N° d'ordre : 4012

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX 1

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Christian Käs**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Analyse et indexation des flux vidéos dans le domaine compressé H.264

♢

Compressed domain indexing and analysis of H.264 streams

Soutenue le : 22 Mars 2010

Après avis des rapporteurs :

Christine Guillemot	Directrice de Recherche
Touradj Ebrahimi ...	Professeur

Devant la commission d'examen composée de :

Christine Guillemot	Directrice de Recherche .	Rapporteur
Touradj Ebrahimi ...	Professeur	Rapporteur
Philippe Joly	Professeur	Examinateur
Christophe Papin ...	Responsable Scientifique	Examinateur
Jenny Benois-Pineau	Professeur	Présidente
Henri Nicolas	Professeur	Directeur de Thèse

– 2010 –

Résumé

Le volume de données audiovisuel disponible continue à augmenter rapidement avec le développement d'applications telles que, par exemple, les réseaux en ligne du type Youtube, des systèmes fermés de vidéo-protection ou encore des archives professionnelles. Les principaux challenges technologiques concernant ce type d'applications concernent le stockage, la distribution ainsi que l'analyse et l'indexation automatique de ces données. De plus, l'accès à ces contenus doit être supportés par des plateformes diverses, allant de stations de travail à des dispositifs mobiles. Un prérequis important pour un stockage et une distribution efficace est l'utilisation de schémas de codage source performants. Le dernier standard en date de la famille MPEG, le codeur H.264 et son extension scalable AVC remplis les prérequis nécessaires et a de ce fait été largement adopté dans une large variété d'applications commerciales ou non. Dans le cadre de cette thèse, nous nous sommes concentrés sur la problématique de l'analyse et de l'indexation automatique de vidéos, ce qui permet de jeter les bases pour permettre un accès efficace aux données. En partant du constat que les vidéos sont, dans la plupart des cas, stockées et transmises sous une forme encodées, nous avons abordé la problématique d'une analyse dans le domaine compressé, ce qui représente un compromis entre d'une part la complexité opératoire, et d'autre part, la qualité des résultats. Les informations les plus pertinentes dans les flux compressés H264 sont les vecteurs de mouvement associées à la structure en macroblocs dans les frames prédites temporellement (type P et B) Nous présentons ici différentes techniques non supervisées permettant de déterminer le mouvement de la caméra, et de détecter et suivre les objets en mouvement à partir des vecteurs de mouvement. Puis nous présentons l'efficacité de l'analyse dans le domaine compressé pour plusieurs contextes applicatifs tels que l'analyse du trafic routier, ou la détection des copies. Les limitations provenant du fait que nous utilisons des informations de mouvement le plus souvent bruités dans le domaine compressé sont évaluées.

Les principales contributions de ce travail sont les suivantes: 1) une évaluation en profondeur des possibilités et limitations de l'analyse dans le domaine compressé en utilisant les dernières versions du codeur H264, et 2) des schémas robustes d'analyse combinant les avantages des méthodes classiques d'analyse dans le domaine image avec les avantages des approches dans le domaine compressé.

Abstract

The amount of generated audiovisual content continues to increase. Applications that heavily rely on video include web-based networks such as Youtube and closed systems such as surveillance networks and professional video archives. The main technological challenges concerning these types of applications are efficient storage, distribution and automated video analysis and indexing. Furthermore, content access should be supported by many platforms, ranging from centralized workstations to mobile devices. A prerequisite for efficient storage and distribution is the employment of powerful source coding frameworks. The latest standard of the MPEG-family – H.264/AVC and its scalable extension SVC – meets the necessary requirements and was therefore adopted as the codec of choice in a variety of commercial and non-commercial applications.

In this work, we concentrate on the challenge of automatic video analysis and indexing, which builds the basis for efficient information access and retrieval. Taking advantage of the fact that video in most cases is stored and transmitted in encoded form, we pursue the approach of compressed domain processing, which represents a reasonable trade-off between computational complexity and quality of analysis results. The most valuable information encoded in H.264 streams is the motion vectors (MVs) that are associated with macroblocks in temporally predicted frames of type B and P. We present a number of completely unsupervised techniques to determine the camera motion and to detect and track moving objects from the extracted MV fields. We furthermore present the potential of compressed domain processing through several example applications, including traffic surveillance and video copy detection. The limitations that arise from the noisy and sparse information in the H.264 domain are evaluated, leading to processing schemes that combine both the computational benefits of compressed domain processing and the rich information found in the image domain. The main contributions of this work are: i) an in-depth evaluation of the possibilities and limitations of compressed domain analysis with respect to state-of-the-art H.264 coded video, and ii) new robust analysis schemes that combine the advantages of classic image domain methods with the benefits of compressed domain approaches.

Acknowledgments

First of all I would like to thank my PhD supervisor Prof. Henri Nicolas for giving me the opportunity to explore the fascinating and challenging world of video indexing and analysis. Without his guidance and insight, this thesis would never have happened. I would also like to thank all other members of the LaBRI, especially the research group Analyse et Indexation Vidéo (AIV) under direction of Prof. Jenny Benois-Pineau for the helpful suggestions and the numerous discussions at work and during the daily lunch break: Claire, Yifan, Boris, Daniel, Hugo, Mathieu, Rémi, Ronan and Svebor. Thanks to all of you.

During my thesis I had the chance to participate in the interesting research project ICOS-HD. My thanks go to all other project members at the institutes I3S in Sophia-Antipolis and IRISA in Rennes for the fruitful discussions during the project meetings.

I would like to thank Prof. Christine Guillemot at IRISA in Rennes and Prof. Touradj Ebrahimi at EPFL in Lausanne for accepting to review this thesis. Their constructive comments and suggestions have been very helpful during the revision of this manuscript. In the same way I would like to thank the remaining members of the jury – Prof. Philippe Joly and Christophe Papin – for the interesting discussion following the defense of this thesis.

I would also like to express my gratitude to Prof. Martin Haardt and to all my former colleagues Ulrike, Florian, Giovanni, João Paolo, Nuan, Mike, Marko, Martin and Vejlko at the Communications Research Laboratory (CRL) of the Ilmenau University of Technology, where I carried out my first steps in the research world.

Last but not least, I want to express my sincere thanks to my family and friends. I'm deeply grateful for the love of my parents and sisters, who always supported and encouraged me in my decisions. Danke für Alles.

Publications

Journals

- C. Käs and H. Nicolas, *Compressed domain indexing of scalable H.264/SVC streams*, In Signal Processing: Image Communication: Special Issue on Scalable Coded Media Beyond Compression, Vol. 26, Issue 6, p. 484-498, July 2009

International Conferences and Workshops

- C. Käs and H. Nicolas, *Rough compressed domain camera pose estimation through object motion*, In Proc. of Int. Conference on Image Processing (ICIP'09), Cairo, 2009
- C. Käs, M. Brulin, H. Nicolas and C. Maillet, *Compressed domain aided analysis of traffic surveillance videos*, In Proc. of 3rd ACM/IEEE Int. Conference on Distributed Smart Cameras (ICDSC'09), Como, Italy, 2009
- C. Käs and H. Nicolas, *Compressed domain copy detection of scalable SVC videos*, In 7th Int. Workshop on Content-Based Multimedia Indexing (CBMI'09), Chania, Crete, 2009
- C. Käs and H. Nicolas, *H.264/SVC Scene Motion Analysis*, In Proc. of Int. Conference on Acoustics, Speech and Signal Processing (ICASSP'09), Taipeh, 2009
- C. Käs and H. Nicolas, *An Approach to Trajectory Estimation of Moving Objects in the H.264 Compressed Domain*, In 3rd Pacific-Rim Symposium on Image and Video Technology (PSIVT'09), Tokyo, 2009
- S. Naci, U. Damnjanovic, B. Mansencal, J. Benois-Pineau, C. Käs, and M. Corvaglia, *The COST292 Experimental Framework for RUSHES Task in TRECVID 2008*, In Proc. of the TRECVID BBC Rushes Summarization Workshop (TVS'08) at ACM Multimedia, Vancouver, 2008
- C. Käs and H. Nicolas, *Joint Global Motion Estimation and Coding for Scalable H.264/SVC High-Definition Video Streams*, In Proc. of 6th Int. Workshop on Content-Based Multimedia Indexing (CBMI'08), London, 2008

Acronyms

1080p	Full-HDTV format. 1920x1080 pixels, progressive scan, 16:9 aspect ratio
2K/nK	Digital film resolution with horizontal resolution of n*1024
720p	HDTV format. 1280x720 pixels, progressive scan, 16:9 aspect ratio
AVC	H.264/Advanced Video Coding, a.k.a. MPEG-4/Part 10
B-frame	Bi-directionally predictive picture (also referred to as B-slice or B-picture)
CBCD	Content-Based Copy Detection
CBVR	Content-Based Video Retrieval
CIF	Common Interchange Format. 352x288 pixels for PAL, 352x240 pixels for NTSC
GME	Global motion estimation
GOP	Group of pictures
HD	High Definition. Also HDTV, High Definition Television
I/IDR-frame	Intra-coded frame | picture | slice / Instantaneous Decoder Refresh
LS	Least squares
MB	Macroblock
MSE	Mean squared error
MV	Motion vector
PCA	Principal component analysis
P-frame	Predictive coded picture (also referred to as P-slice or P-picture)
PSNR	Peak signal-to-noise ratio
SD	Standard Definition
SNR	Signal-to-noise ratio
SVC	Scalable Video Coding. Scalable extension to H.264/AVC
WLS	Weighted least squares

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past few decades, digital video has become omnipresent in both private and professional life. Besides classic broadcasting applications, video cameras are now used in industrial production environments, video surveillance networks, mobile phones, photo cameras and personal computers, to name a few examples. Furthermore, visual entertainment in the form of television, DVD or Internet-based video portals like YouTube is more popular than ever. With the growing amount of available content, efficient automated analysis techniques to understand and organize the generated information are needed. Numerous related fields of research such as computer vision, image processing or video indexing and analysis focus on such problems, which underlines the difficulty of the task.

The traditional approach for computing information about the content of video sequences starts with the smallest entities of visual information: the individual pixels. While frame-wise, pixel-level brightness and color information is extremely rich, the methods are computationally expensive and consume a lot of memory. This problem becomes worse when analyzing *High Definition* (HD) content, where resolutions of up to 1920x1080 pixels are reached. Digital cinema employs resolutions of up to 4K, and the next generation standard *Ultra High Definition* (UHD) or *Super Hi-Vision* (SHV) currently in development have resolutions of up to 8K.

Different approaches exist in order to handle this flood of information. The most common is to down-scale the individual input images prior to processing. Since downscaling reduces the amount of information to be processed, the images initially have to be read in full resolution, and proper down-scaling is both a computationally expensive and memory-demanding operation.

In this work, we pursue another approach to decrease the computational overhead: *Compressed domain analysis*. In contrast to image domain analysis, this approach does not require full stream decoding back to pixel level. The coded image representation itself is used in order to compute high-level information about the actual video content. This approach is

motivated by the fact that videos are most often distributed and stored in compressed form. Therefore, decoding back to pixel level can be avoided. With MPEG-type videos in mind, a second advantage is the presence of motion information, which is otherwise expensive to compute. Block-wise displacement vectors and transform coding coefficients, if they are available, are the main source of information in MPEG coded streams.

Compressed domain indexing and analysis is an interesting and challenging research topic, where the goal is to save computing time by reusing information already present in the stream. Possible applications of such algorithms include video surveillance networks, video retrieval and fast scene analysis. Although image and video analysis tasks working in the pixel domain are usually more precise and reliable, they are often too computationally expensive to employ in real-world applications. The decrease in robustness of compressed domain approaches usually stems from the lack of color information, from block-based coding and from the sparse and noisy nature of the motion vectors (MVs) that are optimized for coding efficiency and not in terms of true optical flow. These motion vectors, which are associated with macroblocks (MBs) in inter-predicted pictures, are nevertheless the most valuable source of information in MPEG streams.

In this thesis, we focus on the analysis of sequences encoded in the state-of-the-art compression format H.264, also know as MPEG-4/Part 10. Due to its superior design and performance, H.264 has replaced former standards like MPEG-2 in a wide range of applications, ranging from the distribution and storage of High-Definition (HD) content to streaming media on mobile devices. Up to now, most applications employ H.264 in its original version, known as *Advanced Video Coding* (AVC). The most recent extension to H.264 is *Scalable Video Coding* (SVC), that enhances the AVC standard by providing spatial, temporal, and quality scalability. Media scalability becomes more relevant in the context of modern applications and heterogeneous distribution networks. End-user devices connected to the same networks range from smart phones to high-definition television sets, raising the need for adaptable solutions. H.264/SVC addresses this issue and provides support for scalable video in a coding-efficient manner.

A goal of this thesis is to provide insight into the possibilities of compressed domain indexing and analysis of H.264 coded video. We present efficient methods for tasks like the estimation of global scene motion and unsupervised detection and tracking of moving objects. The versatility and limitations of compressed domain analysis is demonstrated through several example applications such as traffic surveillance and copy detection in video databases. Results are provided on a variety of videos, ranging from broadcasting content to surveillance sequences.

