree. In the first experimental test, the effect of the initial partition of the indexing representation is analyzed. Three different initial partitions are studied:

- $S = 1$ Initial partition using video segments of a fixed size of 1 frame (frame shuffling)
- $S = 8$ Initial partition using video segments of a fixed size of 8 frames
- $S = N$ Initial partition using video segments of variable size corresponding to complete shots of the sequence (shot shuffling)

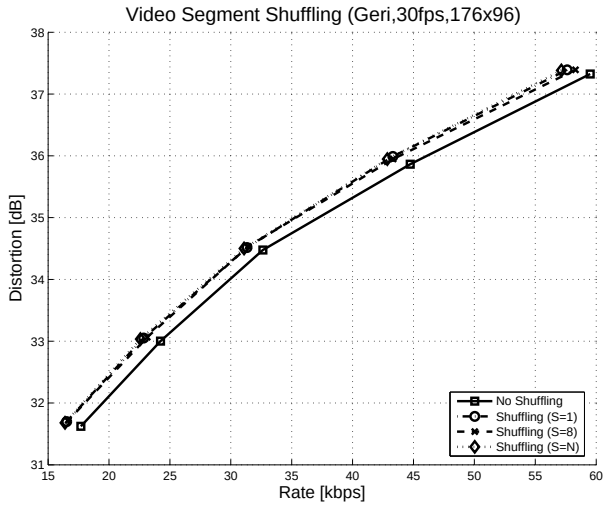


Figure 13.7: Rate-distortion curves using the original and the shuffled versions of the test sequence *Geri*. Three different initial partitions are studied: using video segments of size 1 frame ($S = 1$), 8 frames ($S = 8$) and complete shots ($S = N$).

Figure 13.7 shows the rate-distortion curves obtained when coding the test sequence *Geri* using the original and shuffled sequences. Three versions of the shuffled sequence are created using the three initial partitions detailed above. Although very similar, the best results are obtained when coding the shuffled sequence using complete shots as initial segments in the indexing representation (rate-distortion curve with legend *Shuffling ($S = N$)* in Figure 13.7) with PSNR gains of 0.6 dB at the same bitrate compared to the coding of the original sequence (rate-distortion curve with legend *No Shuffling*). Also, similar results are obtained for the other test sequences. This is not surprising as, in general, the gains in rate-distortion obtained by the H.264 codec for the shuffled sequence are concentrated at the beginnings of the shots

where the shuffled sequence is able to group similar shots together and therefore it is easier to find better reference frames there than in the original sequence. Also, using complete shots as initial video segments reduces the complexity of the indexing representation, as there are fewer nodes in the hierarchical representation, and, as a result, the final traversing of the tree is simpler.

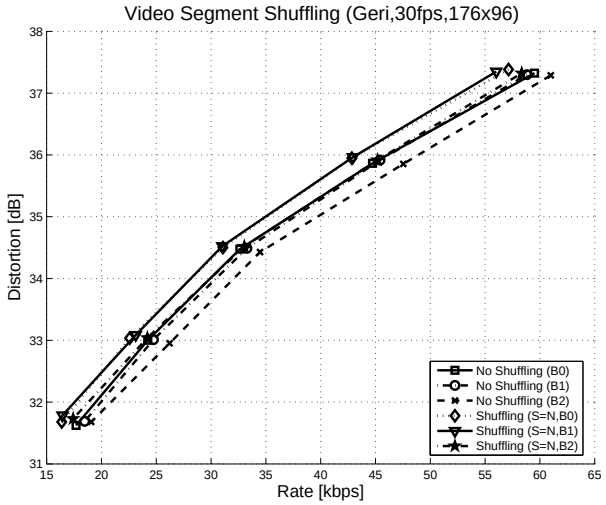


Figure 13.8: Rate-distortion curves using the original and the shuffled versions of the test sequence *Geri*. Different number of B frames in the GoP structure are tested; $B0$ (only P frames), $B1$ (1 B frame between P frames) and $B2$ (2 B frames between P frames). All three experiments use complete shots as the initial partition ($S = N$).

Figure 13.8 is devoted to the study of the effect of different GoP structures in the coding of the shuffled sequence. Three GoP structures are considered using a different number of B frames between P frames in the GoP. In all three experiments, the initial partition consists of video segments which correspond to shots ($S = N$). As shown in the Figure, similar improvements are obtained by using the video segment shuffling scheme for all B configurations.

Figure 13.9 shows the results for the sequence *Jornal da noite* in the MPEG-7 test set. The Figure is used to study two different configurations of the H.264 codec, $B0$ and $B1$ GoP structures. The $B2$ configuration is removed as it obtained similar results. The shuffled sequence is obtained using complete shots as the initial partition of the indexing representation ($S = N$). The use of the shot shuffling scheme results in bitrate savings for both GoP structures. Experimental results show lower bitrate savings (up to 0.5 dB for the same bitrate) than the results obtained for the *Geri* sequence. This is because *Jornal da noite* has longer shots and the improvement achieved by the shuffling has less impact on global averages. However, the coding efficiency around the starting frames of shots improves considerably.

The same experiments are performed for the test sequence *Telediario* in Figure 13.10.

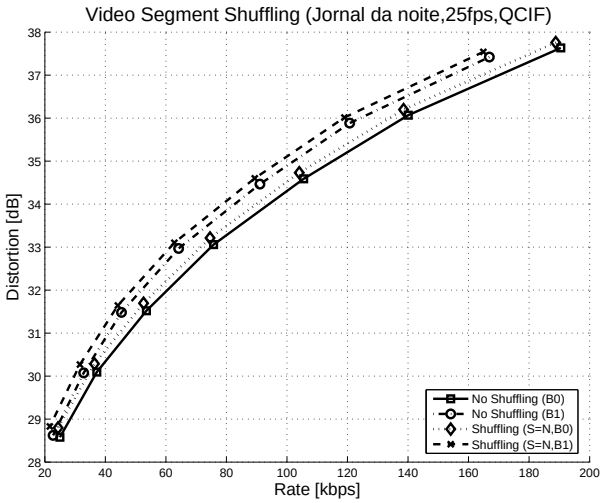


Figure 13.9: Rate-distortion curves using the original and the shuffled versions of the test sequence *Jornal da noite*. Experiments are run with using complete shots as the initial partition. Two GoP configurations are tested, $B0$ (only P frames) and $B1$ (1 B frame between P frames) s).

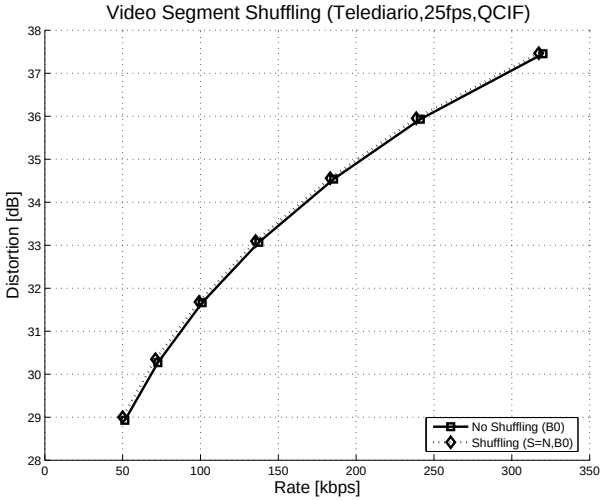
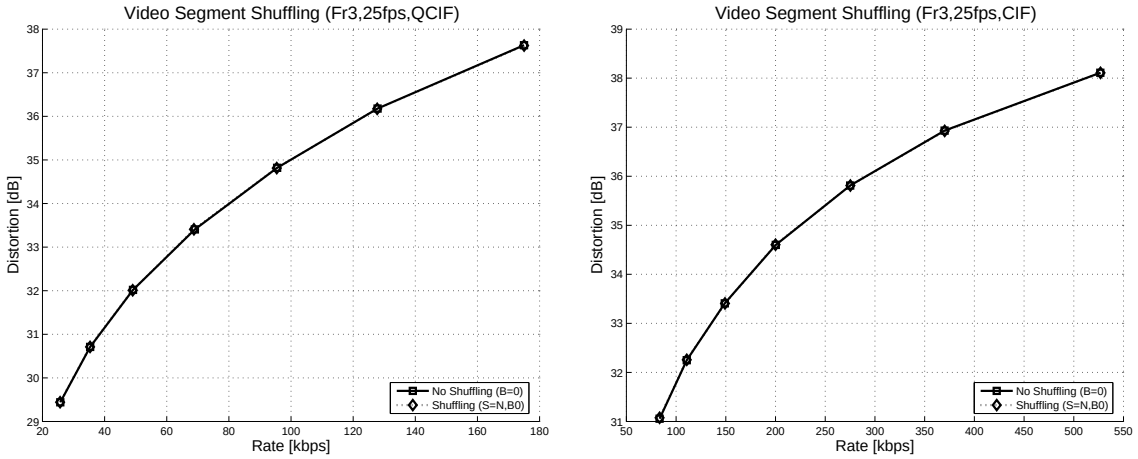


Figure 13.10: Rate-distortion curves using the original and the shuffled versions of the test sequence *Telediario*. Experiments are run with the $B0$ GoP configuration and using the shot shuffling scheme ($S = N$).

This time, only the $B0$ configuration is shown as similar results are obtained for the $B1$ and $B2$ cases. In particular, the coding efficiency using the video segment shuffling scheme is the same as the one obtained without using it. These results can be explained by the type of test sequence. The *Telediario* test sequence has long shots with almost no repetition. Even though, some shots are visually similar, the test sequence includes various football shots, for

instance, the resulting shuffled sequence does not help in exploiting the temporal redundancy by the H.264 codec.



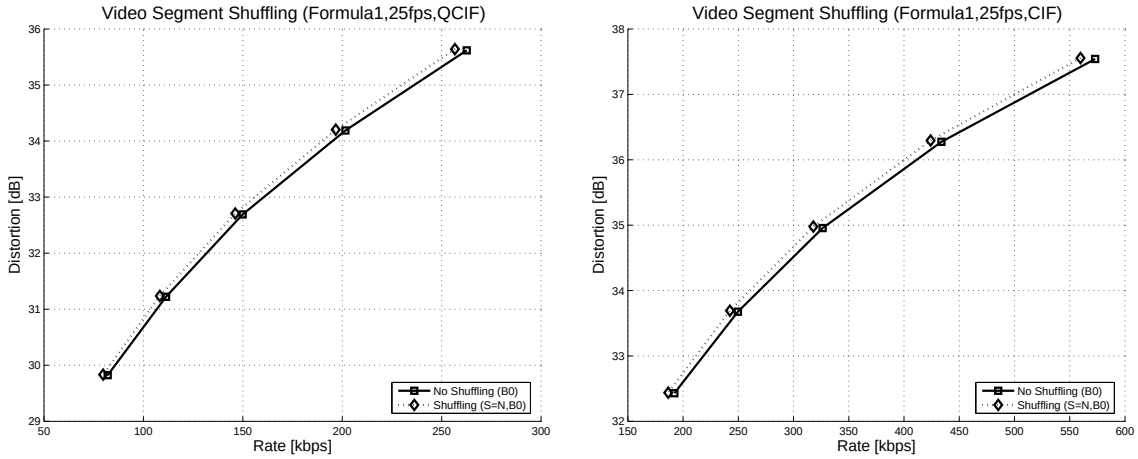
a) QCIF b) CIF

Figure 13.11: Rate-distortion curves using the original and the shuffled versions of the test sequence *Fr3*. Experiments are run with the $B0$ GoP configuration and using the shot shuffling scheme ($S = N$).

A similar problem appears with test sequence *Fr3* and the proposed video segment shuffling scheme does not improve the coding efficiency of the H.264 codec. Figure 13.11 shows the rate-distortion curves when coding both the original and the shuffled version of the test sequence at QCIF and CIF resolution. In both resolutions, the final efficiency of the video segment shuffling scheme is similar to the original, non-shuffled, sequence. Again, this is explained by the fact that the *Fr3* test sequence is composed of long, non-repeated shots, and, therefore, the video segment shuffling scheme is not able to group similar frames together.

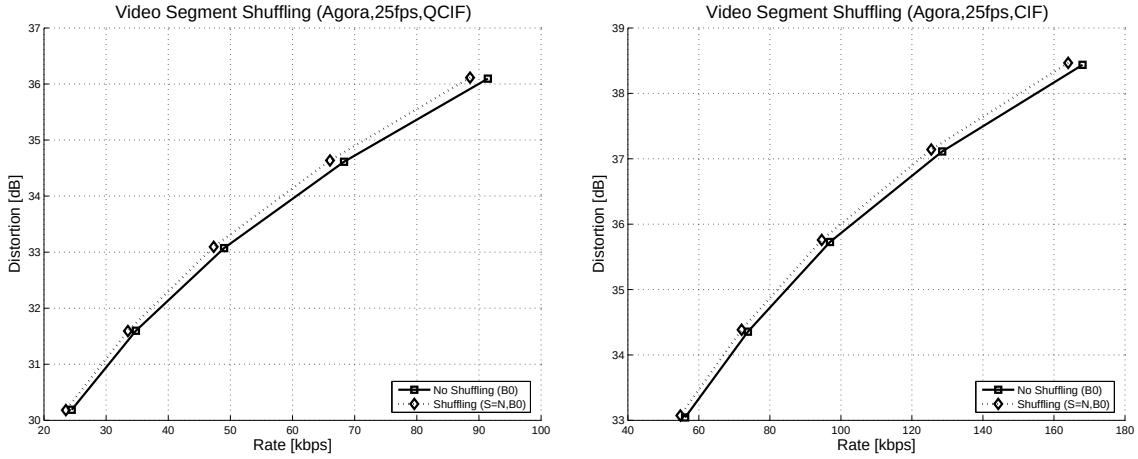
In the case of the *Formula1* test sequence, however, the video segment shuffling scheme is able to improve the efficiency of the H.264 codec. Figures 13.12.a and 13.12.b show the rate-distortion curves when coding both the original and shuffled versions of the *Formula1* test sequence at QCIF and CIF resolution respectively. The sequence *Formula1* is composed of short shots with some repetition between them. The selected indexing representation is able to create a useful hierarchical representation to be used by the shot shuffling and, finally, the video segment shuffling scheme is able to obtain small gains of up to 0.3 dB for both QCIF and CIF resolutions.