It has to be pointed out that, except from Chapter 7, the integral part of this work is about compressed domain *analysis*, which represents a special case of compressed domain *processing*. In pure analysis, the content - in our case the bit stream - is not altered.

1.1 Contributions

This thesis contributes in the following way to the challenging field of compressed domain video indexing and analysis.

- The first contribution is the evaluation of previous compressed domain approaches in the context of state-of-the-art AVC or SVC streams. As a result, we adapted a robust, MPEG-2-based algorithm for global motion estimation to the H.264 domain. Based on these results, a simple object detection and tracking technique using only the motion vectors is proposed. The technique is completely unsupervised, allows for multi-object tracking, temporary occlusions and the presence of camera motion.

- As a second contribution, we used compressed domain motion information on single-view sequences to estimate basic geometric scene properties such as the relative distance of moving objects and the orientation angle of the camera. Knowledge about scene geometry is an important step towards a better scene understanding and facilitates tasks like segmentation and object classification.

- The third contribution is the demonstration of the suitability of compressed domain techniques in the context of traffic surveillance and video copy detection. The limitations of pure compressed domain analysis are discussed, leading to schemes that combine the flexibility of traditional image domain approaches with the efficiency of compressed domain analysis.

1.2 Organization

This thesis is organized into three principal parts. Part I introduces the basics and concepts of compressed domain indexing and analysis. Chap. 2 provides a brief introduction to video coding and the coding mechanisms behind H.264/AVC and SVC. We focus on technical details and properties that influence the design and performance of compressed domain analysis methods. Chapter 3 describes a series of unsupervised analysis techniques that solely rely on the motion information present in H.264 streams. A robust global motion estimation algorithm is presented and object detection and tracking algorithms are proposed.

Part II demonstrates how different types of applications can benefit from compressed domain analysis. Chapter 4 presents a framework combining the benefits of image domain techniques with the efficiency of compressed domain methods in the context of traffic surveillance. A novel compressed domain method for camera orientation estimation through object motion from a single view is described in Chap. 5. In Chap. 6, we propose

and evaluate retrieval schemes based on scalable compressed domain features in a video copy detection framework.

We conclude this work with Part III. In Chap. 7, we present an approach to mitigate the negative effects of noisy and corrupted displacement vectors on the quality and robustness of compressed domain analysis methods by correcting the motion field already on encoder side. Finally, conclusions and possible future directions in the challenging field of compressed domain video analysis are provided in Chapter 8.

Part I

Compressed Domain Analysis

Chapter 2

Scalable Video Coding

The market and demand for efficient, scalable video coding solutions is continuously growing. This is mainly due to two major developments: On the *end-user* side, recording and viewing devices like handheld, low-power smartphones are becoming more and more popular while at the same time, an increasing number of users is equipped with high-definition cameras, TV sets and home cinema systems. On the *network* side, on-demand solutions for content distribution are gaining increasing importance in addition to classical broadcasting. Distribution is more and more performed on heterogeneous IP networks and over a large variety of connections, ranging from slow, wireless links to high-speed networks. Scalable source coding schemes have been introduced as a response to these changing conditions. The term *scalability* with respect to video streams comprises three aspects of adaptability:

- **Spatial scalability** refers to the possibility to encode different spatial resolutions in one single video file or stream. The resolution levels have to be pre-defined by the service provider prior to encoding.

- **Temporal scalability** enables the extraction or decoding of the stream at different frame rates.

- **Quality scalability** denotes the capability to vary the visual quality by maintaining the spatial and temporal resolution.

The main motivation behind scalable video coding is to enable a single system to dynamically adapt to different environments. Up to now, this is usually achieved by encoding and storing multiple versions of the same source video (e.g., simulcast), which is inefficient with respect to pre-computation time and storage space, and impractical when deploying streaming applications.

This chapter gives an overview of the most promising, standardized approaches to scalable video coding. No proprietary solutions like *Real*[1] or *Microsoft Windows Media*[2] are covered and focus is put on the two international, scalable coding standards JPEG2000 and H.264/SVC. Before going into technical details, the main motivation behind video coding in general is briefly explained.

2.1 Video Coding Basics

Efficient video coding is a challenging task due to the enormous amount of data that is generated by visual sensors. Fortunately, video signals usually contain a lot of spatial and temporal redundancy that can be exploited during encoding. Furthermore, certain properties of the *Human Visual System* (HVS) allow to omit parts of the video signal prior to encoding without decreasing the perceived quality. In order to better understand some basic prefiltering and compression principles, the most important findings on the limitations of the HVS and their implications on video compression are listed below:

- The HVS has limited resolution and is more sensitive to brightness than color. This allows for low-pass filtering of the input signal and subsampling of the chroma components without loss in the perceived visual quality (e.g., 4:2:0 sampling).
- Human motion perception is limited in frequency, so 25 frames per second are enough to create the impression of continuous motion.
- Errors in high spatial frequencies are less noticeable, so high frequency components can be quantized more coarsely.

The efficiency of todays video codecs is illustrated by a small sample calculation. A raw video stream in the high definition format 1080p (1920x1080 pixels, progressive scan) at 25 fps, with 24 bit color depth and 4:4:4 sampling requires the enormous bit rate of 1244.16 Mbps[3]. By taking the low color sensitivity of the HVS into account, the bit rate can already be cut in half to 622.08 Mbps by performing 4:2:0 subsampling of the chroma components. However, high rates like this are still very impractical or impossible to handle, so almost all audiovisual content is stored and distributed in compressed form. After lossy encoding with MPEG-2 in DVD quality, bit rates of typically 12-20 Mbps are achieved. With the newer and more efficient H.264 codec, bit rates of around 6-12 Mbps can be obtained at similar visual appearance, so the initial data volume is reduced to up to 0.5 % while almost preserving the perceived quality.

[1] `http://www.real.com`
[2] `http://www.microsoft.com/windows/windowsmedia/`
[3] 1920x1080 pixels x 3 channels x 8 bit x 25 fps = 1244.16 Mbps

These tremendous bit rate savings become possible due to spatial and temporal prediction mechanisms, transform coding, quantization and sophisticated entropy coding schemes. Scalable codecs additionally apply prediction between different stream layers. Spatial and temporal prediction is very effective since most scenes contain large visually similar regions that only change slowly over time. Transform coding like *Discrete Cosine Transform* (DCT), *Integer Transform* (IT) or *Discrete Wavelet Transform* (DWT) project the video signal from the spatial in the frequency domain. Like for audio signals, the frequency domain is better suited to compression because the limitations of human perception can be better exploited than in the spatial domain. Since reconstruction errors in high frequency parts are less noticeable, high frequency coefficients can be quantized more coarsely. Quantization is the fundamental reason for quality loss in most compression schemes, because it represents the coding step where information is irreversibly lost. After quantization, entropy coding is performed, which represents a lossless compression technique where fixed size input symbols are translated into variable length output symbols. The length of the output symbol depends on the probability of the input symbol; frequent input symbols are mapped onto short output sequences and vice versa. Examples for well-known entropy coding schemes are Huffman coding [5] and arithmetic coding [6].

Besides compression, another important aspect of video coding is to provide mechanisms to store, access and distribute the encoded data. Furthermore, error concealment strategies should be included to recover and conceal data loss during transmission or because of defective storage devices. A review of such strategies is not within the scope of this work and the interested reader is referred to the respective literature, e.g., [7].

In the following sections, the video coding parts of the two international standards *Motion JPEG2000* and *H.264/SVC* are briefly reviewed. Emphasis is put on H.264 since the remainder of this work concentrates on the analysis of compressed H.264 streams.

2.2 Motion JPEG2000

Motion JPEG-2000 is defined in ISO/IEC 15444-3 and replaces the former Motion JPEG standard. It is based on the JPEG2000 (J2K) format and does not provide temporal prediction mechanisms of any kind, so each video frame is encoded independently without exploiting the temporal redundancy. This enables direct access to each frame and delivers high quality streams, which are the main reasons why it was adopted as the codec of choice by the *Digital Cinema Initiatives*[4] (DCI). It provides both lossy and lossless coding schemes based on the Discrete Wavelet Transform (DWT).

[4]Official DCI website: http://www.dcimovies.com/

Compared to JPEG, J2K offers superior performance with respect to rate-distortion [8] and various other advantages like the mentioned lossless mode, the possibility for progressive download and error correction mechanisms. The use of the DWT is the fundamental difference between J2K and JPEG, which applies a DCT on blocks of 8x8 pixels. The basic coding steps and the DWT transform behind J2K is briefly explained in the following in order to better understand the spatial and quality scalability that is provided by the standard. Since no inter-frame prediction is performed in Motion J2K, temporal scalability is simply achieved by discarding unwanted frames. The five encoding steps that are repeated for each video frame are illustrated in Fig. 2.1 and are briefly explained in the following.

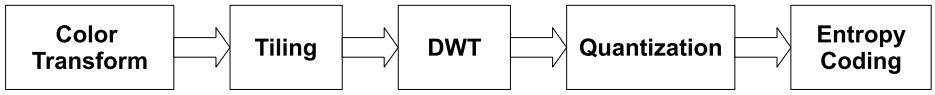


Figure 2.1: Block diagram of frame-level JPEG2000 encoder

Assuming an RGB input image, the first step consists of separating the luminance and the chrominance components. This step is common to most image and video codecs and enables to treat brightness and color information in different ways, which is desirable regarding the properties of the HVS (see Sec. 2.1). The standard provides two possible **color transform** choices, namely the *Irreversible Color Transform* (ICT) and *Reversible Color Transform* (RCT). Since the ICT operates with floating point values, irreversible rounding errors are introduced. The RCT algorithm is implemented in integer arithmetic, so no rounding errors are introduced and the transformation back to RGB happens without information loss. Opposed to DCT based codecs, subsampling of the chroma components (C_rC_b for ICT and UV for RCT) has no to be performed prior to encoding and is handled later in the multi-resolution wavelet domain.

Tiling is the second, optional coding step and may be used to cut the input image in even-sized rectangular regions that are coded separately. Tiling reduces the coding efficiency and can introduce block artifacts, so it is often omitted by using one tile that covers the whole image. The idea behind tiling is to reduce memory consumption during encoding/decoding and to enable partial decoding.

Transform coding in J2K is performed via a 2D **DWT** using Cohen-Daubechies-Feauveau (CDT) wavelets [9]. As for the color transformation, an irreversible and a reversible version of the DWT is part of the standard, using floating point and integer arithmetic, respectively. The original pixel values of each channel are converted from the spatial domain into the wavelet domain and the resulting wavelet coefficients are organized in four equal-sized sub-bands. The total number of coefficients equals the number of input pixels, so no data reduction is introduced by the DWT itself.

The four sub-bands are referred to as LL, HL, LH and HH, where L and H represent the low- and high-frequency parts of the signal, respectively. LL is basically a low-pass filtered, down-scaled version of the input image, HL covers the vertical, LH the horizontal and HH the diagonal high-frequency details. The same decomposition can be recursively applied on the LL sub-band, leading to a multi-level wavelet representation like illustrated in Fig. 2.2, which shows a two-level wavelet decomposition of a sample image. It can be noticed that the DWT captures both frequency and location, i.e., the initial spatial structure of the input image is preserved in all four sub-bands. It also becomes clear that spatial scalability is inherently supported. The lowest LL band represents the smallest possible resolution and subsequent spatial layers can be reconstructed by performing the inverse DWT on each level until the image is fully decoded back to its original size.

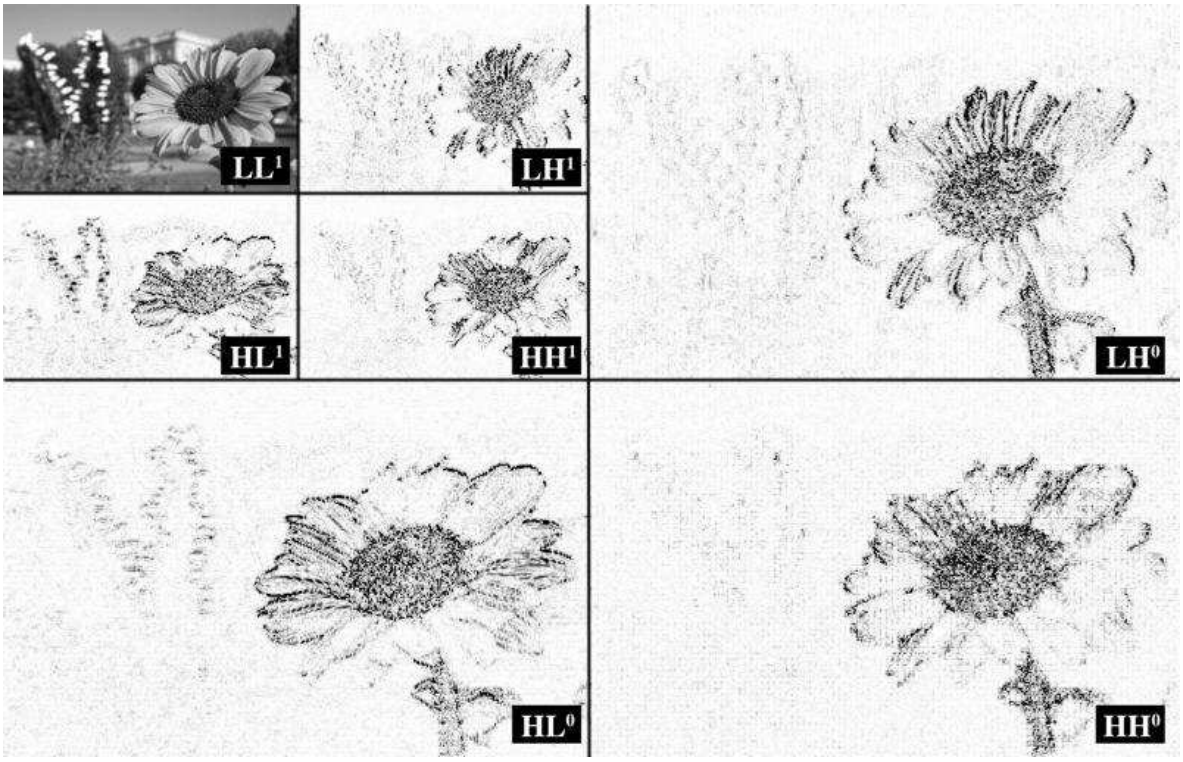


Figure 2.2: Example of DWT sub-bands. Coefficients are shown inverted and amplified for better visibility. Input image ©*Pierre Lasserre*

In lossy mode, the wavelet coefficients are then **quantized** in order reduce the amount of data. The remaining bits are finally coded to produce the J2K bit stream. The **coding** process is designed to provide quality scalability by simply truncating the bit stream at arbitrary positions. Therefore, the bits are re-ordered so that the most important information is coded first and the least important last. The wavelet sub-bands are split into code blocks which are processed by the so-called *Embedded Block Coding with Optimal Truncation* (EBCOT) scheme. Given all code blocks, EBCOT starts by encoding the most significant bit plane and succes-

sively continues with the less significant bits. Additionally, the coefficients within each bit plane are processed with decreasing importance in three coding passes named *Significance Propagation*, *Magnitude Refinement* and *Cleanup pass*. After arithmetic coding, the coded bits are arranged into packets and layers. The resulting bit stream can be arbitrarily truncated to yield lower quality versions.

For more details on the JPEG2000 standard, the interested reader is referred to [10]. Although a number of 2D+t wavelet video coding schemes have been presented in the literature [11, 12], their discussion is not within the scope of this thesis. We will focus on MPEG coded video in the following due to its wide acceptance and popularity regarding a variety of systems and applications, like storage and distribution of content, surveillance and streaming.

2.3 MPEG approaches

End of 2003, the *Moving Picture Experts Group* (MPEG) issued a call for proposals concerning the system design of a scalable video coding standard. Among the fourteen proposals, twelve were based on the wavelet transform and two were extensions to the well-established H.264/AVC codec. The MPEG committee decided in favor of one of the H.264/AVC based approaches, which proved to be superior to the presented wavelet coding schemes with respect to the targeted applications. After the finalization of the so-called *Scalable Video Coding* (SVC) standard, first industry applications appeared, and SVC is expected to make an impact in a number of applications like surveillance and large-scale media distribution.

Since the remainder of this work is centered around the analysis of compressed SVC streams, an introduction to SVC coding is provided in the following. Technical details are presented from the indexing and analysis perspective, i.e., important properties that can be exploited for tasks related to computer vision and indexing are highlighted, while others are omitted. We start with a short history of selected MPEG standards in order to provide a better understanding of the SVC system design.

2.3.1 MPEG-2

The international standard MPEG-2, also known as ISO/IEC 13818 or ITU-T H.262, is the core of most digital television and DVD formats in Standard Definition (SD) to date. It is mainly similar to the former MPEG-1 standard but provides better support for the coding of interlaced video, added more profiles that are tailored to specific application domains and allows for input color sub-sampling schemes other than 4:2:0.

In the following, focus is put on the video part of the standard, ISO/IEC 13818-2. Figure 2.3 illustrates the basic components of a MPEG-2 coded video stream and introduces some important MPEG related terms that will reappear throughout this work.

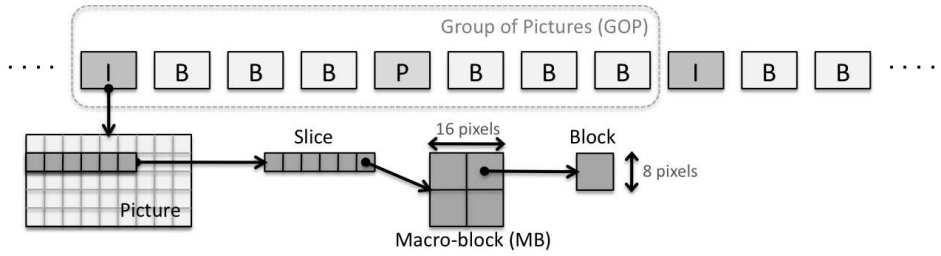


Figure 2.3: Overview of MPEG-2 video stream

An MPEG-2 video stream is composed of different types of coded pictures: *Intra-coded* **I-frames**, *predicted* **P-frames** and *bi-predicted* **B-frames**. They are arranged in a periodically repeated pattern, the so-called *Group-of-Picture* (GOP).

Intra-coding: I-frames

The en- and decoding of **I-frames** does not depend on any other frame and is very similar to JPEG coding of still images. The input picture is transformed into the YC_rC_b color space and is sub-divided into non-overlapping *macroblocks* (MBs), quadratic regions that cover an area of 16x16 pixels of the original input image. Each of these MBs is further sub-divided into four *blocks* of size 8x8 pixels, which are then processed independently by transform coding and quantization. MPEG-2 applies a two-dimensional, irreversible *Discrete Cosine Transform* (DCT) to map from the spatial domain into the frequency domain. The initial 8x8 brightness values result in 8x8 frequency coefficients, where the constant component of the signal, i.e., the average brightness value, is referred to as the DC-coefficient. The high-frequency components are called AC-coefficients. The ensemble of DC-coefficients represents a subsampled version of the input image, where each original region of size 8x8 pixels is replaced by its mean value.

The two-dimensional DCT of size NxM is defined as [13]

$$F(u,v) = \frac{2}{\sqrt{NM}} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2M}, \text{ with}$$
$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u,v = 0. \\ 1 & \text{otherwise.} \end{cases} \tag{2.1}$$

The inverse DCT (IDCT) is defined as

$$f(x,y) = \frac{2}{\sqrt{NM}} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} C(u)C(v)F(u,v) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2M}, \qquad (2.2)$$

where (x,y) are spatial co-ordinates in the image block and (u,v) the co-ordinates in the DCT coefficient block [14].

Like the wavelet decomposition presented earlier, the DCT is performed because compression can be carried out more efficiently in the frequency domain by quantizing the separated frequency components. High-frequency components can be usually quantized more coarsely than low-frequency parts because the human visual system is less sensitive to errors in high-frequency regions. After quantization, many AC coefficients tend to be zero, most notably in the high-frequency range. The coefficients of a block are scanned in zig-zag order to produce long runs of zeros. This allows for more efficient *Run-length encoding* (RLE), which represents the last compression step.

Predictive Coding: P- and B-frames

Natural video scenes show high temporal redundancy, i.e., two successive frames usually are not substantially different and almost share the same background with similar objects, though often slightly displaced due to camera operation and object motion. This property is exploited in MPEG-2 by performing motion-compensated inter-frame prediction. Since natural scenes are composed of multiple independent objects and regions, prediction in MPEG-2 is not performed on the image as a whole but independently on a MB basis.

The first type of predicted frames are called **P-frames**, which take a previous I- or P-frame as reference and which may also be used as a reference for further prediction. For each 16x16 region covered by a P-frame MB, the encoder searches for the most similar region of equal size in the reference picture. This algorithm is called *block matching* and choses the best fit according to the minimum error between the block being coded and the prediction. To determine the block displacement with minimum error, a large number of trial offsets are tested by the coder using the luminance component of the picture. Only translational block motion is considered, so no changes in scale or rotational motion is taken into account. The prediction error contains mostly high-frequency parts and is coded and quantized in the DCT domain.

The optimal MB displacement is stored as a motion vector (MV) in half-pel precision together with the respective MB. Figure 2.4 shows an example of the MB grid and the MVs of a random MPEG-2 video stream. Motion vectors are optimized in terms of coding efficiency, so they do not necessarily reflect the real scene motion.

In cases where block matching leads to poor results in terms of prediction error, the transmission of the MV together with high-valued DCT coefficients would introduce unnecessary overhead. MBs in predicted frames may be coded in intra-mode to overcome this problem. On the other hand, if the prediction error with respect to the reference frame is already below a certain threshold without motion compensation, the MB is coded in SKIP mode, so no information except its mode is transmitted. Static background is the most common reason for skipped MBs.

Figure 2.4: Example of MBs and the associated MVs in a MPEG-2 stream. Original sequence ©*Taurus Media Technik*

Bidirectionally predictive pictures (**B-frames**) can use the previous and next I- or P-frame for motion-compensation and offer the highest degree of compression. Each block in a B-picture can be either intra-coded or forward, backward or bidirectionally predicted. To enable backward prediction from a future frame, the coder reorders the pictures from natural *display order* to *coding order* so that the B-frame is transmitted after the previous and next pictures it references. This introduces a reordering delay dependent on the number of consecutive B-pictures.

While keeping the decoded picture quality constant, coding using different picture types produces a different number of bits. In a typical example sequence, a coded I-frame is about three times larger than a coded P-frame, which is itself approximately 50 % larger than a coded B-picture.

Profiles and Levels

The MPEG-2 standard defines a number of coding tools, algorithms and mechanisms. Depending on the target application, only a subset of all possible features needs to be supported by an encoder for a particular use. A set of different so-called *Profiles* and *Levels* are

introduced in order to limit the complexity and the implementation overhead in real-world applications.

Profiles define sets of allowed tools and coding mechanisms, while more quantitive measures like maximal resolution or bit rate are defined in levels. Not all possible combinations of profiles and levels are allowed.

The two non-scalable profiles are called *Simple Profile* (SP) and *Main Profile* (MP). The SP only includes a rather basic subset of MPEG-2 functionalities and is mostly aimed at low-performance applications like video conferencing and streaming to mobile devices. One of the most drastic limitations of the SP is the exclusion of B-frames, so only I- and P-frames are allowed. Furthermore, no interlaced source material is allowed in the SP, making it unsuitable for television and broadcasting purposes. The MP adds support for B-frames and interlaced video and is by far the most widely used profile. It has been adopted as the coding standard for video DVDs and also in first generation *Digital Video Broadcasting* (DVB) services, replacing analog broadcasting standards like SECAM, PAL or NTSC.

MPEG-2 also defines two scalable modes, the SNR profile (SNRP) and the spatial profile (SP). The SNRP adds support for quality scalability and the SP enables the coding of multiple spatial layers. However, both profiles received very little attention and are barely used. Finally, the *High Profile* (HP) aims at high-performance applications and includes all features from the before mentioned lower profiles, including scalability and 4:2:2 sampling, making in suitable for studio applications.

Limits on the allowed frame size, the frame rate or the maximal bit rate are defined in four so-called *Levels*, namely the *Low Level* (LL), *Main Level* (ML), *High 1440* (H-14) and *High Level* (HL). The most popular combination of profile and level is MP@ML for DVDs and DVB in Standard Definition (SD). High-Definition (HD) applications are supported by MPEG-2 and usually apply MP@HL.

For more details on MPEG-2, the interested reader is referred to one of the numerous sources available, including the contributions from Tudor [14] and Sikora [15]. In the following, an introduction to MPEG-4/Part 2 (MPEG-4/Visual) will be omitted and focus is put on state-of-the-art video coding with MPEG-4/Part 10 and its extensions, better known as H.264.

2.3.2 H.264/Advanced Video Coding (AVC)

H.264/AVC, formally known as ISO/IEC 14496-10 or MPEG-4/Part 10, is the result of the combined efforts of two standards bodies - the *ITU-T Video Coding Experts Group* (VCEG) and the *ISO/IEC Moving Picture Experts Group* (MPEG), who together formed the *Joint Video Team* (JVT). It arose from the need for a more efficient video coding framework that is applicable

over a wide range of applications and underlying distribution systems. At similar decoded picture quality, H.264 achieves about 50 % savings in bit rate compared to MPEG-2.

The first version of AVC was finalized in 2003. By now, multiple extensions and more profiles have been added to the original standard. For example, a set of extensions that target professional production environments, known as the *Fidelity Range Extensions* (FRExt), have been added to H.264 in 2004 [16] and the scalable extension SVC was finalized end of 2007 (see Sec. 2.3.3).

Due to its superior performance in a wide range of applications, H.264 has already been adopted in many major solutions and systems like Adobe Flash Video[5], Apple Quicktime[6], the second generation of DVB in Europe and Japan, in the ITU-T video conferencing specification H.241 and also for video coding in mobile phones. Furthermore, H.264 has been ratified as mandatory in both the HD-DVD and Blu-ray specifications for high definition content.