Figure 13.13 shows the same experiments but for the *Àgora* test sequence. In this test, the video segment shuffling scheme also provides similar gains than the ones on previous tests. The similar shots on the *Àgora* sequence are grouped by the indexing representation and, therefore, the H.264 codec is able to better exploit the temporal redundancy of the test sequence.



a) QCIF b) CIF

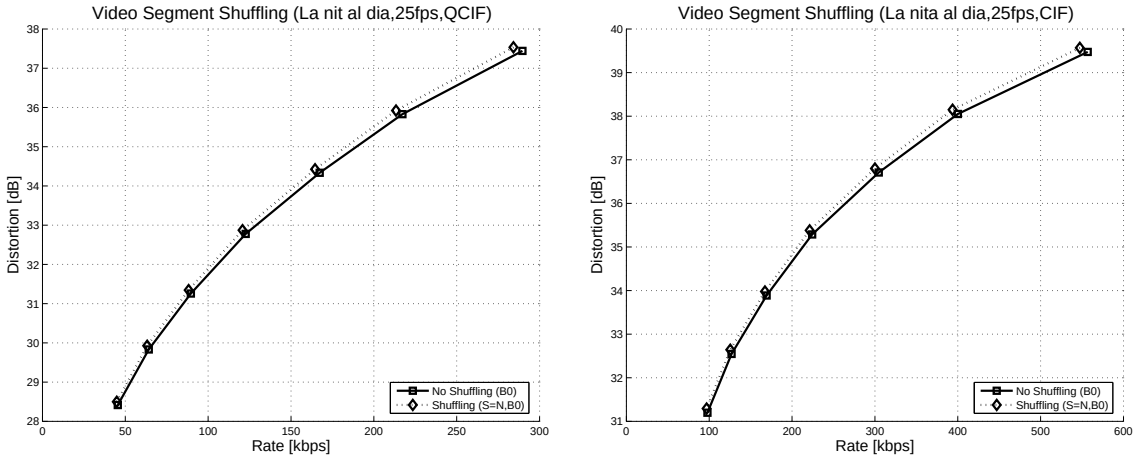
Figure 13.12: Rate-distortion curves using the original and the shuffled versions of the test sequence *Formula1* at QCIF and CIF resolutions. Experiments are run with the $B0$ GoP configuration and using the shot shuffling scheme ($S = N$).



a) QCIF b) CIF

Figure 13.13: Rate-distortion curves using the original and the shuffled versions of the test sequence *Àgora* at QCIF and CIF resolutions. Experiments are run with the $B0$ GoP configuration and using the shot shuffling scheme ($S = N$).

Finally, the *La nit al dia* test sequence is used for the experiments. Again, the video segment shuffling scheme provides small gains when coding the test sequence. Figure 13.14 shows the rate-distortion curves of the H.264 codec when coding the original sequence and the shuffled version. The PSNR gains obtained when using the shuffled sequence are similar between QCIF and CIF resolutions and consist on an average of 0.3 dB in PSNR when coding



a) QCIF b) CIF

Figure 13.14: Rate-distortion curves using the original and the shuffled versions of the test sequence *La nit al dia* at QCIF and CIF resolutions. Experiments are run with the $B0$ GoP configuration and using the shot shuffling scheme ($S = N$).

at the same bitrate.

13.5 Indexing Representation not Available at the Decoder

In the applications where the video segment shuffling scheme might be used, principally storing, it makes sense to have an indexing representation of the sequence to allow quick access to any part of it. For this reason, previous results do not include the bitrate needed to encode the *VideoSegment* DS. However, in the case where the indexing representation is not available at the decoder end, some information must be streamed with the content. As sending the entire hierarchical representation to only create the shuffled sequence may not be very efficient, a different coding method is proposed.

In order to re-organize the shuffled sequence in the original visual order, we only need to send a list specifying the video segments positions and the place where they should be moved to. This list is called shuffling information. Consider, for example, the video sequence segmented into shots as seen in Figure 13.1. The proposed shuffled sequence constructed from the indexing representation is shown in Figure 13.3.

At the decoder side, video segments are arranged in the visual order again. In order to perform such arrangement, a list containing the location of the first video segment frame before and after applying the shuffling scheme (Table 13.2) is needed. Finally, after each video segment is decoded, it is moved to its marked place. Video segment by video segment, the sequence is recovered as Figure 13.15 shows.

Segment	Shuffled seq. (1^{st} frame)	Original seq. (1^{st} frame)
A	0	0
D	120	520
F	330	810
B	520	120
C	750	350
E	920	730

Table 13.2: Shuffling information for the sequence described in 13.1.

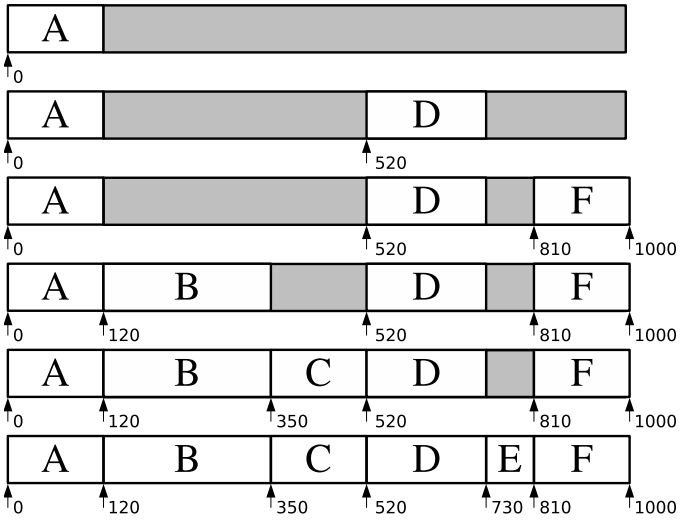


Figure 13.15: Sequence for recovering the shuffled video segments into the original display order.

The amount of bits necessary to address any frame of the entire sequence is $\lceil \log_2 L \rceil$, where L is the length in frames of the sequence. Consequently, the total number of bits necessary to encode the markers list is $2N\lceil \log_2 L \rceil$ (where N is the number of video segments). Dividing by L, we obtain the average bits/frame that are required in order to send the information needed to recover the original visual order:

$$\frac{2 \cdot N \cdot \lceil \log_2 L \rceil}{L} = \frac{2 \cdot \lceil \log_2 L \rceil}{L/N} \tag{13.1}$$

The smaller L/N is, the larger would be the amount of bits required. But a small L/N also means a large number of scattered small shots in the original sequence, hence, the bitrate savings achieved by the video segment shuffling scheme will be higher. Additionally, if two or more contiguous video segments are not separated by the shuffling, they can be considered as a single segment and only one position on the list is required.

Test	Data H.264 [kbps]	Savings [kbps]	Distortion [dB]	Shuffling Information [kbps]
Geri	18.48	2.03 (11.0%)	0.10	0.012 (0.06%)
	58.82	2.79 (4.8%)	0.05	0.012 (0.02%)
Jornal da noite	22.74	1.08 (4.8%)	0.21	0.003 (0.01%)
	166.89	1.99 (1.2%)	0.12	0.003 (0.00%)
Telediario	51.43	1.46 (2.8%)	0.07	0.005 (0.01%)
	320.06	2.76 (0.9%)	0.01	0.005 (0.00%)
Fr3	25.74	0.05 (0.2%)	0.00	0.004 (0.02%)
	174.99	−0.04 (0.0%)	−0.01	0.004 (0.00%)
Formula1	82.07	2.28 (2.8%)	0.01	0.010 (0.01%)
	262.60	5.89 (2.2%)	0.02	0.010 (0.00%)
Àgora	24.47	0.94 (3.8%)	−0.01	0.002 (0.01%)
	91.44	2.86 (3.1%)	0.02	0.002 (0.00%)
La nit al dia	45.53	0.86 (1.9%)	0.08	0.004 (0.01%)
	289.88	4.34 (1.5%)	0.09	0.004 (0.00%)

Table 13.3: Comparisons between the data bitrate and the bitrate needed to send the information to recover the original visual order for video segment shuffling scheme.

If the shuffling information has to be sent, the worst case to study is the lowest bitrate and resolution one, because the absolute number of bits saved is lower, but the number of bits required for encoding the shuffling information is the same at all qualities. As for previous schemes, the methodology of presenting the results of this section is changed so the small differences in bitrate between sending or not the shuffling information can be easily assessed. Table 13.3 shows the bitrate needed to send the video data and the shuffling information (at low and high bitrates) in the test sequences (which is the same for all resolutions). The first column in Table 13.3 lists the data bitrate for the standard H.264 codec when coding the original (non-shuffled) test sequence. The second column details the total savings when using the selected indexing representation to shuffle the video segments of the test sequences. As for previous experimental results, complete shots, $S = N$, are used as the initial partition of the indexing representation. In brackets the savings percentage with respect to the first column is shown. The third column reports the distortion difference between the shuffled version and the original version of the test sequence. Finally, the last column shows the bitrate needed to stream the shuffling information (in brackets the percentage with respect to data bitrate in first column).

In general, sending the shuffling information only represents up to 0.06% of the total bitrate. Therefore, it does not greatly modify the achieved gains of the video segment shuffling scheme. For instance, in the case of the sequence *Geri*, the bitrate savings are 11% with respect to H.264 while the penalty of sending the extra information is 0.06% of the data bitrate. In the case of the sequence *Jornal da noite* the penalty of sending the shuffling information is

0.01% with a bitrate savings of 4.8% at low resolution. Results in Table 13.3 show that even if the indexing representation is not available in the decoder, the shuffling information can be streamed together with the content with almost no penalty in bitrate savings, thus, the decoder can recover the original display order.

In conclusion, results for this scheme show interesting gains (bitrate savings up to 11% for the same quality factor) in sequences that present scattered, short and similar shots (common in television programs such as interviews, video clips, etc.). In other kind of sequences, the efficiency gain is relatively small or nothing; but video segment shuffling could be a convenient way of storing a sequence, i.e. to ease the access through a *VideoSegment* DS. For all experiments, using complete shots as the initial partition, $S = N$, performs better than any other video segment size.

Chapter 14

Frame Type Selection

This section is devoted to the use of indexing representations to improve the frame type selection in GoP structures. The frame type selection scheme is based on a low-level indexing representation which is used to select a fixed GoP structure to encode the sequence.

14.1 Motivation

The motivation of this approach relies on improving the frame type selection of hybrid codecs based on the type of content to be coded. In current hybrid video codecs, video sequences are coded using group of pictures (GoP) with a fixed GoP structure through the entire video. For instance, a common GoP structure consists of using two B frames between P frames, hence, the sequence of frame types is $IBBPBBPB \cdots BPBBP$ for this GoP structure. Another common GoP structure consists of using no B frames so the sequence of frame types is $IPPPP \cdots PPP$. In general, the number of B frames between P frames defines the different GoP structures of the video sequence.

The GoP structure, however, does not need to be static as in the above structures. Several techniques reported in the literature [51, 97] aim at reducing the bitrate associated to a GoP by carefully choosing the frame type to be used for each individual frame within the GoP. However, for these techniques, the relationship between coded P and B frames makes the frame type selection a very challenging problem as future decisions will depend on current ones. Often, finding the best frame type for a specific frame in the GoP involves searching on a very large space of possible frame type combinations.

In this section, a scheme for selecting the frame type in GoPs is presented. Again, indexing representations, even if extracted with indexing functionality in mind, can help to the frame type selection decision. For instance, some indexing representations based on motion may describe a video scene by indicating the presence of high or low internal motion in the sequence. This can be used to select different GoP structures according to the amount of motion present

in the video scene.