A brief introduction to H.264/AVC is provided by pointing out some major improvements over MPEG-2 (see Sec. 2.3.1) with main focus on technical details of the actual video coding part which play an important role in the analysis of compressed H.264 streams.

Intra-Coding: I-Slices

Series of intra-coded MBs in H.264 are referred to as I-slices. Since most of the time, an I-slice covers an entire frame, we use the terms slice and frame interchangeably throughout this work. Like in MPEG-2, the coding of MBs in I-frames does not depend on any other frame. Some important improvements over the I-frame concept in MPEG-2 are listed below.

- Intra-coding in MPEG-2 is always performed on fixed size blocks that cover 8x8 pixels of the input image. For each luma block in H.264, the encoder adaptively choses between one of the newly introduced intra-coding modes called Intra_4x4, Intra_16x16 and I_PCM. In order to improve efficiency, most regions are coded in Intra_4x4 mode, whereas low-textured areas are coded as Intra_16x16. The I_PCM mode bypasses all prediction and transform coding processes and transmits pixels values uncoded as is, which is advantageous for lossless coding or in case of anomalous content like regions of white noise.

- For transform coding of pixel values and prediction residuals, the formerly used DCT is replaced by a separable integer transform (IT) of size 4x4 or 8x8. Since the inverse transform is defined by exact integer operations, inverse-transform mismatches are

[5] `http://www.adobe.com/devnet/video/`
[6] `http://www.apple.com/quicktime/`

avoided. In order to transform regions of size 16x16 (Intra_16x16 mode), a 4x4 transform is applied and the 16 resulting DC coefficients undergo a second 4x4 transform coding run. The transform matrix H is given as

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}. \tag{2.3}$$

- H.264 employs intra-prediction in the spatial domain, i.e., prior to transform coding, each 4x4 block is predicted from spatially neighboring samples on the left and above the current block. This is illustrated in Fig. 2.5, together with the eight possible prediction directions. Pure vertical prediction (Mode 0) would result in predicting the block $a - p$ from the upper samples $A - D$. Four of the nine possible prediction modes are illustrated in Fig. 2.6.

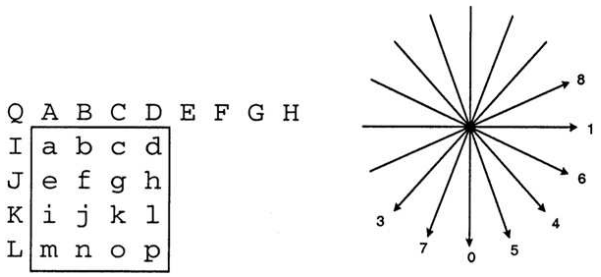


Figure 2.5: *left*: Samples used for Intra_4x4 prediction of AVC. *right*: Eight of nine possible prediction modes [1]

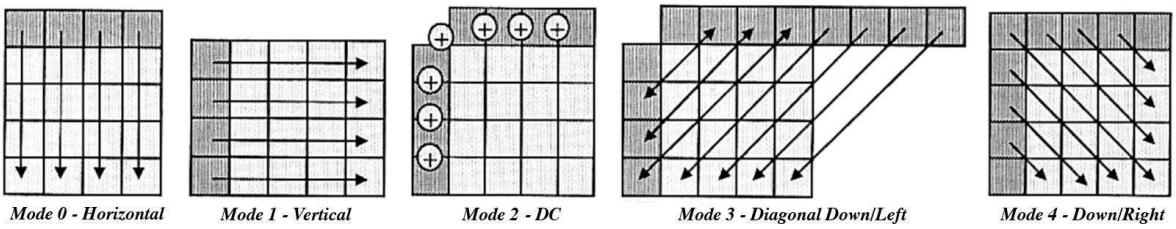


Figure 2.6: Illustration of five of the nine possible prediction directions [1]

In H.264, the notion of *Instantaneous Decoder Refresh* (IDR) frames is introduced, which are a special case of standard I-frames. They technically do not differ from normal I-frames except from one constraint on their position in the video stream, i.e., no frame *after* the IDR-frame can refer back to any frame *before* the IDR-frame. Since any decoder can only begin playback starting with an IDR-frame, their frequency determines the random access capabilities. The IDR rate is to be set by the user prior the encoding. If the minimum IDR-frame

interval is set to one, each I-frame is considered as IDR and playback can start at the beginning of every new GOP.

Predictive Coding: P- and B-Slices

H.264 inherits from MPEG-2 the concept of predicted pictures of type P and bidirectionally predicted B-frames. A lot of improvements and new techniques have been introduced in H.264 in order to further increase the efficiency of predictive coding. Some important enhancements are briefly presented in the following. The list is not exhaustive and a complete revision would go beyond the scope of this work. We focus on details that have an impact on compressed domain processing.

- H.264 features *Variable Block-Size Motion Compensation* (VBSMC). For each MB of size 16x16, the encoder dynamically decides to further sub-divide it into sub-MB partitions or blocks of variable size. The decision is based on coding efficiency, so after trying multiple different configurations and MB modes, the encoder choses the one for which the least number of coded bits is needed. To limit the computational overhead, the number of MB partition sizes is limited to seven possible choices, illustrated in Fig 2.7. If a MB is coded in one of the first three modes (16x16, 16x8, 8x16), it cannot be subdivided any further. For the fourth MB coding mode 8x8, the encoder decides for each of the four 8x8 blocks independently if it will be subdivided a second time. The four possible block partition sizes in this case are 8x8, 8x4, 4x8, and 4x4.

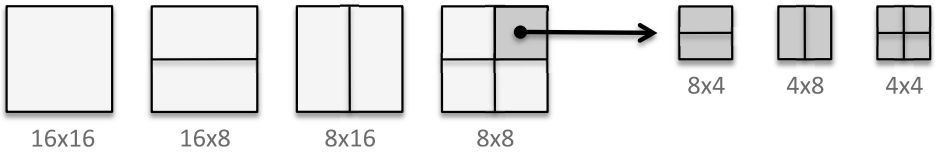


Figure 2.7: The seven possible MB partition sizes in H.264

- In H.264, multiple reference pictures per B-frame are allowed which are organized in two lists: LIST_0 holds the reference frames that lie temporally in the past and LIST_1 references future frames. Up to 16 reference frames may be used per B-frame, in contrast to MPEG-2, where the imposed limit is two in B-frames and one in P-frames. Hence, for a 16x16 MB in H.264, a maximal number of 32 MVs may be present, where a corresponding region in MPEG-2 can hold only up to two. This particular feature allows modest improvements in bit rate and quality in most scenes, but in certain types of scenes, such as those with repetitive motion, back-and-forth scene cuts or uncovered background areas, it allows a significant reduction in bit rate while maintaining clarity.

- The accuracy of the motion compensation process has been improved from half-pel precision in MPEG-2 to quartel-pel precision in H.264. This results in increased MV accuracy and lower prediction residuals.
- H.264 allows for motion estimation beyond the picture boundaries, which was unsupported in MPEG-2. This feature is specially useful in moments of camera motion like panning or tilting since it allows the motion estimation of boundary macroblocks, even if pointing beyond the frame limits.
- In MPEG-2, P-slice MBs in SKIP mode represent static areas and do not have a MV assigned. In H.264, they may either correspond to static areas or to regions of constant motion, so they may have a motion vector assigned that is copied from a neighboring MB.
- The corresponding P-frame SKIP mode is called DIRECT in H.264 B-frames coding. In this mode, no additional MV information is transmitted, since the MV is derived from neighboring MBs.
- Weighted prediction is introduced that allows to offset and scale the input signals during motion compensation. This is of particular interest during the coding of scene transitions like fade-to-black or illumination changes in the scene.

An example of MB partitions and the associated MVs of H.264 is shown in Fig. 2.8. When compared to MPEG-2 (see Fig. 2.4), it can be noticed how the average MB partition size diminishes in highly detailed moving regions. As a consequence, H.264 coded frames contain more MVs on average and their density is unevenly distributed over the image.

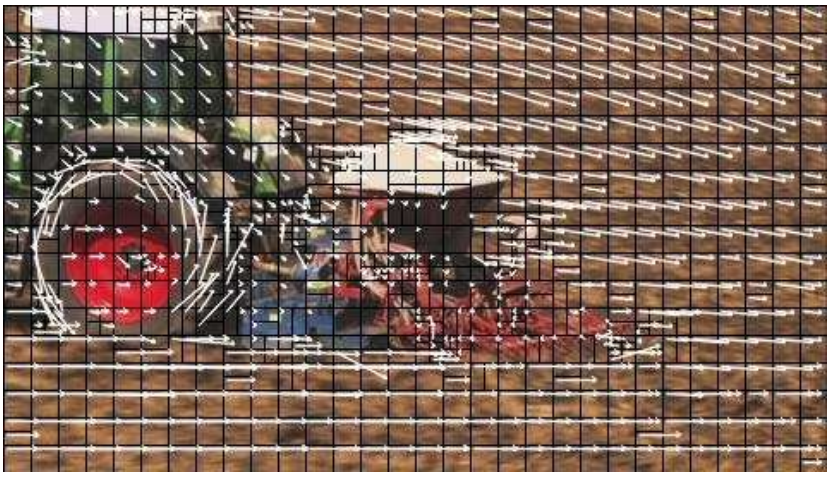


Figure 2.8: Example of MBs and the associated MVs in a H.264/AVC stream. Original sequence ©*Taurus Media Technik*

Additional Features

H.264/AVC introduces a large number of additional features, some of which are highlighted in the following.

- H.264 includes a number of loss resilience features, e.g., *Flexible Macroblock Ordering* (FMO), *Arbitrary Slice Ordering* (ASO), *Redundant Slices* (RS) and *Data Partitioning* (DP). These features provide the ability to separate more important and less important syntax elements into different packets of data, enabling the application of unequal error protection and generally increasing the error/loss robustness.
- In order to better support professional studio work, the possible sample bit depth can range from 8 to 14 bits per sample, depending on the selected profile. Furthermore, coding in lossless coding mode is supported.
- An in-loop deblocking filter helps to avoid block artifacts common to other DCT-based image compression techniques, resulting in better visual appearance and compression efficiency.
- Two new entropy coding schemes are introduced, namely the high-performance *Context-Adaptive Binary Arithmetic Coding* (CABAC) and the lower-complexity alternative *Context-Adaptive Variable-Length Coding* (CAVLC). CABAC compresses data more efficiently than CAVLC but requires considerably more processing to decode.
- Switching slices, called SP and SI slices, allowing an encoder to direct a decoder to jump into an ongoing video stream for such purposes as video streaming or bit rate switching. Switching slices are only supported in the Extended Profile (see below).

Although the introduced coding techniques lead to a significant increase in coding efficiency, the encoding process is also computationally more expensive compared to MPEG-2. Regarding the complexity of the different coding stages, motion compensation alone accounts for about 80 % of the encoder complexity [17].

Profiles and Levels

Similar to MPEG-2, H.264 defines a number of profiles and levels that are targeted towards different application scenarios. Up to now, 16 different profiles are defined. Four of them specify all-intra coding profiles that are designed for professional production applications. The latest three introduce scalability features and have been added as a result of the SVC extension in 2007 (see Sec. 2.3.3 – H.264/Scalable Video Coding). Each scalable profile extends one of the profiles that are included in the original version of the AVC standard, namely the *Baseline Profile* (BP), the *High Profile* (HiP) and a generalized *High Intra Profile* (HIntraP).

The **baseline profile** aims at low-cost applications with limited resources, so most of all mobile devices. Hence, the use of computationally expensive and memory consuming B-slices and the CABAC entropy coding scheme is prohibited. Most of the devices that are targeted by this profile are usually connected to the network via error-prone wireless links, so error resilience tools like FMO, ASO and RS are enabled.

The **high profile** was primarily designed for broadcast and storage applications. Among other things it features B-slices, CABAC entropy coding and 8x8/4x4 transform coding adaptivity to increase coding efficiency. The HiP was adopted in both the HD-DVD and the Blu-ray disc specification.

For professional studio work like production and post-production, four different **High Intra Profiles** have been defined: High 10 Intra, High 4:2:2 Intra, High 4:4:4 Intra and CAVLC 4:4:4 Intra. All of them are constrained to the use of intra-coded frames only, which enables direct frame access without motion compensation. It is comparable to Motion-JPEG2000 in the sense that it does not feature inter-frame prediction since each picture is coded independently like a still image.

A complete review of AVC is not within the scope of this work. The interested reader is referred to the official standard[7], introductory articles like the overview paper published by Wiegand et al in [1] or Richardson's book entitled *H.264 and MPEG-4 Video Compression* [7].

2.3.3 H.264/Scalable Video Coding (SVC)

The SVC standard is an extension to H.264/AVC (see Sec. 2.3.2) and enables spatial, temporal and quality scalability. SVC adds three additional, scalable profiles to the H.264 standard. Scalability with respect to coded video refers to the property that a video stream or file contains multiple layers, where lower layer versions of the same video can be extracted by ignoring or removing higher layer parts of the bit stream. The goal of scalable video coding is to obtain one multi-layer stream that is more efficient than multiple co-existing single-layer streams with respect to the number of coded bits and regarding decoding complexity. During the design process of SVC, the following essential requirements have been addressed [2]:

- Similar coding efficiency compared to single-layer coding for each subset of the scalable bit stream.
- Little increase in decoding complexity compared to single-layer decoding that scales with the decoded spatio-temporal resolution and bit rate.
- Support of temporal, spatial, and quality scalability.