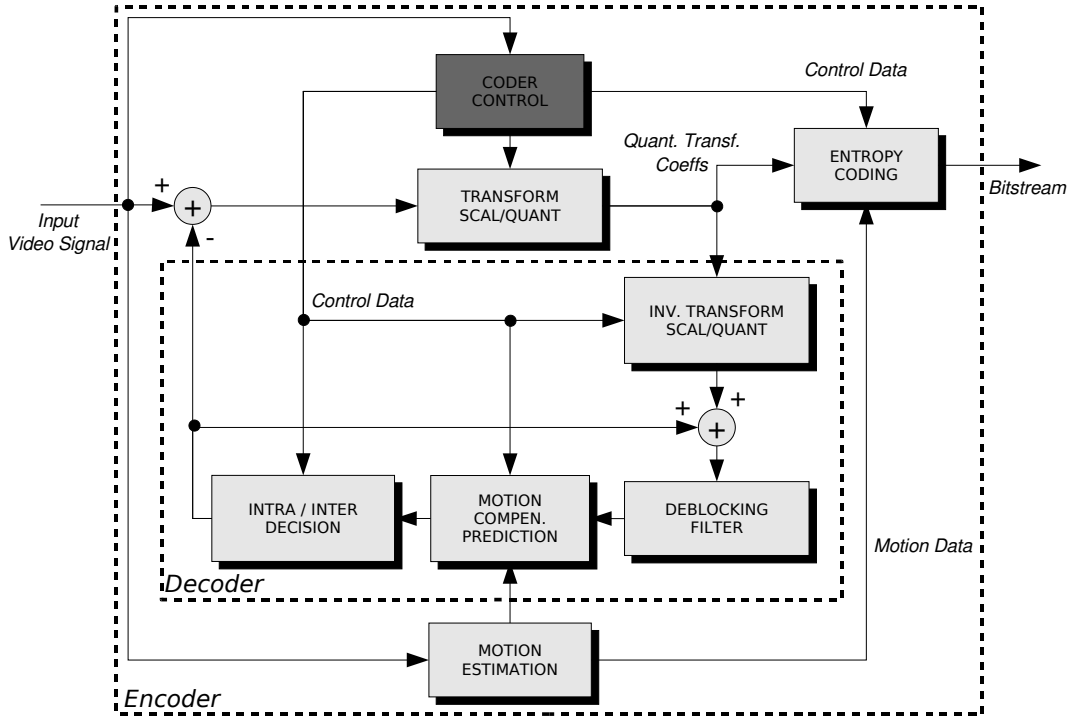


Figure 14.1: Indexing representation enhanced H.264 coding scheme. In dark gray the module that is modified in order to exploit the indexing representation when selecting the frame type in GoP structures.

If the proposed indexing representation enhanced scheme for the frame type selection of GoP structures is introduced in standard hybrid codecs, only the coder control module (highlighted in Figure 14.1) must be modified. The control module is responsible, among other things, for selecting the frame type of coded frames. If the encoder is aware of already existing indexing representations attached to the video content, the selection can be performed by carefully inspecting the information provided by the indexing representation. As the only modified module is not part of the decoder, the resulting bitstream is still compatible with standard decoders and the bitstream can be decoded without any use of the indexing representations.

14.2 Indexing Representation Enhanced Coding Scheme

The selection strategy which is used in here is based on the assumption that sequences with very high internal motion are better encoded using a low number of B frames. When internal

motion is rather low, the reference frames (previous and past) used to predict the current B frame are very similar and the B frame can be predicted with almost no error. In the presence of high internal motion, the reference frames can change considerably and the quality of the predicted B frame decreases. Indexing representation based on motion can inform the encoder whether to use more B frames when less motion is present in the sequence.

In general, choosing a specific frame type on a frame by frame basis is a very difficult problem as current decisions affect future ones (due to the relation between P and B frames). To simplify the frame selection, the input sequence is divided into GoPs, video segments of a fixed size of L frames. For each GoP, a fixed GoP structure is employed according to the indexing representation information about motion. 4 different GoP structures are chosen based on different number of B frames between P frames. Table 14.1 lists the 4 different GoP structures used in the proposed frame type selection. The next section reviews the mapping between the selected indexing representation and the GoP structures detailed in Table 14.1.

B frames	Frame types	GoP structure
0	$\cdots PPPPPPPPP \cdots$	B0
1	$\cdots PBPBPBPBP \cdots$	B1
2	$\cdots PBBPBBPBB \cdots$	B2
3	$\cdots PBBBPBBBP \cdots$	B3

Table 14.1: Fixed GoP structures for frame type selection using indexing representations.

Any frame size L of the GoP could be used for the frame type selection, from a small number of frames to a large number. Also, GoPs do not need to be of the same size through the sequence and can be, for instance, created from the video shots, thus, each GoP could correspond to a shot of the sequence. In the experimental tests of Section 14.4, GoPs of size 8 frames and 16 frames are studied.

14.3 Selected Indexing Representation

An indexing representation useful for this proposed scheme is the MPEG-7 *Motion Activity* D. This descriptor indicates the amount of internal motion present in a video sequence. For instance, sequences with a high amount of internal motion include scenes such as a goal scoring in a soccer match or a high speed car chase. On the other hand, scenes such as a news reader or an interview represent sequences with a low activity of motion. The *Motion Activity* D includes the following five attributes:

- Intensity of Activity expresses the amount of internal motion. A high value of intensity indicates high activity while a low value of intensity indicates low activity.

- Direction of Activity indicates the dominant direction of the internal motion if any. The dominant direction indicates the main direction followed by the majority of the objects in the scene.

- Spatial Distribution of Activity indicates whether the activity is spread across many regions or restricted to one large region. It is an indication of the number and size of "active" regions in a frame. For example, a talking head sequence would have one large active region, while an aerial shot of a busy street would have many small active regions.

- Spatial Localization of Activity expresses spatial distribution of motion intensities over the duration of the video segment. For example, a video segment that has high motion activity in the left side can be categorized or retrieved.

- Temporal Distribution of Activity expresses the variation of activity over the duration of the video segment. In other words, whether the activity is sustained throughout the duration of the sequence or confined to a temporal part.

The XML schema syntax that describes the definition of the *Motion Activity* D is:

```
<!-- ################################################ -->
<!-- Definition of MotionActivity D                   -->
<!-- ################################################ -->
<complexType name="MotionActivityType" final="#all">
  <complexContent>
    <extension base="mpeg7:VisualDType">
      <sequence>
        <element name="Intensity">
          <simpleType>
            <restriction base="mpeg7:unsigned3">
              <minInclusive value="1"/>
              <maxInclusive value="5"/>
            </restriction>
          </simpleType>
        </element>
        <element name="DominantDirection" type="mpeg7:unsigned3" minOccurs="0"/>
        <element name="SpatialDistributionParams" minOccurs="0">
          <complexType>
            <attribute name="numOfShortRuns" type="mpeg7:unsigned6" use="required"/>
            <attribute name="numOfMediumRuns" type="mpeg7:unsigned5" use="required"/>
            <attribute name="numOfLongRuns" type="mpeg7:unsigned5" use="required"/>
          </complexType>
        </element>
        <element name="SpatialLocalizationParams" minOccurs="0">
          <complexType>
            <choice>
              <element name="Vector4">
                <simpleType>
                  <restriction>
                    <simpleType>
                      <list itemType="mpeg7:unsigned3"/>
                    </simpleType>
                    <length value="4"/>
                  </restriction>
                </simpleType>
              </element>
              <element name="Vector16">
```

```
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned3"/>
                </simpleType>
                <length value="16"/>
              </restriction>
            </simpleType>
          </element>
          <element name="Vector64">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned3"/>
                </simpleType>
                <length value="64"/>
              </restriction>
            </simpleType>
          </element>
          <element name="Vector256">
            <simpleType>
              <restriction>
                <simpleType>
                  <list itemType="mpeg7:unsigned3"/>
                </simpleType>
                <length value="256"/>
              </restriction>
            </simpleType>
          </element>
        </choice>
      </complexType>
    </element>
    <element name="TemporalParams" minOccurs="0">
      <simpleType>
        <restriction>
          <simpleType>
            <list itemType="mpeg7:unsigned6"/>
          </simpleType>
          <length value="5"/>
        </restriction>
      </simpleType>
    </element>
  </sequence>
 </extension>
</complexContent>
</complexType>
```

The semantics of the *MotionActivityType* are:

Intensity This element is expressed by an integer lying in the range $[1, 5]$. The value of 1 specifies the lowest intensity, whereas the value of 5 specifies the highest intensity.

DominantDirection This element specifies the dominant direction and is expressed as an angle between 0 and 360 degrees.

SpatialDistributionParams consists of three fields: *numOfShortRuns*, *numOfMediumRuns* and *numOfLongRuns*, which specify the numbers of short, medium and long runs of zeros, respectively. Short, medium and long runs of zeros are computed from the motion vectors of a given frame and consist of a histogram with the number of times that these short, medium and long runs of zeros appear in the motion vector field.

SpatialLocalizationParams This element specifies the relative activity of specific regions of the sequence. The activity for each region is defined as the average of motion vector

magnitudes in each region. Relative activity is the ratio of the activity of the region to the sum of activities in the shot.

TemporalParams This is a histogram consisting of 5 bins, where histogram bins with indexes $0, 1, 2, 3, 4$ correspond to *Intensity* values of $1, 2, 3, 4, 5$ respectively. The histogram expresses the relative frequency of different levels of activity in the sequence as defined by the intensity element above. Each value is the percentage of occurrences of the corresponding quantized intensity level uniformly quantized to 6 bits.

In the case of the frame type selection, the *Intensity* attribute of the descriptor is the most useful one. This field indicates the amount of internal motion of the video segment by a quantized scale in the $[1, 5]$ range. The value is computed by thresholding the standard deviation of the motion vector magnitudes of a given frame [14, 74, 92]. If the image frame is of size $W \times H$ in pixels, f is the rate measured in frames per second of the video and σ denotes the standard deviation of all motion vector magnitudes for the given frame, the quantification can be computed using the following thresholds:

$$\begin{aligned} Intensity &= 1 \quad if \quad \sigma < t_1 \\ Intensity &= 2 \quad if \quad t_1 < \sigma < t_2 \\ Intensity &= 3 \quad if \quad t_2 < \sigma < t_3 \\ Intensity &= 4 \quad if \quad t_3 < \sigma < t_4 \\ Intensity &= 5 \quad if \quad \sigma > t_4 \end{aligned} \tag{14.1}$$

where the thresholds t_1, t_2, t_3, t_4 are calculated as:

$$\begin{aligned} t_1 &= 0.0857 \cdot \frac{\sqrt{W^2 + H^2}}{f} \\ t_2 &= 0.2353 \cdot \frac{\sqrt{W^2 + H^2}}{f} \\ t_3 &= 0.4267 \cdot \frac{\sqrt{W^2 + H^2}}{f} \\ t_4 &= 0.7037 \cdot \frac{\sqrt{W^2 + H^2}}{f} \end{aligned} \tag{14.2}$$

A very simple mapping can be performed between the intensity values and the number of B frames between P frames in the GoP structure. For instance, Table 14.2 lists a possible mapping between the different motion intensities of the *Motion Activity* D into the 4 fixed GoP structures of Table 14.1. Experimental tests of Section 14.4 use this mapping to preselect the GoP structures to encode each GoP of the test sequence. The intensities used in Table 14.2 are the quantized mean value of the *Motion Activity* intensities of all image frames that compose the analyzed GoP.

Intensity	GoP structure
1	B3
2	B3
3	B2
4	B1
5	B0

Table 14.2: Mapping between the *Motion Activity* intensity attribute and the GoP structure to be used in the encoding.

14.4 Experimental Results

Experimental tests are performed to analyze the efficiency of the proposed indexing representation enhanced scheme. The same test sequences as in previous sections are employed. Table 14.3 lists the properties of the video sequences used in the experiments. A mixture of sequences with high and low internal motion at QCIF and CIF resolution is employed.

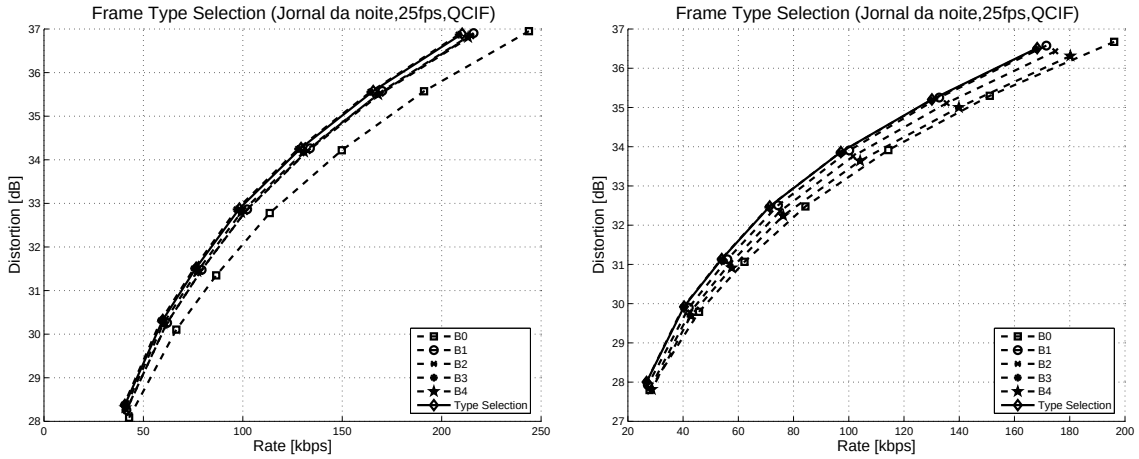
Name	Frames	Resolution	fps	Motion	Type of content
Jornal da noite	3000	QCIF	25	low,medium	news
Telediario	3000	QCIF	25	low,high	news
Formula1	2500	QCIF,CIF	25	high	sports
Fr3	5000	QCIF,CIF	25	medium	documentary
Àgora	5000	QCIF,CIF	25	low	interview
La nit al dia	5000	QCIF,CIF	25	low,medium	news

Table 14.3: Sequences used for the experimental tests of the type frame selection scheme.

In order to assess the efficiency of the proposed scheme, test sequences are encoded with the standard H.264 codec using the same fixed GoP structure for all the frames of the sequence. Sequences are coded using 5 fixed GoP structures by changing the number of B frames between P frames in the GoP from a value of 0 to 4. In the following Figures, legends *B0*, *B1*, *B2*, *B3* and *B4* are used to indicate the rate-distortion curves of the standard H.264 codec using the indicated GoP structures for all GoPs of the sequence. The conditions and settings for the H.264 video codec are the same as these of previous experiments and indicated in Table 11.2.

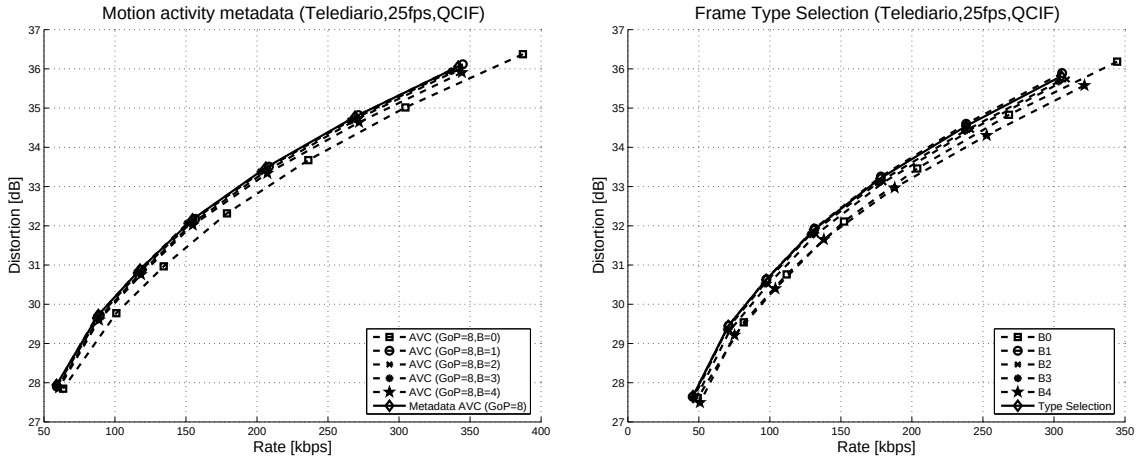
The 5 variations of the fixed GoP structure are compared to the frame type selection scheme (rate-distortion curve with legend *Type Selection*). In this experiment, the *Motion Activity* D is employed to select the best GoP structure for each GoP of the sequence following the mapping of Table 14.2.

The initial analyzed sequences are *Jornal da noite* and *Telediario*. Both sequences cor-



a) GoP size of 8 frames b) GoP size of 16 frames

Figure 14.2: Rate-distortion curves for the frame type selection scheme for the sequence *Jornal da noite*.



a) GoP size of 8 frames b) GoP size of 16 frames

Figure 14.3: Rate-distortion curves for the frame type selection scheme for the sequence *Telediario*.

respond to a subset of a news program of 3000 frames at QCIF resolution. Sequences are composed of a mixture of video shots with different internal motion: from low when the presenter is on the scene to medium or high motion on other scenes, such as a soccer match, etc. Two different GoP sizes are analyzed. GoP size is chosen to be 8 frames in Figs 14.2.a and 14.3.a while the GoP size is 16 frames for Figures 14.2.b and 14.3.b. For both GoP size configurations similar results are obtained. In general, Figures 14.2 and 14.3 show how the frame type selection scheme is able to obtain good results compared to the fixed GoP

structures.

Even if the results of the proposed scheme are, on average, better than the ones of the fixed GoP configurations, the gains obtained by the proposed scheme are not as good as they could have been initially expected. This is due to the fact that the simple mapping of Table 14.2 is not optimum for all GoPs of the sequence and it sometimes selects GoP structures than are not the best choice to encode the current GoP.

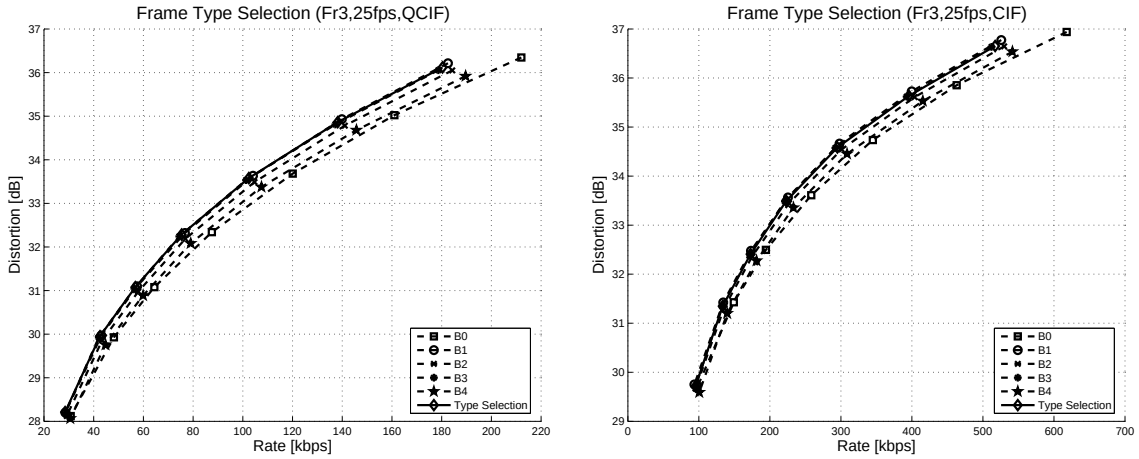
As an example, Table 14.4 lists 5 GoPs from the *Telediario* test sequence. In the example, the first two GoPs (number 28 and 29) correspond to video segments with the presenter so the internal motion is very low. The other 3 GoPs (number $30-32$) correspond to video segments with a soccer match and therefore the internal motion is high. The second column shows the bitrate needed for coding each GoP using fixed GoP structures (using B0, B1, B2 and B3 GoP structures respectively). The third and fourth columns list the *Motion Activity* intensity value for each GoP and the final number of B frames chosen from the mapping of Table 14.2. The last column indicates the final bitrate when using the proposed mapping. The example shows how, for instance, in shot number 28, the motion intensity is low and the mapping uses B3 as the GoP structure. However, the assumption made by the proposed mapping occasionally fails as sometimes using a lower number of B frames between P frames to code high motion sequences is not optimum. For example, in GoP number 31, the proposed mapping uses B1 while for that specific GoP the best alternative is to use B3 to encode the GoP.

GoP Number	**Bitrate [kbps]**	***Motion Activity***	**Mapping (B frames)**	**Final Bitrate [kbps]**
28	45.9, 42.9, 43.2, 41.6	1	3	41.6
29	37.5, 36.8, 37.0, 36.1	1	3	36.1
30	391.0, 325.7, 312.4, 335.2	3	2	312.4
31	476.8, 396.3, 394.3, 380.2	4	1	396.3
32	499.2, 425.8, 435.9, 435.0	4	1	425.8

Table 14.4: Detail of the frame type selection scheme using the *Motion Activity* D for several GoPs of the *Telediario* test sequence. GoP size is 16 frames. Bitrates in the second column are shown using B0, B1, B2 and B3 GoP structures. Quality is approximately 35 dB at all configurations ($Q = 19$).

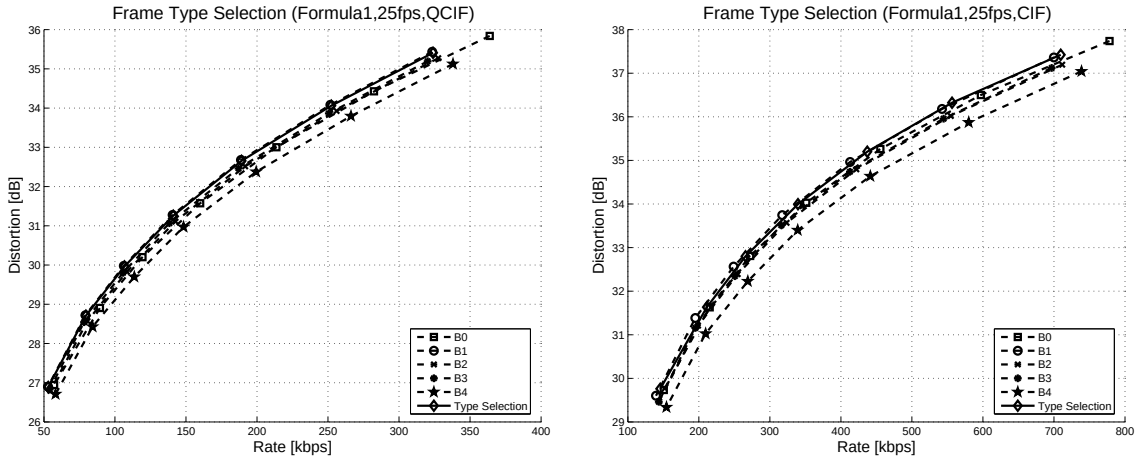
Figure 14.4 studies the frame type selection scheme for the *Fr3* test sequence at QCIF and CIF resolutions and a GoP size of 16 frames. The experiments for this sequence give *B1* and *B3* as the best fixed GoP structures in terms of rate-distortion efficiency. Using the proposed scheme, similar rate-distortion values are obtained at both QCIF and CIF resolutions.

In the case of the *Formula1* test sequence, the best fixed GoP structure to encode the



a) QCIF b) CIF

Figure 14.4: Rate-distortion curves for the frame type selection scheme for the sequence *Fr3* at QCIF and CIF resolution and a GoP size of 16 frames.

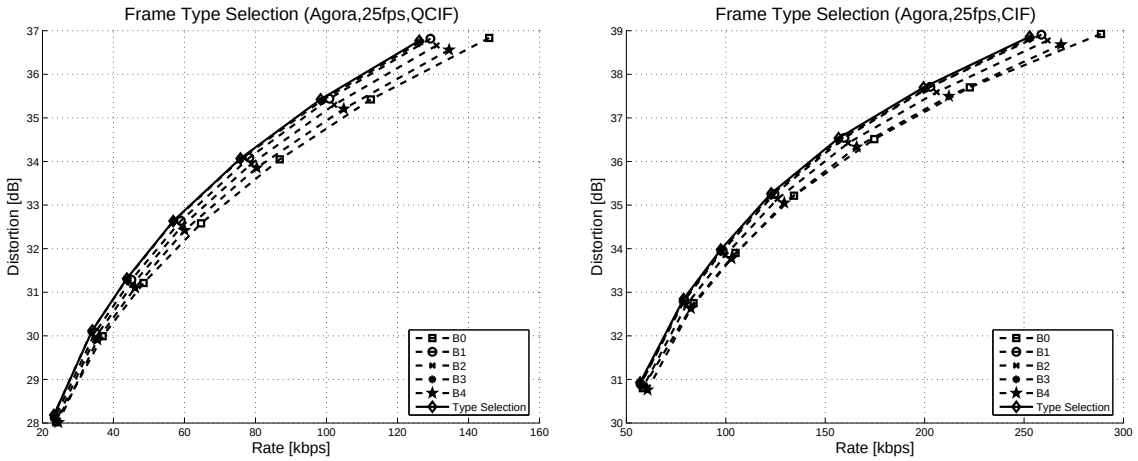


a) QCIF b) CIF

Figure 14.5: Rate-distortion curves for the frame type selection scheme for the sequence *Formula1* at QCIF and CIF resolution and a GoP size of 16 frames.

entire sequence is *B1*. As the internal motion of the sequence is high, the metadata mapping usually selects B1 or B0 GoP structures. The final rate-distortion curve for the proposed scheme in Figure 14.5 is therefore similar to the best fixed GoP structure. However, it can be seen in Figure 14.5.b how the efficiency of the frame type selection decreases at low bitrates. This is due to the fact that, for some sequences, the relation between low motion and the number of B frames between P frames in the GoP structure is not consistent across all bitrates and may change for different quality factors Q. Therefore, if the same selection is used for all

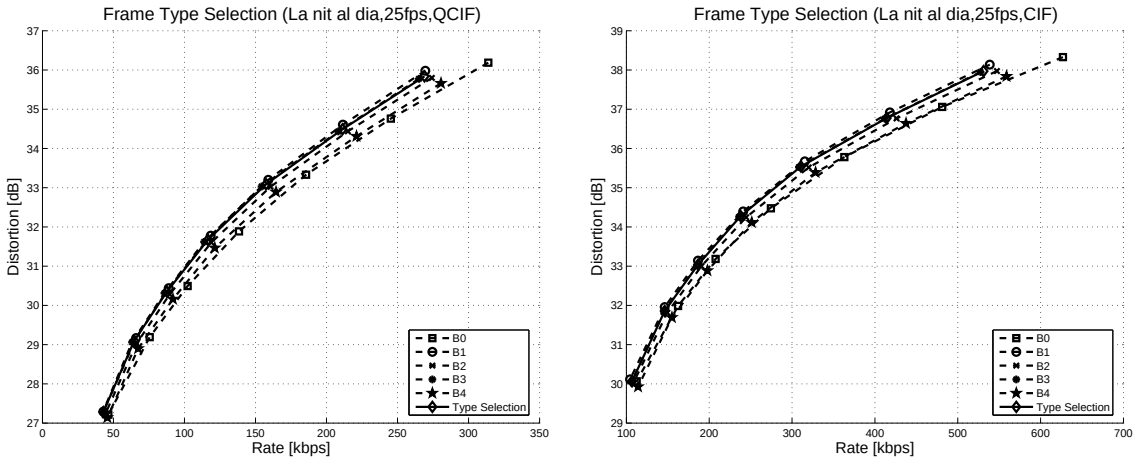
bitrates, the final encoding may decrease its efficiency.



a) QCIF b) CIF

Figure 14.6: Rate-distortion curves for the frame type selection scheme for the sequence *Àgora* at QCIF and CIF resolution and a GoP size of 16 frames.