[7] ISO/IEC 14496-10: `http://www.itu.int/rec/T-REC-H.264`

- Support of a backward compatible base layer (H.264/AVC in this case).
- Support of simple bit stream adaptations after encoding.

This section gives a brief overview of the provided coding techniques in order to meet the mentioned requirements. The basic SVC encoder structure is shown in Fig. 2.9. The first important property to notice is the H.264/AVC compatible base layer (Layer 0 in Fig. 2.9) that can be fully decoded by any pre-SVC implementation capable of processing H.264/AVC streams. The internal processing of higher layers is also performed with techniques provided by AVC, but the traditional single-layer spatio-temporal prediction is extended by inter-layer prediction of intra-coded frames, of motion information and of prediction residuals.

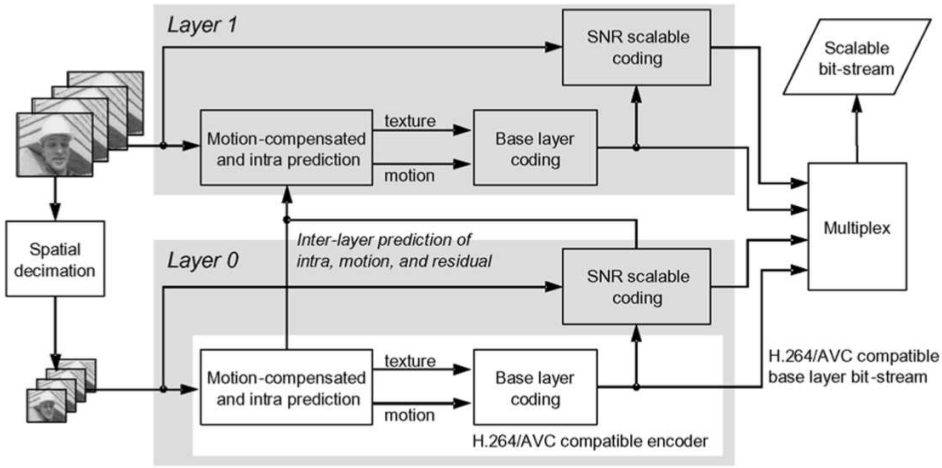


Figure 2.9: SVC encoder structure [2]

A short revision of the techniques that enable all three major types of scalability is provided below, as well as remarks on how each type of scalability influences the coding efficiency.

Temporal scalability

Temporal scalability refers to the ability to extract different temporal resolutions, i.e., to decode and play video at reduced frame rates while ignoring any information related to dropped frames. Although former standards like MPEG-2 and MPEG-4/Visual also incorporate to concept of temporal scalability, SVC provides a significantly increased flexibility for temporal scalability because of its reference picture memory control. It allows the coding of picture sequences with arbitrary temporal dependencies, which are only restricted by the maximum usable *Decoded Picture Buffer* (DPB) size.

Hence, for supporting temporal scalability with a reasonable number of temporal layers, no changes to the design of H.264/AVC have been made. The only related change in SVC

refers to the signaling of temporal layers, denoted as *Hierarchical Prediction* (HP). The underlying principle is simply that lower layer frames are not allowed to be predicted from higher layer frames. Through HP, temporal downscaling is achieved by discarding higher layer B- or P-slices.

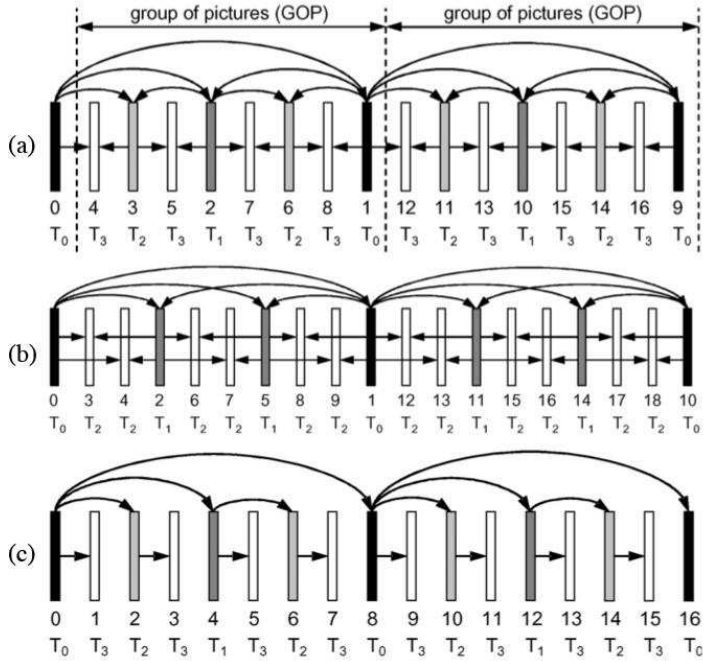


Figure 2.10: Hierarchical prediction structures for enabling temporal scalability: (a) coding with hierarchical B or P pictures, (b) non-dyadic hierarchical prediction structure, (c) hierarchical prediction structure with a structural encoder/decoder delay of zero. The numbers directly below the pictures specify the coding order; the symbols T_k specify the temporal layers with k representing the corresponding temporal layer identifier [3].

Fig. 2.10 illustrates the three supported types of hierarchical prediction, where the most common and efficient is hierarchical B-picture prediction (Fig. 2.10a) that enables a dyadic reduction of the frame rate at each temporal level. SVC provides a second scheme for scenarios where a non-dyadic step width is required, illustrated in Fig. 2.10b. For time critical applications like video conferencing, the referencing of future frames may introduce an unacceptable structural decoding delay. The third scheme, shown in Fig. 2.10c, is dedicated to such low-latency applications and only references past video frames during motion compensation.

Temporal scalability can be typically achieved without losses in rate-distortion performance [2], most of all by performing hierarchical B-picture prediction. However, the use of non-dyadic or low-delay schemes typically decreases the coding efficiency.

Spatial Scalability

Spatial scalability in SVC is based on the conventional approach of multilayer coding, which can be also found in former standards like MPEG-2 or MPEG-4/Visual. The basic idea is illustrated in Fig. 2.11, which also underlines the fact that different types of scalability can be combined. In the shown example, the spatial base layer comes at a lower temporal resolution than the enhancement layer.

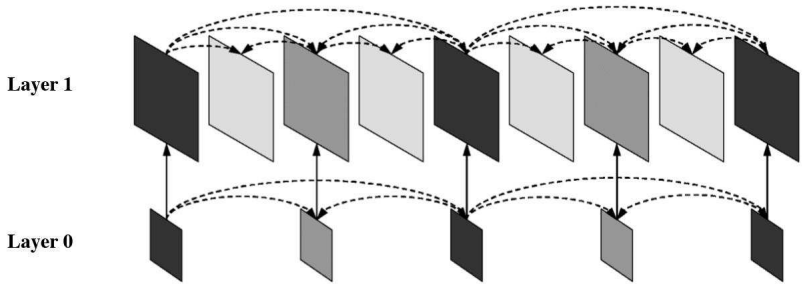


Figure 2.11: Multilayer structure with additional inter-layer prediction for enabling spatial scalable coding [2]. Dashed lines denote intra-layer prediction, solid lines inter-layer prediction.

In each spatial layer, intra-prediction and motion-compensated prediction are employed as for single-layer coding. Different from previous MPEG standards where inter-layer prediction was based uniquely on an up-sampled version of the reconstructed base layer, two additional inter-layer prediction mechanisms are introduced in SVC to increase efficiency. The three available inter-layer prediction concepts are chosen on a MB basis, allowing an encoder to select the coding mode that gives the highest coding efficiency. The three modes are:

- **Inter-layer intra prediction**: In this mode, illustrated in Fig. 2.12a, the corresponding reconstructed intra-signal of the reference layer is up-sampled and no additional side information is transmitted. For up-sampling the luma component, one-dimensional 4-tap FIR filters are applied horizontally and vertically. The chroma components are up-sampled by using a simple bilinear filter. Filtering is always performed across sub-MB boundaries using samples of neighboring intra-blocks. When the neighboring blocks are not intra-coded, the required samples are generated by specific border extension algorithms. To prevent disturbing signal components in the prediction signal, the H.264/AVC deblocking filter is applied to the reconstructed intra-signal of the reference layer before up-sampling.

- **Inter-layer macroblock mode and motion prediction**: If at least one of the co-located reference layer blocks is not intra-coded, the enhancement layer MB is inter-picture predicted as in single-layer H.264/AVC coding. The MB partitioning – specifying the

decomposition into smaller blocks with different motion parameters – and the associated motion parameters are completely derived from the co-located blocks in the reference layer (see Fig. 2.12b). This concept is also referred to as inter-layer motion prediction.

- **Inter-layer residual prediction**: This inter-layer prediction tool targets a reduction of the bit rate required for transmitting the residual signal of inter-coded macroblocks. With the usage of residual prediction, the up-sampled residual of the co-located reference layer blocks is subtracted from the enhancement layer residual (difference between the original and the inter-picture prediction signal) and only the resulting difference, which often has a smaller energy then the original residual signal, is encoded using transform coding as specified in H.264/AVC (see Fig. 2.12c).

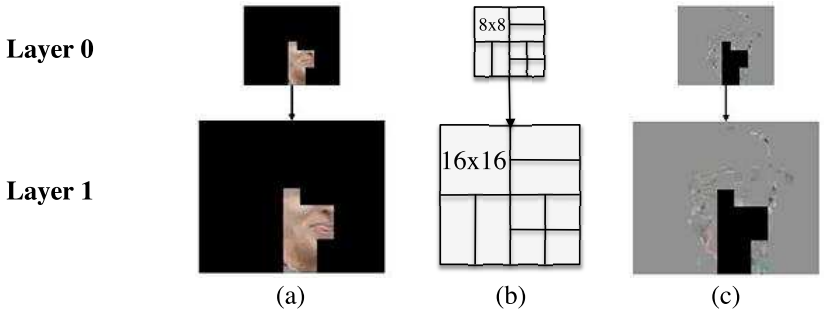


Figure 2.12: Illustration of inter-layer prediction tools: (a) up-sampling of intra-coded macroblock for inter-layer intra prediction, (b) up-sampling of macroblock partition in dyadic spatial scalability for inter-layer prediction of macroblock modes, (c) up-sampling of residual signal for inter-layer residual prediction.

An important feature of SVC is that only a *single motion compensation loop* is required for each spatial enhancement layer. For the employed reference layers, only the intra-coded macroblocks and residual blocks that are used for inter-layer prediction need to be reconstructed (including the deblocking filter operation) and the motion vectors need to be decoded. The computationally complex operations of motion-compensated prediction and the deblocking of inter-picture predicted macroblocks only need to be performed for the target layer to be displayed.

Concerning the efficiency, the bit rate increase relative to non-scalable H.264/AVC coding at the same fidelity can be as low as 10 % for dyadic spatial scalability. It should be noted that the results typically become worse as spatial resolution of both layers decreases and results improve as spatial resolution increases [2].

Quality Scalability

Quality scalability in SVC is regarded as a special case of spatial scalability where the base layer and the enhancement layer have identical size. The same inter-layer prediction tools are employed, but without using the corresponding up-sampling operations. This is referred to as *Coarse Grain Quality Scalability* (CGS). When utilizing inter-layer prediction for coarse-grain quality scalability, a refinement of texture information is typically achieved by re-quantizing the residual texture signal in the enhancement layer with a smaller quantization step size relative to that used for the preceding CGS layer.

However, this multilayer concept for quality scalable coding only allows a few selected bit rates to be supported in a scalable bit stream. In general, the number of supported rate points is identical to the number of layers. Switching between different CGS layers can only be done at defined points in the bit stream. Furthermore, the multilayer concept for quality scalable coding becomes less efficient when the relative rate difference between successive CGS layers gets smaller. Especially for increasing the flexibility of bit stream adaptation and error robustness, but also for improving the coding efficiency for bit streams that have to provide a variety of bit rates, a variation of the CGS approach, which is referred to as *Medium Grain Quality Scalability* (MGS), is included in the SVC design. The differences to the CGS concept are a modified high-level signaling, which allows a switching between different MGS layers in any access unit, and the so-called *key picture concept*, which allows the adjustment of a suitable tradeoff between enhancement layer coding efficiency for hierarchical prediction structures and *drift* for packet-based quality scalable coding. Drift describes the effect that the motion-compensated prediction loops at encoder and decoder are not synchronized, e.g., because quality refinement packets are discarded from a bit stream.

The key picture concept of SVC is illustrated in Fig. 2.13d, together with different former concepts that are employed by MPEG-2 or MPEG-4/Visual, shown in Fig. 2.13a-c. All pictures of the coarsest temporal layer are transmitted as key pictures, and only for these pictures the base quality reconstruction is inserted in the DPB. Thus, no drift is introduced in the motion compensation loop of the coarsest temporal layer. In contrast to that, all temporal refinement pictures typically use the reference with the highest available quality for motion-compensated prediction, which enables a high coding efficiency for these pictures. Since the key pictures serve as resynchronization points between encoder and decoder reconstruction, drift propagation is efficiently limited to neighboring pictures of higher temporal layers. The tradeoff between enhancement layer coding efficiency and drift can be adjusted by the choice of the GOP size or the number of hierarchy stages.