Figure 14.6 shows the experimental results for the *Àgora* test sequence. This sequence represents the opposite behavior of previous test *Formula1*. In this case, the sequence is composed of low motion shots instead of high motion ones. Therefore, the proposed mapping usually selects a high number of B frames between P frames as the GoP structure. According to the rate-distortion curves of Figure 14.6, this is a good GoP structure to use when coding the test sequence both at QCIF and CIF resolutions.



a) QCIF b) CIF

Figure 14.7: Rate-distortion curves for the frame type selection scheme for the sequence *La nit al dia* at QCIF and CIF resolution and a GoP size of 16 frames.

Finally, results are obtained for the *La nit al dia* test sequence. The sequence is composed of a mixture of low and high internal motion shots. Even though the frame type selection scheme obtains a good rate-distortion efficiency, it is not able to obtain the same efficiency of, for instance, the one obtained by the H.264 codec using a B3 fixed GoP structure for the entire sequence.

Previous experimental results show how, in general, the simple mapping proposed in Table 14.2 obtains a moderate efficiency when selecting GoP structures for all kind of test sequences from low to high internal motion. However, the proposed mapping is not always optimum, and may fail for different sequence or at different quality factors. These problems could be resolved if not only motion information provided by the *Motion Activity* D was to be used in the mapping. Among other things, texture descriptors may also improve the selection of GoP structures. Likewise, studying other GoP structures rather than using only the fixed ones proposed in Table 14.1 may help to better encode specific video segments. In conclusion, the proposed scheme may be a good starting point to select GoP structures when there is no prior knowledge of the original sequence to be coded. Thus, this indexing representation enhanced scheme may reduce the computational complexity when selecting GoP structures in standard hybrid video codecs.

14.5 Indexing Representation not Available at the Decoder

In the frame type selection scheme proposed in the previous sections, the encoder simply optimizes its encoding strategy by inspecting the available indexing representations. The resulting bitstream is still compatible with indexing representations unaware decoders. Therefore, in this scheme, the decoder does not need to access the indexing representation to successfully decode the sequence and, even if not available in the decoder end, the selected indexing representation does not need to be streamed with the content.

However, sending the indexing representation may be useful for other purposes, such as search and retrieval of videos based on motion information. In the case of the *Motion Activity* intensity information, only an integer in the range from 1 to 5 needs to be sent for each GoP. Using 3 bits to store each value and at 25 fps, the streaming only gives 4.7 bps in the case of 16 frames GoP size (or 9.4 for GoP segments of 8 frames). Therefore, even if the indexing representation is streamed with the content, it would not affect at all the final coding bitrate.

Chapter 15

Conclusions and Perspectives

This chapter concludes the dissertation by summarizing the main developments and results of this work and by drawing some final conclusions of the proposed techniques. Possible directions for further research and indications for potential applications are given as well.

15.1 Contribution

The main objective in this thesis has been the study of the synergy between compression and indexing representations. In this work, indexing representations refer to indexing information that has been generated to describe the content with the purpose of searching, querying, browsing or summarizing. The study of the synergy between representations has been divided into two tasks. In the first one, reported in Part II of the dissertation, compression representations have been exploited to create an indexing representation to describe video shots. In the second one, reported in Part III of the thesis, indexing representations have been exploited to improve the coding performance of standard video codecs.

In the first task, a method to create an indexing representation to describe video shots is proposed. The proposed indexing representation is very different from the classical approach of using one or more key-frames to describe a video shot. The representation is content based in the sense that the contents of the video scene are analyzed and separated into background and foreground to create the representation. It is also more suitable to represent video shots with high internal motion or shots of very large duration where finding representative key-frames is more difficult. Specifically, the proposed indexing representation is based on a mosaic to represent the background information and several key-regions to represent foreground objects of the scene.

Mosaics were chosen as candidates to be exploited as they are already employed by some standard codecs, such as MPEG-4 in its mosaic coding scheme, to create compression representations. In this work, the algorithm used to build the mosaic has been improved using

morphological operators to better represent the background information without any penalty on their coding performance. Also, mosaics have been used to extract the representations, called key-regions, of the foreground objects of the shot. Again, this work proposes the use of mathematical morphology to segment the foreground objects of the shot in order to create these key-regions automatically. Moreover, key-regions are created in such a way that non-rigid foreground objects can also be represented by them. In order to create a better representation of the video shot, motion information can also be attached to the indexing representation, therefore, the viewer can have an idea of how the key-regions move through the scene. Results presented in Chapter 8 have shown how the proposed indexing representation can effectively provide functionalities of summarization or browsing of video shots to the final user. In return, one of the main drawbacks of the proposed algorithm to create these indexing representations remains in the need of a two-pass over on the sequence to build the representation, thus, it cannot be used in real-time applications.

In the second task, four different schemes have been developed to prove the ability of existing indexing representations to improve the coding efficiency. In the first proposed scheme, video transitions are encoded using information extracted from a transition descriptor. The proposed scheme is able to reduce the bitrate needed to encode a transition by up to 90% for short transitions and about 65% for longer transitions. As a positive side effect of the bitrate reduction, the scheme can reduce the peaks of bitrate traffic in the presence of transitions. However, as the bitrate gains are concentrated in the transitions, the overall gain in standard sequences is extremely dependent on the number of transitions in the sequence.

The second proposed scheme reformulates part of the motion estimation step of current hybrid codecs into a search and retrieval problem. The scheme improves the long term selection of reference frames by using low level indexing representations such as color descriptors. Using this scheme the possible number of reference frames can be increased considerably with almost no penalty on the computational cost as the indexing representation is selected to be fast and efficient on searches. Experimental results of this scheme have shown bitrate savings of up to 15% for different sequences coded at the same visual quality. The ability to increase the coding efficiency is usually associated with the presence of objects re-appearing in the video scene after several frames or at the beginning of shots so that similar reference frames can be found by the indexing representation.

The third presented scheme, video segment shuffling, uses a high level descriptor to create a new shuffled sequence where video codecs can better exploit its temporal redundancy. Experimental results have shown that video codecs are able to increase their efficiency coding the shuffled sequence instead of the original one with bitrate savings of up to 11% using the same visual quality. This bitrate savings are maximized when using sequences that present scattered, short and similar shots. Sequences composed of television programs such as interviews or video clips are, thus, good candidates to be used in this scheme. Regrettably, for

other kind of sequences the bitrate savings for this scheme are relatively small or nothing. In any case, using the shuffled sequence may be a convenient way of storing the sequence so it matches the structure of the indexing representation and can ease the access through it.

Finally, the fourth proposed scheme consists of a frame type selection based on indexing representations. In this scheme, motion information extracted from the indexing representation is introduced in the video codec so it can select an optimum GoP structure in the encoding process. The experimental results have shown that the proposed scheme yields similar bitrates to the best fixed GoP structure possible for each particular test sequence. From these results, the proposed scheme may be a good starting point to automatically select the best GoP structures to be used when coding a sequence, thus, reducing the computational complexity associated to the selection of GoP structures in standard hybrid video codecs. One of the main advantages of this scheme is that the indexing representation is needed only at the encoder side and the resulting bitstream can still be handled by decoders that are not aware of indexing representations.

15.2 Future research

In this section, some extensions to the work presented in this thesis are given. Following the parts discussed in the dissertation the following research work may be devised:

- Further integration of key-regions in compression representations: The indexing representation of video shots proposed in Part II is based on key-regions to represent foreground objects of the scene. The key-regions are only used in the indexing representation and it is still open to integrate them into the mosaic coding scheme. Future research may focus on exploiting key-regions to improve the coding efficiency in a similar manner the mosaic is used in the mosaic coding scheme. This will allow to further integrate both compression and indexing representations into a more compact representation.

- Study of the activity of key-regions: The key-region representation stores information of the activity followed by a key-region through the video shot. In the case of non-rigid foreground objects, the key-regions can be further analyzed to detect different activity of the represented foreground object. Figure 8.1 in page 100 is a good illustration of how the appearance, contour and texture images contain information of the activity followed by the key-region. The two key-regions of the Figure represent two different persons walking in front of the camera. In the key-region images, higher body parts show no relative motion while lower parts show a considerable amount of relative motion. This kind of behavior may be the key to detect specific activities such as, for instance, people walking, seating, etc [66, 109].

- Combination of compression schemes: Part III of the thesis presented four schemes to

exploit indexing representations in compression ones. The four techniques have been analyzed and studied separately. A further research topic would be to combine them and investigate the global gains achieved by the their combination. Even though it is difficult to envisage, at this point, if the combination will result in an increase of the coding efficiency, the creation of a combined scheme could be a very interesting topic. Additionally, this combined scheme could be used to predict which schemes, among the four proposed or others, may be more appropriate for a specific sequence or for a particular application. For instance, both the long term selection of reference frames and the video segment shuffling schemes exploit the temporal redundancy of video sequences; therefore, it is expected that the combination of both schemes will not achieve greater coding gains. A combined scheme, on the other hand, may be able to predict which scheme is more suitable in a particular situation.

- Improved long term selection of reference frames: In the proposed scheme, a search and retrieval technique using indexing representations has been employed to select good reference frames in the long term prediction buffer. Future research may focus on applying the search and retrieval technique to the entire motion estimation step. The application of this would be when encoding sequences with similar content in the long term, such as mobile phone video conferences, television news, etc. where the encoder has to deal with similar sequences all the time. In these cases, the encoder could employ a learning based coding scenario in which a database is built with representative parts of the sequence that can be searched and retrieved later if similar content needs to be encoded.

- Improved frame type selection: The frame type selection scheme only uses motion information provided by the indexing representation to make the decision of the encoder GoP structure. For that reason, the final solution may not be the optimal and the final coding efficiency may not be as high as expected. It may be useful to study other possibilities such as the use of texture based indexing representations to create more complex decisions.

- Study of other compression representations: All compression representations that have been investigated in this work correspond to standard hybrid codecs, in particular the MPEG-4 and H.264. It would be interesting to translate the ideas proposed in this thesis into different video codecs, for instance, 3D or wavelet codecs. Unfortunately, some of the proposed ideas may be very difficult to be adapted to other video codecs. For example, the use of mosaics or key-regions in 3D codecs may result in a very hard challenge. However, some ideas of other proposed schemes, such as the frame type selection, could be easily adapted to other video codecs.

Appendix A

MPEG-7 Description Definition Language

A.1 Introduction

The DDL provides the language for defining the structure and the content of MPEG-7 documents. It also provides the syntactic rules by which users can combine, extend and refine existing D and DSs. This annex reviews the DDL language focusing on the structural elements and attributes associated with the MPEG-7 indexing representations [60].

A.2 XML Schema Structural Components

The XML Schema consists of three categories of schema components. The primary components are namespaces, element declarations, attribute declarations and type definitions. The secondary components are attribute group definitions, model group definitions, identity-constraint definitions and notation declarations. The third category is composed of annotations, model groups, particles and wildcards.

A.2.1 Namespaces

XML namespaces provide a simple method for assigning universally unique names to element types or attribute names within a XML document so they can be reused in other XML documents. In the context of MPEG-7, the namespace mechanism enables D and DSs from multiple MPEG-7 schemas to be reused and combined to create new schemas.

A.2.2 Element Declarations

An element declaration specifies a type definition for a schema element. The definition can be either explicit or by reference and may provide occurrence and default information. For example, the following element declaration associates the name *Country* with an existing type definition *CountryCode*. It also specifies the default value *en* and that the *Country* element can occur zero or more times.

```
<element name="Country" type="countryCode" default="en" minOccurs="0" maxOccurs="unbounded"/>
```

The accepted values of *minOccurs* are the actual value of the attribute if present or the value 1 otherwise. For *maxOccurs*, accepted values include *unbounded*, the actual value of the attribute if present or value 1 otherwise.

The element declaration can use a reference to an existing element rather than declare a new one. For example, the following XML code declares an existing element *Country* that was declared elsewhere in the schema.

```
<element ref="Country" minOccurs="1"/>
```

A.2.3 Attribute Declarations

Attribute declarations associate an attribute name to an element or a data type. Within the attribute declaration. The *use* attribute specifies the presence (with valid values of *required*, *optional* or *prohibited*). The attribute declaration can have a fixed value or a default. For example, the following code declares the *lang* attribute as optional with the default value of *en-uk*.

```
<attribute name="lang" type="language" use="optional" default="en-uk"/>
```

The valid declaration of an element with the attribute *lang* is:

```
<element name="Annotation">
  <complexType><attribute ref="lang"/></complexType>
</element>
```

A valid instance of an *Annotation* element is:

```
<Annotation lang="en-us"/>
```

A.2.4 Type Definitions

There is a fundamental distinction between type definitions (which create new types) and declarations, which enable the appearance of elements and attributes with specific names and types. For example, the definition of type *contryCode* is:

```
<simpleType name="countryCode">
  <restriction base="string">
    <length value="10"/>
  </restriction>
</simpleType>
```

And the declaration of an element of type *countryCode* is:

```
<element name="Country" type="countryCode" default="en" minOccurs="0" maxOccurs="unbounded"/>
```

XML Schema provides simple, complex and derived definitions.

Simple Type Definitions

Simple types cannot have children elements and cannot carry attributes. XML Schema provides a large number of built-in data types reviewed in Section A.3.

Complex Type Definitions

Complex types allow children elements in their content and may carry attributes. Complex definitions provide constraints on the appearance and nature of the attributes and children elements. The set of rules that describe the content of an element can be:

- *empty*: no child elements, only attributes.
- *mixed*: character data appears between elements and their children.
- *complexContent*: the default content type which consists of elements and attributes. Three compositors are used to structure the included elements: *sequence* where elements appear in the same order in which they are declared, *choice* where only one of the elements may appear in an instance and *all* where elements may appear once or not at all in any order.
- *simpleContent*: used when deriving a *complexType* from a *simpleType* when adding a one or more attributes.

An example of a *complexType* definition together with an element declaration is:

```
<complexType name="PersonNameType">
  <sequence>
    <element name="Title" type="string" minOccurs="0"/>
    <element name="Forename" type="string" minOccurs="0" maxOccurs="unbounded"/>
    <element name="Surname" type="string" minOccurs="1" maxOccurs="1"/>
  </sequence>
  <attribute name="id" type="ID" use="required"/>
</complexType>

<element name="PersonName" type="PersonNameType"/>
```

Derived Type Definitions

It is possible to derive new complex types by *extension* or *restriction* of other simple or complex types. For instance, to extend the previous *PersonNameType* example to define a new *FriendNameType* the following DDL code can be used:

```
<complexType name="FriendNameType">
  <complexContent>
    <extension base="PersonNameType">
      <sequence>
        <element name="Nickname" type="string"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Anonymous Type Definitions

When using a definition in only one element, the XML Schema provides a mechanism to create the definition and declaration at the same time. For example, the following code defines and declares the *Country* element:

```
<element name="Country">
  <simpleType name="countryCode">
    <restriction base="string">
      <length value="10"/>
    </restriction>
  </simpleType>
</element>
```

A.2.5 Group Definitions

The *attributeGroup* and *group* elements provide mechanisms for creating and naming groups of attributes and groups of elements respectively. The compositors *sequence*, *choice* and *all* can be used to construct unnamed groups of elements within complex content. In the example, the *ContactGroup* element is defined as a choice between two elements:

```
<group name="ContactGroup">
  <choice>
    <element ref="Organization"/>
    <element ref="Person"/>
  </choice>
</group>
```

A.3 XML Schema Data Types

The XML Schema data types are provided in order to constrain the possible values of MPEG-7 descriptions. Built-in primitive and derived data types, facets, lists and unions are included in the XML Schema data types.

A.3.1 Primitive Data Types

Among all the built-in primitive data types provided within the XML Schema (refer to [8] for the complete list), the more useful data types include *string*, *boolean*, *float*, *double*, *duration* and *time*.

A.3.2 Derived Data Types

Among all the built-in derived data types provided within the XML Schema (refer to [8] for the complete list), the more useful data types include *integer*, *nonPositiveInteger*, *negativeInteger*, *nonNegativeInteger*, *positiveInteger*, *long*, *int*, *short*, *byte*, *unsignedLong*, *unsignedInt*, *unsignedShort* and *unsignedByte*.

A.3.3 Facets

A derived data type is defined by applying constraining facets to a primitive data type or another derived data type. The following lists of facets are available:

- Bounds facets: *minInclusive*, *minExclusive*, *maxInclusive* and *maxExclusive*.
- Numeric facets: *totalDigits* and *fractionDigits*.
- Pattern facet: *Pattern*.
- Enumeration facet: *Enumeration*.
- Length facets: *Length*, *minLength* and *maxLength*.
- White space facet: *WhiteSpace*.

The example below illustrates the use of the *minInclusive* and *maxInclusive* facets to a *float* data type:

```
<simpleType name="HeightType">
  <restriction base="float">
    <minInclusive value="0.0"/>
    <maxInclusive value="120.0"/>
  </restriction>
</simpleType>
```

The *enumeration* facet can be used to restrict the possible values of every simple type (except the *boolean* type). For example:

```
<simpleType name="TemporalRelationType">
  <restriction base="string">
    <enumeration value="before"/>
    <enumeration value="during"/>
    <enumeration value="after"/>
  </restriction>
</simpleType>
```

A.3.4 List Data Type

List types are composed of sequences of atomic types, separated by white spaces. List types cannot be created from existing list types of from complex types but facets can be applied to derive new list types. For example:

```
<simpleType name="integerVector">
  <list itemType="integer"/>
</simpleType>

<simpleType name="integerVector4">
  <restriction base="integerVector">
    <length value="4"/>
  </restriction>
</simpleType>

<integerVector4>1 2 3 4</integerVector>
```

A.3.5 Union Data Type

Union types enable elements or attributes to be one or more instances of one type obtained from the union of multiple atomic and list types. For example:

```
<element name="IntegerOrDirection">
  <simpleType>
    <union memberTypes="integer directionType"/>
  </simpleType>
</element>