The efficiency of SNR scalability in SVC is comparable to spatial scalability. When applying an optimized encoder control, the bit rate increase relative to non-scalable H.264/AVC

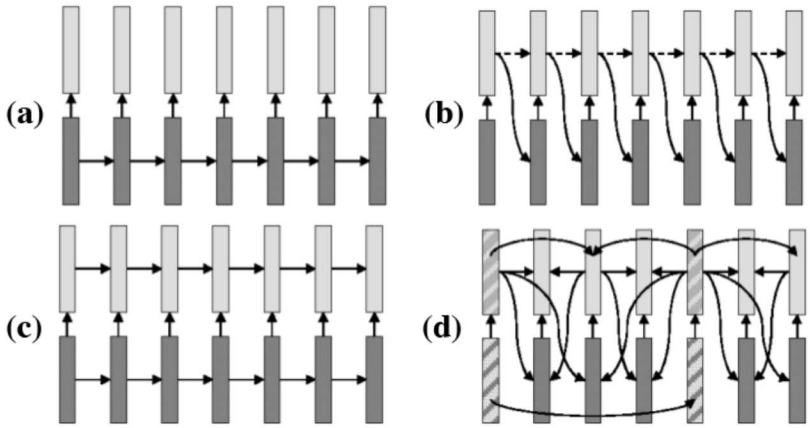


Figure 2.13: Various concepts for trading off enhancement layer coding efficiency and drift for packet-based quality scalable coding. (a) Base layer only control. (b) Enhancement layer only control. (c) Two-loop control. (d) Key picture concept of SVC for hierarchical prediction structures, where key pictures are marked by the hatched boxes [2].

coding at the same fidelity can be as low as 10 % for all supported rate points when spanning a bit rate range with a factor of 2-3 between the lowest and highest supported rate point [2].

2.4 Summary and Conclusions

After a brief introduction of video coding in general, an overview of the state-of-the-art in scalable video coding was provided. Focus was put on the two major scalable standards Motion JPEG2000 and the latest recommendation within the MPEG family, H.264/SVC. Since Motion JPEG2000 primarily targets professional high-quality production environments like Digital Cinema, only intra-coding mechanisms are employed and no valuable motion information is present in the coded stream.

H.264 with its scalable extension SVC represents a more versatile video coding framework. Its efficiency and bit stream scalability can be exploited on various levels and in different application scenarios. The following list gives some examples of applications that can benefit from SVC:

- **Video surveillance** can benefit from SVC in various aspects. With IP cameras that deliver SVC coded video[8], high-resolution video may only be transmitted if necessary in order to save bandwidth. Remote monitoring on portable devices as well as in-house visualization on televisions or computer screens can be realized by using the same bit stream.

[8]The first H.264/SVC surveillance applications are already available, e.g. at `http://www.aventuratechnologies.com/`

- **Streaming media** applications that dynamically adapts to different viewing devices and network conditions becomes possible. Media-Aware Network Elements (MANEs), which receive feedback messages about the terminal capabilities and/or channel conditions, can remove the non-required parts from a scalable bit stream before forwarding it. Thus, the loss of important transmission units due to congestion can be avoided and the overall error robustness of the video transmission service can be substantially improved.
- **Broadcasting** of a single, multi-layer stream can serve a wide range of different end-user devices at the same time. The end-user terminal can discard unnecessary parts of the bit stream according to the given decoding capabilities.
- **Video archiving** can benefit from scalable videos in terms of storage space management. The most recent videos can be kept in full resolution and quality for a certain period of time, while for old or less important sequences, higher layers may be dropped to free up storage space while still keeping a lower quality version for reference.

H.264 streams that are encoded with profiles other than the all-Intra class already include motion information, which is otherwise very time-consuming to post-estimate but which delivers valuable information concerning video indexing, analysis and computer vision related tasks. For instance for time-critical applications like surveillance, a first rough compressed domain analysis may be performed on lower spatial or temporal layers of the stream. Depending on the targeted application and under certain pre-defined circumstances, e.g., high motion activity is detected, the system may switch to computationally more demanding higher layers that provide more detail.

The following chapter describes how H.264 compressed domain information can be extracted and processed in order to assist various indexing and analysis applications, some of which are presented in detail in Part II of this work.

Chapter 3

Analysis

Traditionally, the majority of indexing and computer vision related tasks are performed on raw, decoded images. Throughout this work, we refer to this as image domain analysis. Input pictures are usually at first projected into a suitable color space like RGB, YUV or HSV. Further analysis takes place either on pixel level on one or all of the color planes, or, depending on the application, in another image representation domain like the *Difference of Gaussians* (DOG).

While frame-wise, pixel-level brightness and color information is extremely rich, it is also very demanding with respect to computational complexity and memory consumption. The resulting high computational complexity becomes even more problematic regarding the rapid deployment of high definition cameras and content, where resolutions of 1920x1080 pixels are reached in television broadcasting and Blu-ray discs.

Different approaches exist in order to handle this flood of information, where the most common one is to down-scale the input images prior to processing. Since this approach diminishes the amount of information that needs to be processed, input videos or images initially still have to be entirely decoded and read in full resolution. Moreover, proper down-scaling itself also is a computationally expensive and memory demanding operation.

Except from professional production environments, content is most often distributed and stored in encoded form to save bandwidth and storage space. In this work, we take advantage of this fact and pursue another approach to decrease the computational overhead, called *compressed domain analysis*. Contrary to image domain analysis, this approach does not necessitate full stream decoding back to pixel level. Instead, the coded image representation itself is used in order to deduce more high level information about the actual video content. In case of MPEG coded video, motion vectors and if available, transform coding coefficients, are the main source of information. The main pros and cons of compressed domain analysis compared to image domain analysis are listed below.

Pros:

⊕ No full stream decoding is necessary, resulting in less computation time and memory consumption.

⊕ Additional information like motion is directly available, since the costly task of motion estimation has already been performed by the encoder.

⊕ Most videos are stored and distributed in compressed form and the original versions are not available.

Cons:

⊖ The provided information is rough and sparse, since no color and pixel level information is available.

⊖ Compressed domain algorithms are codec and implementation dependent, so less general than image domain approaches.

Certain applications like face detection and recognition are less suited for pure H.264 compressed domain analysis due to the mentioned limitations. Other applications such as the estimation of camera motion benefit from the additional information present in compressed streams, most notably motion vectors. Like also stated by Cédras in [18], motion perception plays a very important role in the human visual system and helps us to recognize objects and their trajectories, infer their relative depth or estimate their rigidity. In this work we exploit the compressed domain motion information to a large extent in order to perform global motion estimation, unsupervised object detection and tracking.

This chapter introduces and explains the concepts of compressed domain analysis at the example of H.264 streams. For a better understanding of the presented algorithms, information about the used test sequences and how they are encoded is provided in Sec. 3.1. Global motion estimation builds an important basis for further processing and is presented in Sec. 3.2, followed by a description of methods to perform scene segmentation in background and moving foreground objects in Sec. 3.3. Details on the proposed compressed domain tracking algorithm are given in Sec. 3.4. A method to estimate the relative distance between moving objects and the camera is presented in Sec. 3.5. We finally close this chapter by concluding remarks in Sec. 3.6.

It has to be noted that all presented algorithms can be applied to single-layer AVC as well as multi-layer SVC streams. In case of SVC streams containing multiple spatial layers, some of the methods take advantage of the scalability in order to increase either the efficiency or the quality of the results.

All techniques in this chapter are of purely analytical nature, i.e., they do not alter the bit stream, reflected by the term *analysis* in contrast to *processing*. In Chap. 7, we present methods and ideas that *process* the input bit stream by modifying the original motion vector fields.

Since each of the mentioned problems stands as a research topic for itself, related work is presented separately in the respective sections.

3.1 Test Videos

Analysis results for a variety of videos with different origins are presented throughout this work. All of them represent natural video scenes that have been shot either indoor or outdoor with different kinds of cameras, ranging from professional high definition to surveillance cameras. While some of the sequences are shot with fixed cameras, others contain camera operations like panning, tilting or zooming. Figure 3.1 shows example screenshots of some of the used test videos; others are provided in upcoming chapters. The sequence *street* which is depicted in Fig. 3.1a will be used as a running example throughout this chapter, so multiple screenshots are shown. The sequence was shot outdoor and shows a man who walks around an obstacle to pick up his bike and walk away. The camera is panning and tilting to follow the person's motion in the beginning of the scene and remains fix in the end.

All videos have been encoded with JSVM, the reference implementation of H.264/SVC, in version 9.8. Among with the final draft of the standard and a series of test sequences, it is publicly available for download at [19]. Reference software for encoding and decoding is provided. The encoder accepts raw YUV files as input and the decoder also outputs raw YUV files. Additional tools to perform down-scaling of the raw YUV input videos and to extract particular spatial or temporal layers are provided as well. It has to be noted that JSVM works well for research purposes, but encoding and decoding are slow since the implementation represents a non-commercial proof-of-concept without optimizations.

Due to the lack of alternative available implementations of SVC at the beginning of this thesis, JSVM was used to carry out all encoding and decoding. By the time of writing, the OpenSVC decoder [20], a more efficient open source implementation of a pure SVC *de*coder has reached mature state and achieves a decoding speed-up of factor 16-52 compared to JSVM[1].

The majority of test sequences have been encoded with similar parameters. The most important common encoder settings are listed in Table 3.1. For reference, an introduction to

[1]OpenSVC decoder: `http://sourceforge.net/apps/mediawiki/opensvcdecoder/`

(a_1) *street.* frame 37 (a_2) *street.* frame 57 (a_3) *street.* frame 97 (a_4) *street.* frame 117

(b) *parkrun* (c) *kun fu* (d) *man in restaurant* (e) *train tracking*

(f) *hall monitor* (g) *car* (h) *surveillance* (i) *traffic*

Figure 3.1: Screenshots of some of the used test sequences

scalable video coding in general and more specifically, to the standard H.264/AVC and its extension SVC is provided in Chap. 2.

The original resolution of the test videos ranges from CIF (352x288 pixels) to the Full-HD format 1080p (1920x1080 pixels). The number of encoded spatial layers per video depends on the original frame size. In most cases, the resolution of the spatial base layer was set to CIF or similar and dyadic spatial scalability was employed for the sake of coding efficiency. To give examples, we encoded CIF content as single-layer AVC streams, SD content of size 704x576 with two spatial layers (0: 352x288, 1: 704x576) and 1080p HD content with three spatial layers (0: 480x270, 1: 960x540, 2:1920x1080), so including one layer with full, half and quarter resolution, respectively. The choice of the minimal base layer size was based on the screen resolution of popular H.264 capable mobile devices like Apple's first generation iPhone, having a display with resolution 480x320 pixels[2].

Quality scalability is inherent to the streams and temporal scalability is enabled by the hierarchical B-picture prediction structure of SVC, illustrated in Fig. 3.2 for a GOP size of 8 frames. Depending on the application, the GOP size was set at 8 or 16 pictures, with each I-frame being also an IDR-frame to enable random stream access. Lower temporal resolutions are obtained by dropping B-frames in inverse coding order. Since the methods we present in

[2] `http://www.apple.com/iphone/specs.html`

Parameter	Value
Video format	H.264/SVC
Profile	Scalable High Profile
Software encoder	JSVM 9.8
GOP size	16 or 8 (fix per video)
IDR frequency	1 (each I-frame)
Reference frames per List	1 (max. 2 per B-frame)
Base layer mode	1 (AVC compatible)
Number of spatial layers	Up to 3 (0:480x270, 1:960x540, 2:1920x1080 for 1080p)
Frame rate	12-25 fps for surveillance, 25 fps otherwise
Temporal scalability	Dyadic scalability with hierarchical B-pictures
Motion search	Block search

Table 3.1: Summary of important JSVM encoder settings

the following rely on motion, i.e., the presence of predicted frames of type B or P, the lowest temporal layer that still contains motion information and can be processed has the structure IDR-B-IDR-B, where IDR stands for *Instantaneous Decoder Refresh*, a special type of I-frame (see Chap. 2 for more details).

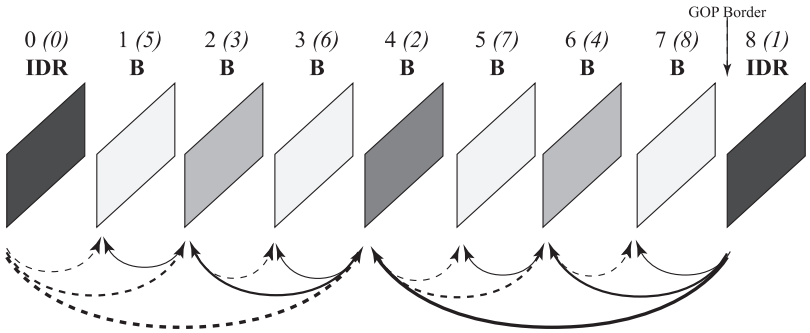


Figure 3.2: Hierarchical B-picture prediction structure of test videos. The numbers on top: display order (coding order). Dashed arrows denote LIST_0 prediction, solid lines stand for LIST_1 prediction.

Besides some surveillance sequences with low frame rates of down to 12 fps, the remaining test videos are encoded at 25 fps with full, progressively scanned images without interlacing. Originally interlaced source videos have been deinterlaced prior to encoding.

In order to give an idea about the achieved bit rates, Fig. 3.3 shows the coded versus the raw video bit rate as a function of the input resolution, represented by the frame width. The values have been obtained by scaling and encoding the same sample input video (sequence *street*) at different resolutions. For each resolution, encoding as high quality, single layer H.264/AVC stream was performed at similar parameters and quality settings. Besides tremendous bit rate savings compared to raw YUV (4:2:0) video, it can also be noted that

the coded bit rate scales with the input resolution in the same manner as the uncoded, raw video size.

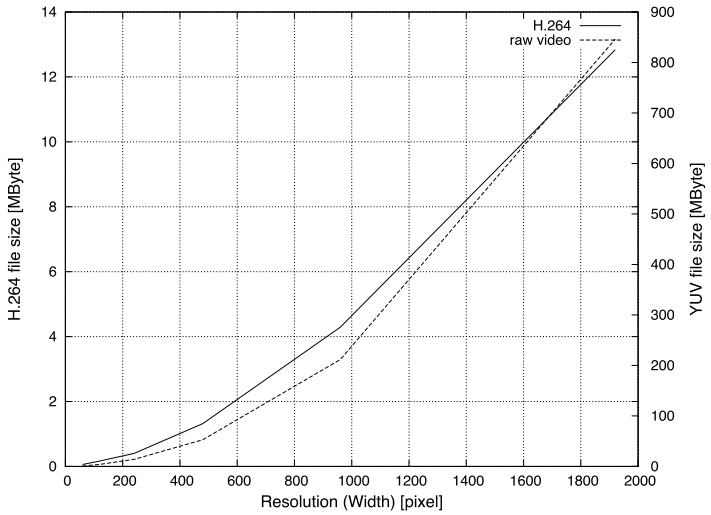


Figure 3.3: Video resolution vs. file size of raw YUV input compared to coded H.264 streams

The majority of algorithms presented in this work is scene based, i.e., we assume separate video scenes without any cuts or transitions. This can be achieved by first applying a compressed domain shot boundary detector. Examples for methods that have been specifically designed for H.264 streams include the approach proposed by Liu et al. [21], which uses Hidden Markov Models (HMMs) for scene cut detections, and an approach by De Bruyne et al. [22, 23]. The latter measures the temporal correlation between successive frames to detect gaps in the temporal prediction chain. When this gap is due to I- or IDR-frame coding, the spatial dissimilarity is calculated to identify abrupt transitions. To locate gradual changes, the percentage of intra-coded macroblocks is analyzed over time, where an increase is typically caused either by gradual transitions or by local or global motion. By analyzing the motion intensity of the estimated foreground and background, these different types of content changes can be distinguished.

In the following section, different methods to estimate the global scene motion in the MPEG compressed domain are presented.

3.2 Global Motion Estimation

The global scene motion describes how the the background area moves with respect to the camera over time. Assuming that the background itself is static, global motion is usually caused by camera operations like panning, tilting, zooming, rotation or traveling.

Robust, frame-wise estimates of the global scene motion can be exploited in several ways for analysis and indexing tasks. Regarding indexing, the frame-wise global motion can be used as a temporal descriptor for video retrieval (see Chap. 6). Concerning analysis and vision tasks, global motion estimation (GME) is needed for motion compensation in case camera motion is present. The estimation results themselves can be used for further tasks like the detection of moving objects (see Sec. 3.3). Since block-based video codecs that employ motion-compensation already contain motion information in the form of motion vectors, GME can be carried out very efficiently in the compressed domain.

3.2.1 Related Work

Depending on the application, different methods to perform GME in the compressed domain have been proposed. Most approaches are based on the motion vectors that are present in MPEG-2 coded streams. The re-use of motion information is the main reason why compressed domain approaches are more efficient than those working on pixel level, because the computational expensive task of motion estimation has already been performed by the encoder.

Kobla [24] proposed a global motion classification method within a video indexing framework. He uses camera operation estimates to segment video scenes into different shots, scenes and sub-scenes. Directional MV histograms are employed to determine the dominant, translational camera movement and zoom detection is based on focus estimation. The approach delivers a macroblock-based classification of the dominant moving direction and the presence of zoom, but no quantitative values are estimated.

Yoo et al. [25] analyze camera operations in the MPEG-2 domain based on the generalized Hough transform. They start by approximating the optical flow (OF) given the MV field as input, followed by spatio-temporal vector median filtering with window-size 3x3x3. A parameter space is constructed from the vector displacements Δx and Δy, where the largest population or density is taken as the amount of pan and tilt.

Hesseler et al. [26] apply two-dimensional motion vector histograms constructed from P-frame MV fields. AC-coefficients from the MPEG-2 domain are used to detect outliers. Peaks in the histograms are regarded as the dominant translational motion and are detected via pyramid filters. Since this approach delivers robust results in case of pure translational motion, it does not account for complex camera motion like zooming, rotation or composed motion.

If robust global motion estimates that cover complex camera operations are required, the majority of approaches, e.g., Wang et al. [27], Bouthemy et al. [28, 29] and Durik et al. [30] apply iterative motion estimation and outlier rejection algorithms based on some form of weighted least squares estimation of a 2-D parametric motion model. Due to the proven

robustness, we adopted an algorithm similar to the one proposed in [30] and [31] and ported it to the H.264 compressed domain. An important advantage of this approach is the iterative outlier rejection scheme, which as a by-product delivers a rough segmentation of the scene in background and foreground regions. The algorithm is described in detail below.

3.2.2 Algorithm

The global motion estimation method consists of a multi-resolution scheme that incorporates an iterative re-weighted least squares estimation of the well known 2D 6-parameter affine motion model. The application of a multi-resolution scheme increases robustness results and fits well in our work given the scalability of SVC streams. After parameter estimation, the individual camera operations are interpreted from the resulting model parameters.

Compared to MPEG-2, some modifications of the algorithm are necessary in order to support H.264 coded video. The main differences regarding the MV fields in MPEG-2 and H.264 streams are the variable block-size in H.264, the increased flexibility regarding reference pictures, the use of B-frames only in the hierarchical B-picture prediction structure and the increased MV precision.

MV Field Extraction

For the extraction of macroblock modes, partition sizes and motion vectors from H.264 streams we modified the publicly available reference decoder JSVM [19] in version 9.8. The MB prediction modes and partition sizes are signaled and can be obtained by parsing the bit stream. In order to retrieve the MV values in quarter-pel precision, the entropy coding of H.264 has to be reversed as the only decoding step. The main source for MVs in temporally scalable SVC streams are bi-directionally predicted B-frames. The extraction of MVs happens on a sub-MB basis and depends on the prediction mode of the respective MB and its sub-partitions:

- **Mode LIST_0**: The MB is predicted backwards only, comparable to P-frame MBs. In this case, we extract the MV from LIST_0 and inverse the signs of the Δx and Δy components.
- **Mode LIST_1**: The MB is forward predicted only. In this case, we extract the MV from LIST_1 and keep the original signs of Δx and Δy.
- **Bi-Prediction**: The MB is bi-directionally predicted. We extract the MVs from both lists, inverse the sign of the LIST_0 MV and calculate the average over both lists, weighted by the distance to the respective reference frame, which is similar for both MVs in the case hierarchical prediction is employed (see Fig. 3.2).

- **Direct mode**: In this mode, no MV is explicitly coded for the MB, but it is predicted from the MVs of neighboring MBs.

To obtain uniform results, all MVs are scaled by the distance to their respective reference pictures. As a MV field approximation for I-frames, which do not contain motion information, we take the mirrored LIST_0 vectors from the subsequent B-frame in display order as an estimation basis.

Motion model

A model is needed that is able to explain the observed scene motion. The instantaneous image velocity field of a 3D scene models the inter-frame displacements and can be expressed as [32, 33]:

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} -(\frac{T_X}{Z} + \Omega_Y) + x\frac{T_Z}{Z} + y\Omega_Z - x^2\Omega_Y + xy\Omega_X \\ -(\frac{T_Y}{Z} + \Omega_X) + y\frac{T_Z}{Z} + x\Omega_Z - y^2\Omega_X + xy\Omega_Y \end{bmatrix}, \tag{3.1}$$

where $(u(x,y),\ v(x,y))^T$ denotes the image velocity at image location (x,y), $T = (T_X, T_Y, T_Z)^T$ denotes the translational motion of the camera, $R = (\Omega_X, \Omega_Y, \Omega_Z)^T$ denotes the camera rotation and Z the depth of the scene point corresponding to (x,y).

According to [34], the approximation of instantaneous image motion of a general 3D scene can be approximated in a simplified way by a 2D parametric motion model under the following assumptions regarding the scene geometry and/or camera motion: (i) the scene is planar, (ii) the 3D scene is sufficiently distant from the camera, or the deviations from a planar scene surface are small relative to the overall distance of the scene from the camera.

The 2D six-parameter affine model proved to be sufficiently rich to characterize motion observed in two successive video frames [35, 30]. With respect to block-based motion vectors in the form $(\Delta x_i, \Delta y_i)^T$, it can be formulated as

$$\begin{aligned} \Delta x_i &= a_1 + a_2(x_i - x_0) + a_3(y_i - y_0) \\ \Delta y_i &= a_4 + a_5(x_i - x_0) + a_6(y_i - y_0), \end{aligned} \tag{3.2}$$

where $(x_0, y_0)^T$ denotes the reference point in the image (e.g., the image center) and $(x_i, y_i)^T$ denotes the MB center. The six parameters $a_1 \ldots a_6$ describe an affine transformation between two planar views of the scene background. With this model, it is also possible to capture complex camera operations like panning and tilting while zooming.

Weighted Least Squares (WLS) Estimation

Given the extracted MV field as input, the estimation of the six model parameters given in Eq. 3.2 can be formulated as a linear regression problem. Assuming no outliers, the solution in the standard least squares sense can be formulated in common matrix notation as

$$\hat{\phi} = (H^T H)^{-1} H^T Z, \tag{3.3}$$

where Z holds the measured motion compensation vectors, H is the observation matrix containing the macroblock centers and $\hat{\phi} = (\hat{a}_1, \ldots, \hat{a}_6)^T$ is the vector containing the estimated motion parameters.

With N being the number of MVs per frame and $(\Delta x_i, \Delta y_i)^T$ denoting the components of MV number i, Z is given as

$$Z = (\Delta x_1, \ldots, \Delta x_N, \Delta y_1, \ldots, \Delta y_N)^T. \tag{3.4}$$

The observation matrix H is given as

$$H = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & x_N & y_N \end{bmatrix}. \tag{3.5}$$

The MV fields present in compressed streams are optimized in terms of coding efficiency and not with optical flow in mind. As a consequence, the MVs do not always correspond to true motion. In particular, MVs with random length and orientation appear in low-textured areas like blue sky or white walls. Furthermore, most video scenes contain moving objects, resulting in numerous MVs that do not represent the global background or camera motion. Since the standard least squares method is not robust against such outliers, a multi-resolution scheme with weighted least squares estimation on each level is applied in order to minimize a robust functional of motion residuals [30]. The objective is to minimize the residuals r between the measured motion vectors d and their estimates $\hat{d}$:

$$r_i = d_i - \hat{d}_i \tag{3.6}$$

In case of scalable streams, the multi-resolution levels are given by the different spatial layers, beginning with the spatial base layer. If single-layer streams are to be analyzed, the

algorithm iterates on the same spatial layer until the outlier configuration converges. After each iteration, outlier MBs are attenuated or rejected according to a weighting function of r. The weights w_i for each MV are obtained by the following Gaussian cost function, which limits the influence of outliers continuously:

$$w_i = \frac{\rho^2}{\sqrt{\pi}} e^{(-\frac{r_i^2}{\rho})}, \tag{3.7}$$

where ρ is set to the standard deviation σ (or to 1, if $\sigma < 1$) and r_i denotes the estimation residual after the last iteration.

The weights w_i are stacked into the diagonal weighting matrix W, given as

$$W = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & w_{2N} \end{bmatrix}. \tag{3.8}$$

After each iteration, the motion parameters are re-estimated in the weighted least squares sense, formulated in matrix notation as

$$\hat{\phi} = (H^T W H)^{-1} H^T W Z. \tag{3.9}$$

Outliers are iteratively attenuated or excluded from the estimation support by the weighting function given in Eq. 3.7, which is depicted in Fig. 3.4 for $\sigma = 2$. For further processing, we are interested in binary outlier masks. Hence, a threshold on the residual r is needed to determine which blocks are treated as foreground blocks. This threshold can be manually chosen or is otherwise set to the standard deviation σ.