<IntegerOrDirection>1</IntegerOrDirection>
<IntegerOrDirection>left</IntegerOrDirection>
```

A.4 MPEG-7 Specific Extensions

In order to satisfy the MPEG-7 DDL requirements, two specific features have been added to the XML Schema:

- Array and matrix data types allow restricting the size of multidimensional matrices and one-dimensional arrays. Two methods are provided for specifying sizes: a *mpeg7:dimension* facet and a *mpeg7:dim* attribute.
- Built-in derived data types *basicTimePoint* and *basicDuration* specify a time point and a duration of a time period respectively.

Appendix B

Test Sequences

The following Figures show several key-frames at different time instants for all the test sequences employed in the experimental results of the thesis.

$t = 0$ $t = 130$ $t = 260$

$t = 390$ $t = 520$ $t = 650$

Figure B.1: Key-frames of the *ETRI_od_C* test sequence (320×240 resolution at 30 fps).

$t = 0$ $t = 20$ $t = 40$

$t = 60$ $t = 80$ $t = 100$

Figure B.2: Key-frames of the *NHKvideo7* test sequence (320×240 resolution at 30 fps).

$t = 0$ $t = 100$ $t = 200$

$t = 300$ $t = 400$ $t = 500$

Figure B.3: Key-frames of the *ETRI_od_B* test sequence (320×240 resolution at 30 fps).

$t = 0$ $t = 50$ $t = 100$

$t = 150$ $t = 200$ $t = 250$

Figure B.4: Key-frames of the *Stefan* test sequence (QCIF and CIF resolutions at 30 fps).

$t = 0$ $t = 60$ $t = 120$

$t = 180$ $t = 240$ $t = 280$

Figure B.5: Key-frames of the *Coastguard* test sequence (QCIF and CIF resolutions at 30 fps).

$t = 0$ $t = 1$ $t = 2$

$t = 3$ $t = 4$ $t = 5$

Figure B.6: Key-frames of the *Linear short A* test sequence (QCIF and CIF resolutions at 30 fps). The transition starts at frame 1 and ends at frame 4.

$t = 0$ $t = 1$ $t = 2$

$t = 3$ $t = 4$ $t = 6$

Figure B.7: Key-frames of the *Linear short B* test sequence (QCIF, CIF and TV resolutions at 25 fps). The transition starts at frame 1 and ends at frame 5.

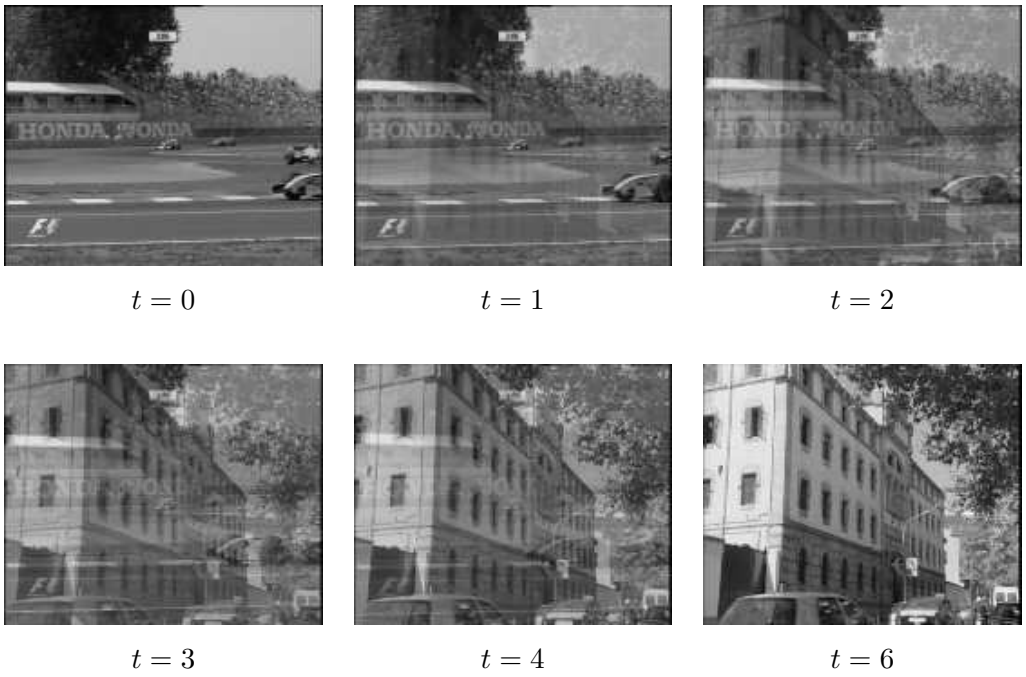


Figure B.8: Key-frames of the *Linear short C* test sequence (QCIF, CIF and TV resolutions at 25 fps). The transition starts at frame 1 and ends at frame 5.

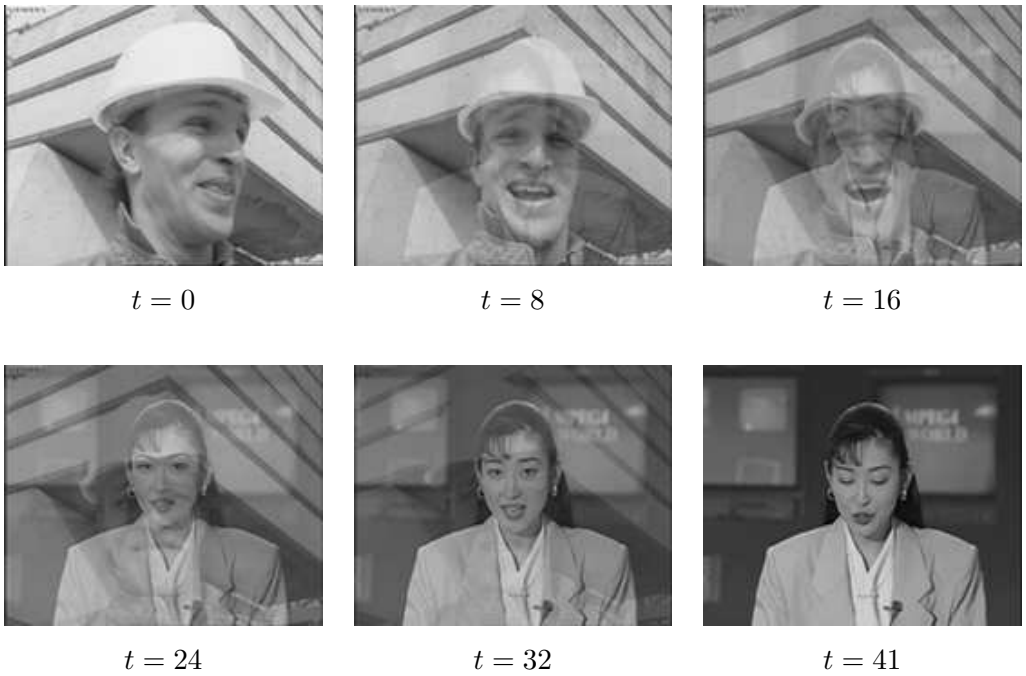


Figure B.9: Key-frames of the *Linear long A* test sequence (QCIF and CIF resolutions at 30 fps). The transition starts at frame 1 and ends at frame 40.

$t = 0$ $t = 6$ $t = 12$

$t = 18$ $t = 24$ $t = 31$

Figure B.10: Key-frames of the *Linear long B* test sequence (QCIF, CIF and TV resolutions at 25 fps). The transition starts at frame 1 and ends at frame 30.

$t = 0$ $t = 6$ $t = 12$

$t = 18$ $t = 24$ $t = 31$

Figure B.11: Key-frames of the *Linear long C* test sequence (QCIF, CIF and TV resolutions at 25 fps). The transition starts at frame 1 and ends at frame 30.

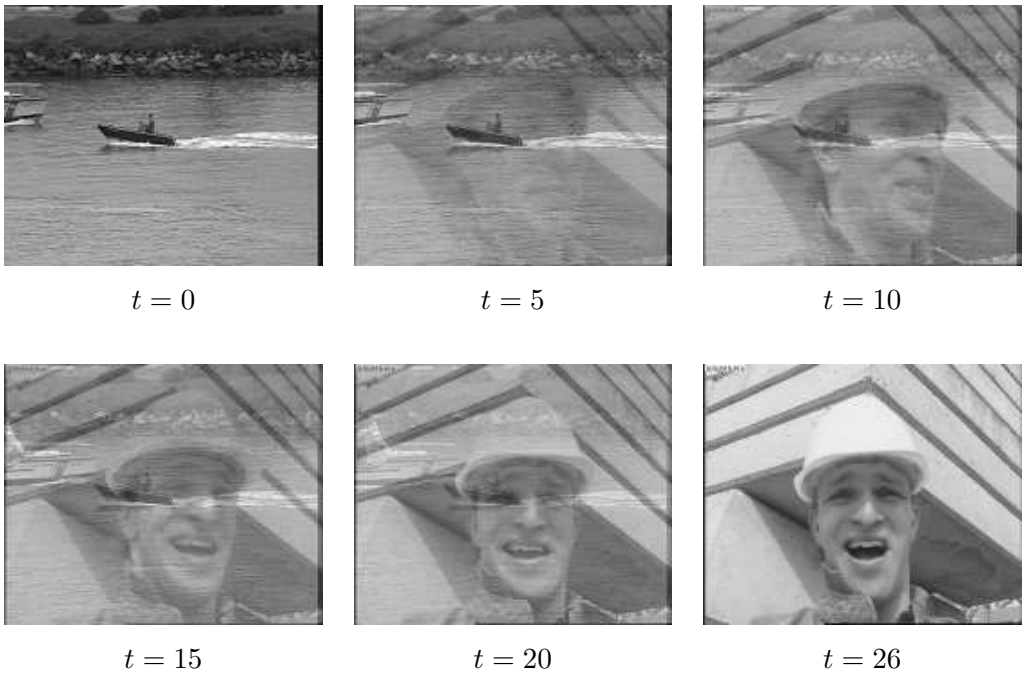


Figure B.12: Key-frames of the *Nonlinear A* test sequence (QCIF, and CIF resolutions at 30 fps). The transition starts at frame 1 and ends at frame 25.

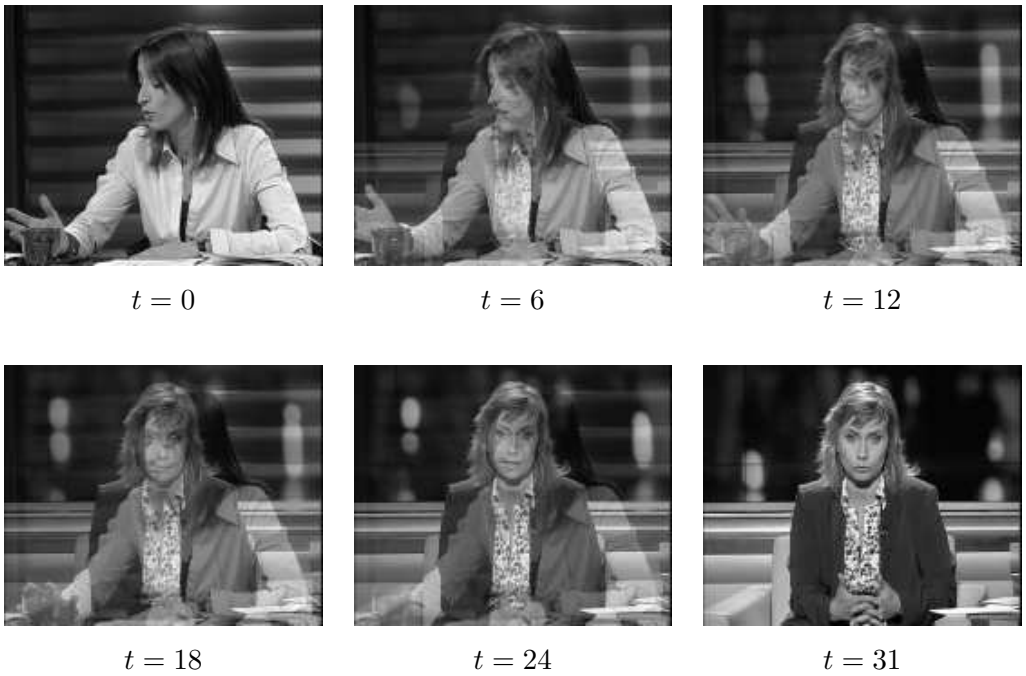


Figure B.13: Key-frames of the *Nonlinear B* test sequence (QCIF, CIF and TV resolutions at 25 fps). The transition starts at frame 1 and ends at frame 30.

$t = 0$ $t = 6$ $t = 11$

$t = 15$ $t = 19$ $t = 22$

Figure B.14: Key-frames of the *Fade A* test sequence (QCIF and CIF resolutions at 30 fps). The fade-out starts at frame 1 and ends at frame 10. The fade-in starts at frame 12 and ends at frame 21.

$t = 0$ $t = 6$ $t = 12$

$t = 16$ $t = 24$ $t = 32$

Figure B.15: Key-frames of the *Fade B* test sequence (QCIF, CIF and TV resolutions at 25 fps). The fade-out starts at frame 1 and ends at frame 15. The fade-in starts at frame 17 and ends at frame 31.

$t = 0$ $t = 73$ $t = 146$

$t = 219$ $t = 292$ $t = 363$

Figure B.16: Key-frames of the *Geri* test sequence (176×92 resolution at 30 fps).

$t = 0$ $t = 600$ $t = 1200$

$t = 1800$ $t = 2400$ $t = 2999$

Figure B.17: Key-frames of the *Telediario* test sequence (QCIF resolution at 25 fps).

$t = 0$ $t = 1200$ $t = 2400$

$t = 3600$ $t = 4800$ $t = 6999$

Figure B.18: Key-frames of the *Jornal da noite* test sequence (QCIF resolution at 25 fps).

$t = 0$ $t = 1000$ $t = 2000$

$t = 3000$ $t = 4000$ $t = 4999$

Figure B.19: Key-frames of the *Fr3* test sequence (QCIF and CIF resolutions at 25 fps).

$t = 0$ $t = 500$ $t = 1000$

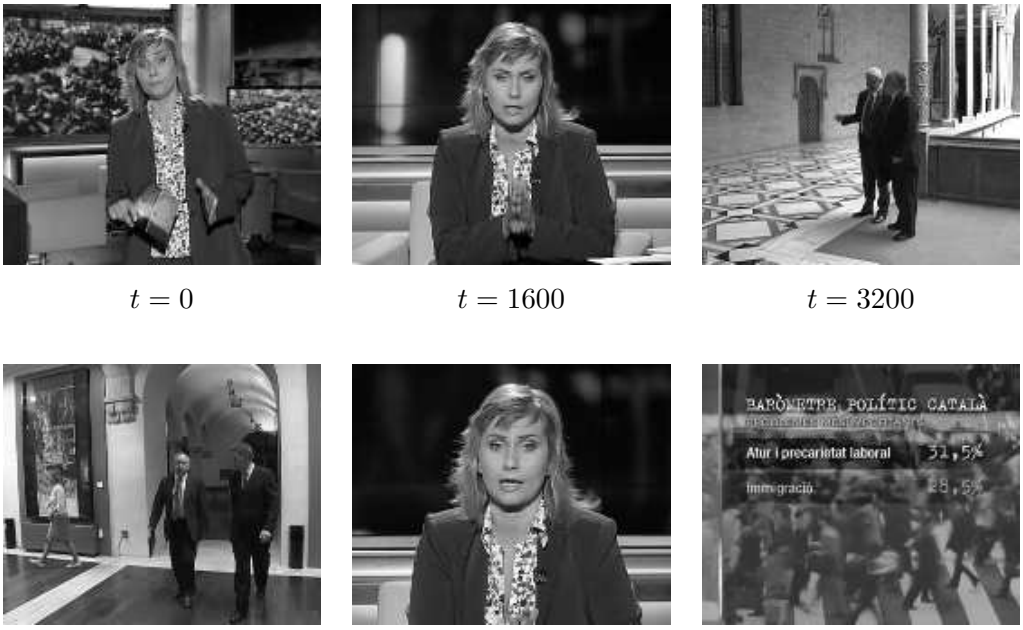
$t = 1500$ $t = 2000$ $t = 2499$

Figure B.20: Key-frames of the *Formula1* test sequence (QCIF and CIF resolutions at 25 fps).

$t = 0$ $t = 2800$ $t = 5600$

$t = 8400$ $t = 11200$ $t = 13999$

Figure B.21: Key-frames of the *Àgora* test sequence (QCIF and CIF resolutions at 25 fps).



$t = 0$ $t = 1600$ $t = 3200$

$t = 4800$ $t = 6400$ $t = 7999$

Figure B.22: Key-frames of the *La nit al dia* test sequence (QCIF and CIF resolutions at 25 fps).

Bibliography

[1] ISO/IEC International Standard 14496-10:2003. Information technology - coding of audio-visual objects - part 10: Advanced video coding, 2003.

[2] D. Adolph and R. Buschmann. 1.15 Mbit/s coding of video signals including global motion compensation. *Signal Processing: Image Communication*, 3(2–3):259–274, June 1990.

[3] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, 23:90–93, January 1974.

[4] P. Aigrain, H. J. Zhang, and D. Petkovic. Content-based representation and retrieval of visual media: a state-of-the-art review. *Multimedia Tools and Applications*, 3(3):179–202, November 1996.

[5] J. S. Boreczky and L. A. Rowe. Comparison on video shot boundary detection techniques. *SPI Storage and Retreival for Image and Video Databases*, 5(2):122–128, April 1996.

[6] J. M. Boyce. Weighted prediction in the H.264/MPEG AVC video coding standard. In *IEEE Proceedings of the International Symposium on Circuits and Systems*, volume 3, pages 789–792, May 2004.

[7] J. Calic, S. Sav, E. Izquierdo, S. Marlow, N. Murphy, and N. E. O'Connor. Temporal video segmentation for real-time key frame extraction. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 3632–3635, Orlando, Florida, May 2002.

[8] World Wide Web Consortium. XML Schema Part 0: Primer, Part 1: Structures, Part 2: Datatypes, W3C Proposed Recommendation, March 2001. http://www.w3.org/TR.

[9] Intel Corporation. Intel MPEG-2 IPP. http://www.intel.com.

[10] A. R. Dasu and S. Panchanathan. A wavelet-based sprite codec. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(2):244–255, February 2004.

[11] J. Davis. Mosaics of scenes with moving objects. In *Proceedings of Computer Vision and Pattern Recognition*, pages 354–360, Santa Barbara, USA, June 1998.

[12] Y. Deng and B. S. Manjunath. NeTra-V: Toward an object-based video representation. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):616–627, September 1998.

[13] A. Divakaran, R. Radhakrishnan, and K. A. Peker. Motion activity-based extraction of key-frames from video shots. In *Proceedings of International Conference on Image Processing*, volume 1, pages 932–935, Rochester, USA, September 2002.

[14] A. Divakaran, A. Vetro, H. Nishikawa, and K. Asai. Video browsing system based on compressed domain feature extraction. *IEEE Transactions on Consumer Electronics*, 46(3):637–644, August 2000.

[15] N. D. Doulamis, A. D. Doulamis, Y. Avrithis, and S. D. Kollias. A stochastic framework for optimal key frame extraction from MPEG video databases. In *Proceedings of International Workshop on Multimedia Signal Processing*, pages 141–146, Copenhagen, Denmark, September 1999.

[16] F. Dufaux and J. Konrad. Efficient, robust and fast global motion estimation for video coding. *IEEE Transactions on Image Processing*, 9(3):497–501, March 2000.

[17] A. Fusiello, M. Aprile, R. Marzotto, and V. Murino. Mosaic of a video shot with multiple moving objects. In *Proceedings of International Conference on Image Processing*, volume 2, pages 307–310, Barcelona, Spain, September 2003.

[18] C. Gomila and F. Meyer. Tracking objects by graph matching of image partition sequences. In *Proceedings of the IAPR-TC15, Workshop on Graph-Based Representations*, Ischia, Italy, May 23-25 2001.

[19] W. C. Graustein. *Introduction to higher geometry.* Macmillan, New York, 1930.

[20] P. N. Hashimah and J. Jiang. Towards key frame extraction in compressed domain. In *Proceedings of International Conference on Electronics, Circuits and Systems*, volume 2, pages 615–618, Sharjah, United Arab Emirates, December 2003.

[21] J. R. Hidalgo and P. Salembier. Morphological tools for robust key-region extraction and video shot modeling. In *International Conference on Scale-Space and Morphology in Computer Vision*, volume 1, pages 407–416, Vancouver, Canada, July 2001.

[22] J. R. Hidalgo and P. Salembier. Robust segmentation and representation of foreground key-regions in video sequences. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 3, Salt Lake City, Utah, USA, May 2001.

[23] J. R. Hidalgo and P. Salembier. Metadata based coding tools for hybrid video codecs. In *Proceedings of Picture Coding Symposium*, pages 473–477, St. Malo, France, April 23-25 2003.

[24] J. R. Hidalgo and P. Salembier. On the use of indexing metadata to improve the efficiency of video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):410–419, March 2006.

[25] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the Institute of Radio Engineers*, volume 40, pages 1098–1102, September 1952.

[26] F. Idris and S. Panchanathan. Review of image and video indexing techniques. *Journal of Visual Computing and Image Representation*, 8(2):146–166, June 1997.

[27] M. Irani and P. Anandan. Video indexing based on mosaic representations. *Proceedings of IEEE*, 86(5):905–921, May 1998.

[28] M. Irani, S. Hsu, and P. Anandan. Video compression using mosaic representations. *Signal Processing*, 7:529–552, 1995.

[29] M. Irani and H. Stephen. Mosaic-based video compression. In *Proceedings of SPIE, Digital Video Compression: Algorithms and Technologies*, volume 2419, pages 242–253, April 1996.

[30] ISO/IEC DIS 13818 (MPEG-2). Information Technology - Generic Coding of Moving Pictures and Associated Audio. ITU-T Recommendation H.262, March 1994.

[31] ISO/IEC JTC1/SC29/WG1. JPEG2000 verification model 7.0, N1684, 2000.

[32] ISO/IEC/JTC1/SC29/WG11. MPEG-7 overview (version 10). N6828, 2004.

[33] ITU-T. Video codec for audiovisual services at px64 kbits: Recommendation H.261, 1993.

[34] ITU-T. Codecs for videoconferencing using primary digital group transmission: Recommendation H.120. Revision 1, 1994.

[35] ITU-T. Video coding for low bitrate communication: Recommendation H.263. Version 2, 1998.

[36] J. R. Jain and A. K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, 29(12):1799–1808, December 1981.

[37] A. H. Jin, C. S. Kim, H. K. Kang, and Y. M. Ro. Generation of the MPEG-7 descriptors in compressed domain. In *Proceedings of SPIE*, volume 4667, pages 160–169, San Jose, California, USA, January 2002.

[38] K. Jinzenji, H. Watanabe, S. Okada, and N. Kobayashi. MPEG-4 very low bit-rate video compression using sprite coding. In *Proceedings of International Conference on Multimedia and Expo*, pages 4–7, August 2001.

[39] H. Jozawa, K. Kamikura, A. Sagata, H. Kotera, and H. Watanabe. Two-stage motion compensation using adaptative global MC and local affine MC. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):75–85, February 1997.

[40] ISO/IEC JTC1/SC29/WG11/N2805. MPEG-4 reference software FDIS. 48th MPEG meeting, July 1999.

[41] ISO/IEC JTC1/SC29/WG11/N5231. MPEG-21 overview v.5, October 2002.

[42] version JM 6.1e JVT Reference Software, 2003.

[43] version JM 7.0 JVT Reference Software, 2003.

[44] E. K. Kang, S. J. Kim, and J. S. Choi. Video retrieval based on key frame extraction in compressed domain. In *Proceedings of International Conference on Image Processing*, volume 3, pages 260–264, Kobe, Japan, October 1999.

[45] S. Kappagantula and K. Rao. Motion compensated interframe image prediction. *IEEE Transactions on Communications*, 33(9):1011–1015, September 1985.

[46] E. Kasutani and A. Yamada. The MPEG-7 color layout descriptor: A compact image feature description for high-speed image/video retrieval. In *Proceedings of International Conference on Image Processing*, Thessaloniki, Greece, October 2001.

[47] D. E. Knuth. *The art of computer programming: fundamentals algorithms*, volume 1. Addison Wesley Longman, 3 edition, May 1997.

[48] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. In *Proceedings of National Telecommunications Conference*, pages G5.3.1–G5.3.5, New Orleans, December 1981.

[49] J. Konrad and F. Dufaux. Improved global motion estimation for N3. ISO/IEC JTC1/SC29/WG11 M3096, February 1998.

[50] G. G. Langdon and J. Rissanen. A double-adaptive file compression algorithm. *IEEE Transactions on Communications*, 31(11):1253–1255, November 1983.

[51] J. Lee and B. W. Dickinson. Rate-distortion optimized frame type selection for MPEG encoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(3):501–510, June 1997.

[52] M. Lee, W. Chen, C. B. Lin, C. Gu, T. Markov, S. I. Zabinsky, and R. Szeliski. A layered video object coding system using sprite and affine motion model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):130–145, February 1997.

[53] D. LeGall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34:47–58, April 1991.

[54] M.D. Levine. *Vision in Man and Machine*. McGraw-Hill, New York, USA, 1985.

[55] J. Llach and P. Salembier. Analysis of video sequences: Table of contents and index creation. In *Proceedings of International Workshop on Very Low Bit-rate Video*, pages 29–30, Kyoto, Japan, October 1999.

[56] Y. Lu, W. Gao, and F. Wu. Fast and robust sprite generation for MPEG-4 video coding. In *Proceedings of IEEE Pacific Rim Conference on Multimedia*, pages 118–125, 2001.

[57] Y. Lu, W. Gao, and F. Wu. Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(5):394–405, May 2003.

[58] Y. Lu, W. Gao, and F. Wu. Efficient video coding with fractional resolution sprite prediction technique. *Electronics Letters*, 39(3):279–280, February 2003.

[59] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[60] B. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7*. Wiley, 2002.

[61] Sorenson Media. Sorenson Squeezer. http://www.sorenson.com.

[62] P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim. Robust regression methods for computer vision: A review. *International Journal of Computer Vision*, 6(1):59–70, 1991.

[63] T. Meier and K. N. Ngan. Automatic segmentation of moving objects for video object plane generation. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):525–538, September 1998.

[64] F. Meyer. Graph-based morphological segmentation. In *Proceedings of the IAPR-TC15, Workshop on Graph-Based Representations*, Castle of Haindorf, Austria, May 10-12 1999.

[65] F. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Computing and Image Representation*, 1(1):21–45, 1990.

[66] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, March 2001.

[67] F. Moscheni, F. Dufaux, and M. Kunt. A new two-stage global/local motion estimation based on a background/foreground segmentation. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 2261–2264, Detroit, May 1995.

[68] ISO/IEC/JTC1/SC29/WG11 N3908. MPEG-4 video verification model version 18.0, January 2001.

[69] R. Narasimha, A. Savakis, R. M. Rao, and R. De Queiroz. Key frame extraction using MPEG-7 motion descriptors. In *Proceedings of Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1575–1579, November 2003.

[70] P. Ndjiki-Nya, B. Makai, A. Smolic, H. Schwarz, and T. Wiegand. Improved H.264/AVC coding using texture analysis and synthesis. In *Proceedings of International Conference on Image Processing*, volume 3, pages 849–852, Barcelona, Spain, September 2003.

[71] P. Ndjiki-Nya, B. Makai, A. Smolic, H. Schwarz, and T. Wiegand. Video coding using texture analysis and synthesis. In *Proceedings of Picture Coding Symposium*, St. Malo, France, April 23-25 2003.

[72] IEEE Transactions on Circuits and Systems for Video Technology. Special issue on H.264/AVC, vol. 13, no. 7, 2003.

[73] N. V. Patel and I. K. Sethi. Video shot detection and characterization for video databases. *Pattern Recognition*, 30(4):607–626, April 1997.

[74] E. Peker and A. Divakaran. Automatic measurement of intensity of motion activity of video segments. In *Proceedings of SPIE: Storage and Retrieval from Multimedia Databases*, San Jose, California, October 2001.

[75] F. Pereira and T. Ebrahimi, editors. *The MPEG-4 Book*. Prentice Hall, 2002.

[76] R. Piqué and L. Torres. Efficient face coding in video sequences combining adaptative principal component analysis and a hybrid codec approach. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, Hong Kong, China, 2003.

[77] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1988.

[78] SMPTE Metadata Dictionary RP210.4. http://www.smpte-ra.org/mdd.

[79] P. Salembier, J.R. Casas, P. Correia, L. Garrido, N. O'Connor, F. Pereira, and J. R. Hidalgo. MPEG-7 description scheme proposal P185, October 2000.

[80] P. Salembier, J.R. Casas, P. Correia, L. Garrido, N. O'Connor, F. Pereira, and J. R. Hidalgo. MPEG-7 description scheme proposal P186, October 2000.

[81] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for filtering, segmentation and information retrieval. In *Proceedings of International Conference on Image Processing*, pages 4–7, Chicago (IL), USA, October 1998.

[82] P. Salembier, J. Llach, and L. Garrido. Visual segment tree creation for MPEG-7 description schemes. *Pattern Recognition*, 35(3):563–579, March 2002.

[83] P. Salembier and F. Marqués. Region-based representations of image and video: segmentation tools for multimedia services. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1147–1169, December 1999.

[84] P. Salembier, A. Oliveras, and L. Garrido. Antiextensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7(4):555–570, April 1998.

[85] P. Salembier and J. Serra. Flat zones filtering, connected operators and filters by reconstruction. *IEEE Transactions on Image Processing*, 3(8):1153–1160, August 1995.

[86] H. S. Sawhney and S. Ayer. Compact representation of videos through dominant multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830, 1996.

[87] H. Schwarz, D. Marpe, and T. Wiegand. Analysis of hierarchical B pictures and MCTF. In *Proceedings of International Conference on Multimedia and Expo*, Toronto, Canada, July 2006.

[88] H. Shen, M. S. Kankanhalli, S. H. Srinivasan, and W. Yan. Mosaic based view enlargement for moving objects in moving pictures. In *Proceedings of International Conference on Multimedia and Expo*, volume 2, pages 807–810, Taipei, Taiwan, June 2004.

[89] A. Smolic, T. Sikora, and J. Ohm. Long-term global motion estimation and its application for sprite coding, content description and segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1227–1242, December 1999.

[90] A. Smolic, Y. Vatis, H. Schwarz, and T. Wiegand. Improved H.264/AVC coding using long-term global motion compensation. In *Proceedings of Visual Communication and Image Processing*, San Jose, USA, January 2004.

[91] A. Smolic, Y. Vatis, and T. Wiegand. Long-term global motion compensation applying super-resolution mosaics. In *IEEE Proceedings of the International Symposium on Consumer Electronics*, Erfurt, Germany, September 24-26 2002.

[92] X. Sun, A. Divakaran, and B. S. Manjunath. A motion activity descriptor and its extraction in compressed domain. In *Proceedings of IEEE Pacific Rim Conference on Multimedia*, pages 450–457, 2001.

[93] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[94] Pinnacle Systems. Pinaccle MPEG-2 encoder. http://www.pinnaclesys.com.

[95] R. Szeliski and H. Y. Shum. Creating full view panoramic image mosaics and texture-mapped models. In *ACM siggraph, Computer Graphics*, pages 251–258, August 1997.

[96] B. Tannenbaum, R. Suryadevara, and S. Hsu. Evaluation of a mosaic based approach to video compression. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 1213–1215, Atlanta, USA, May 1996.

[97] B. Tao, B. W. Dickinson, and H. A. Peterson. Adaptive model-driven bit allocation for MPEG video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):147–157, February 2000.

[98] D. Taubman and M. Marcellin. *JPEG2000*. Kluwer Academic Plublisher, 2002.

[99] C. Toklu, A. T. Erdem, and A. M. Tekalp. Two-dimensional mesh-based mosaic representation for manipulation of video objects with occlusion. *IEEE Transactions on Image Processing*, 9(9):1617–1630, September 2000.

[100] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:583–598, 1991.

[101] A. J. Viterbi and J. K. Omura. *Principles of Digital Communications and Coding*. Mc Graw-Hill, New York, 1979.

[102] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, September 1994.

[103] T. Wiegand and B. Girod. *Multi-frame motion-compensated prediction for video transmission*. Kluwer Academic Plublisher, 2001.

[104] T. Wiegand, X. Zhang, and B. Girod. Long-term memory motion-compensated prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 9:70–84, 1999.

[105] S. F. Wu and J. Kittler. A differential method for simultaneously estimation of rotation, change of scale and translation. *Signal Processing: Image Communication*, 2(1):69–80, May 1990.

[106] M. Yeung and B. Liu. Efficient matching and clustering of video shots. In *Proceedings of International Conference on Image Processing*, volume 1, pages 338–341, October 1995.

[107] M. Yeung and B. L. Yeo. Time-constrained clustering for segmentation of video into story units. In *Proceedings of International Conference on Pattern Recognition*, pages 375–380, 1996.

[108] H. Yi, D. Rajan, and L. T. Chia. Global motion compensated key frame extraction from compressed videos. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 453–456, Philadelphia, USA, March 2005.

[109] A. Yilma and M. Shah. Recognizing human actions in videos acquired by uncalibrated moving cameras. In *Proceedings of International Conference on Computer Vision*, volume 1, pages 150–157, 2005.

[110] H. J. Zhang, J. Wu, D. Zhong, and S. W. Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643–658, April 1997.

[111] D. Zhong and S. F. Chang. Amos: An active system for MPEG-4 video object segmentation. In *Proceedings of International Conference on Image Processing*, volume TP5, pages 0–4, Chicago, USA, October 1998.

[112] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, May 1977.