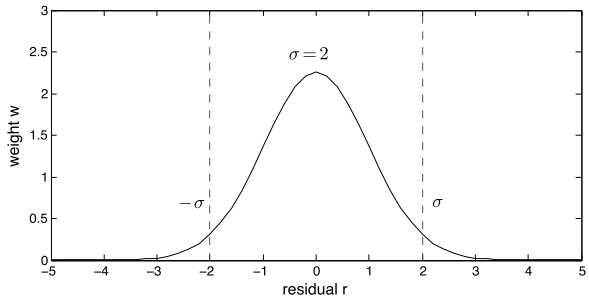


Figure 3.4: Outlier weighting function for $\sigma = 2$

As a final result, robust estimates of the six model parameters are obtained together with outlier masks.

Parameter Characterization

In order to obtain a more intuitive understanding of the model parameters $a_1 \dots a_6$, we translate them into another basis of elementary camera motion descriptors

$$\theta = (pan, tilt, zoom, rot) \tag{3.10}$$

by [29]

$$\begin{aligned} pan &= a_1, & tilt &= a_4, \\ zoom &= \gamma \cdot (a_2 + a_6), & rot &= \gamma \cdot (a_5 - a_3), \end{aligned} \tag{3.11}$$

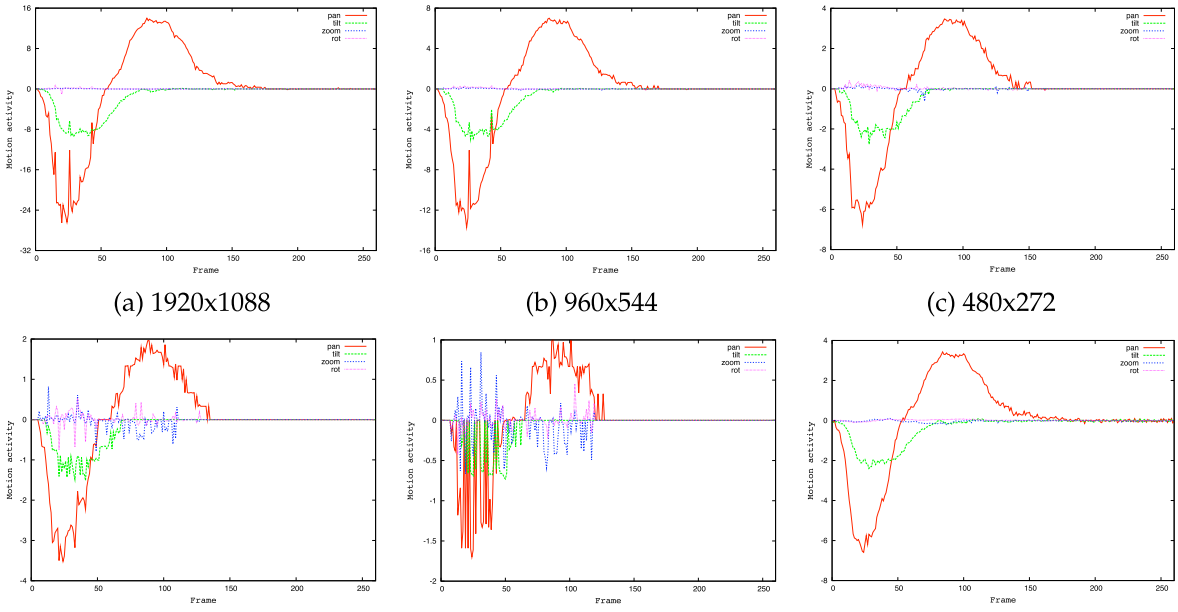
with $\gamma = \sqrt{height^2 + width^2}/4$ being a resolution dependent scaling factor to project *zoom* and *rot* in the same dimension as *pan* and *tilt*. The values *zoom* and *rot* represent ratios and not pixel values like the two translational motion parameters.

It has to be noted that with this approach, a pure camera pan, i.e., a rotation of the camera around its vertical axis, cannot be distinguished from sideways traveling. If the scene is distant, both types of camera operations result in approximately the same MV pattern, leading to non-zero values of a_1 (*pan*). The same applies for tilting versus vertical traveling and zooming versus forward or backward traveling.

3.2.3 Results

Exemplary results on five different spatial layers of the sequence *street* are given in Fig. 3.5a-e. Besides a scaling factor, the results at different spatial resolutions are very similar, showing the robustness and scalability of the method. Experiments showed however that at resolutions smaller than CIF, the results become very noisy, most notably on the variables *zoom* and *rotation*, but remain stable at higher resolutions. This is due to the small amount of remaining MVs at low resolutions. For reference, Fig. 3.5e shows the GME results obtained by the method proposed by Odobez et al. in [36] at a resolution of 480x272. This approach works on pixel level in the image domain and exploits the spatio-temporal derivatives of the image intensity function. An implementation of this incremental multi-resolution estimation method is available at [4].

Besides minor variations, the compressed domain results are identical to those obtained on pixel level. Image domain analysis leads to more stable results at low resolutions due to the dense estimation support.



(a) 1920x1088 (b) 960x544 (c) 480x272

(d) 240x144 (e) 128x80 (f) Motion2D on 480x272

Figure 3.5: (a)-(e): Compressed domain GME results for 5 different spatial layers. (f): Reference global motion results obtained with [4] on pixel level. Sequence *street*.

Computational Complexity

Concerning the computational complexity of the method, we achieve an average performance of 36 fps on an Intel Core2Duo with 2.16 GHz and 2 GB of RAM when analyzing the spatial base layer (480x272) extracted from a Full-HD stream. This enables real-time analysis. The achieved performance in terms of frames per second (fps) as a function of the video resolution is shown in Fig. 3.6. The black line denotes the real-time limit, which is 25 fps for our test sequences. In comparison to the image domain method presented in [36], which estimates similar information on pixel level, the achieved compressed domain analysis time is approximately seven to ten times faster.

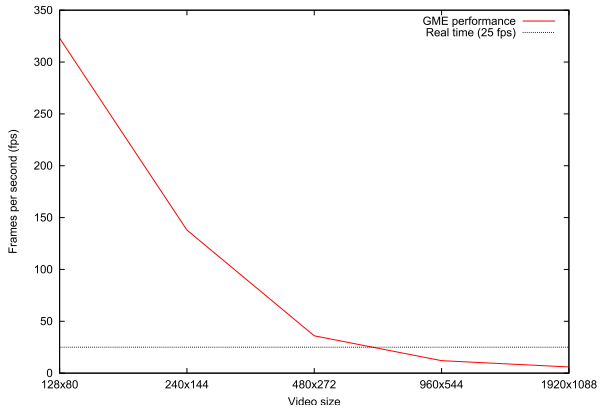


Figure 3.6: Computation time of the proposed compressed domain GME algorithm

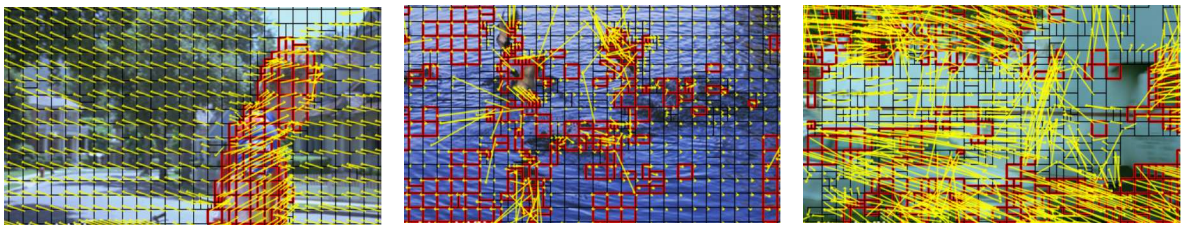
3.2.4 Limitations

Although the algorithm performs very well on a large variety of videos, it fails under certain conditions. We identified three problematic situations that compromise the estimation results:

1. Moving objects that occupy a large portion of the visible screen and thus hiding the background
2. Non-static, moving background (e.g., water)
3. Large, low-textured areas (e.g., flat white walls)

Figure 3.7 shows screenshots of different scenes together with the SVC macroblock partitions and their associated motion vectors. Detected outlier MBs are framed in red. A video under normal conditions is depicted in Fig. 3.7a. Here, the algorithm achieves to correctly identify outlier MBs and the camera motion estimation delivers reliable results. Figure 3.7b shows an example for the second case with non-static background and very noisy estimation results. A combination of the first and third problem can be observed in Fig. 3.7c. The camera is panning to the right while filming some quickly by-passing vehicles, which together nearly fill the whole screen. The motion-blurred white truck is weakly textured and parts of the MVs have random orientation. All of the mentioned problems affect the quality of the GME.

The first two situations pose severe problems to global motion estimation even in the pixel domain and can not be resolved without additional information. Concerning compressed domain object tracking in such cases, at least the initialization has to be the done either manually or on a decoded frame with color information available. The third problem,



(a) *Street* (b) *DucksTakeOff* (c) *Traffic*

Figure 3.7: Examples of SVC macroblock partitions and motion vectors. Outliers of GME are marked in red. MVs were scaled up for better visibility

low-textured areas, also disturbs the GME in the compressed domain. This is due to the block-matching algorithm of MPEG-type encoders. The MVs associated to the MBs are optimized in terms of coding efficiency and not with respect to the real optical flow, and with missing texture, the resulting MVs are either zero or show arbitrary orientations.

In Chap. 7 we present an approach to overcome the third problem by correcting the MVs in a joint indexing/coding framework in case precise GME results are required, e.g., for compressed domain mosaicing applications [37]. If the possibility of using a modified encoder is not provided, a combination with segmentation algorithms in the spatial image domain can be used to detect and process flat areas separately, e.g., like demonstrated in [38].

3.3 Object Detection

The detection of moving objects in video scenes is one of the fundamental problems regarding numerous indexing and vision tasks. Some of the applications which heavily rely on robust moving object detection are

- the construction of mosaics,
- background construction,
- object tracking and unsupervised initialization of tracking algorithms,
- behavior analysis and
- video surveillance.

The most popular approach in the pixel domain is background subtraction, where a model of the scene background is created for a fixed camera view and the foreground is obtained by subtracting the estimated background from the current image. Various background modeling techniques have been proposed in the literature, including basic methods like simple frame differencing or averaging, or more advanced methods such as back-

ground modeling through mixture of Gaussians [39], kernel density estimators [40], mean-shift based estimation [41] or Eigenbackgrounds [42]. While an introduction to pixel level foreground estimation techniques is not within the scope of this work, more details on background subtraction by using Gaussians Mixture Models (GMMs) are provided in Chap. 4 in the context of a traffic surveillance application.

An overview of existing compressed domain object detection methods is provided below, followed by an explanation of the approach we proposed in [43] regarding H.264 streams.

3.3.1 Related Work

A number of methods concerning object segmentation in the MPEG-2 domain can be found in the literature, including [44, 45, 46, 26, 47]. They usually rely on motion information and/or DCT coefficients, which both can be easily extracted from MPEG-2 coded videos.

Sukmarg et al. [44] presented an approach that works solely with DC images and AC coefficients. The segmentation algorithm consists of four main stages, i) the initial segmentation using sequential leader and adaptive k-mean clustering, ii) region merging based on spatiotemporal similarities, iii) foreground/background classification, and iv) object detail extraction. Object details are obtained by partial decoding of chosen MBs. Mezaris et al. [45] use an iterative rejection scheme based on the bilinear motion model for foreground/background segmentation. Yu [46] uses a combination of MV clustering and background subtraction of DC images to segment moving objects.

Treetasanatavorn et al. [48] presented a statistical Bayesian estimation framework for segmentation and tracking of moving regions from MPEG-4/Visual compressed videos. For each motion field, the algorithm initializes a partition that is subject to comparisons and associations with its tracking counterpart. Each tracked region is classified as a background or a foreground object based on an approximation of the logical mass, momentum, and impulse. Although the results on fixed-view standard test sequences like *Table Tennis* are promising, the method is very complex and no indications about the performance or compatibility for sequences with non-static cameras are given.

Another MPEG-4/Part 2 based approach that is limited to static cameras was proposed by Babu et al. in [49, 50]. They accumulate motion vectors (MVs) over time, followed by a K-Means clustering to determine the number of objects in the scene and the EM algorithm for object segmentation. Since only MVs are processed, the algorithm is in principle portable to the H.264 domain. However, the limitation to fixed view sequences imposes constraints on possible application scenarios.

The main reason why most algorithms based on MPEG-2 and MPEG-4/Visual cannot be ported to the H.264 domain is the use of DC and AC coefficients, e.g. like in [44, 46, 47]. H.264 employs intra-prediction in the spatial domain, so transform coding coefficients no

longer represent the total energy of the respective MB, but only an energy difference. Since in H.264 the prediction is already performed in the spatial domain, all former blocks would have to be decoded to obtain useful information, thus nullifying the computational benefit of compressed domain processing.

Only few approaches have been published that are specific to H.264. Zeng et al. [51] employ a block-based Markov Random Field (MRF) model to segment moving objects from the sparse MV field that is extracted from H.264 compressed streams. Object tracking is integrated in the uniform MRF model and exploits the object temporal consistency simultaneously. As a first processing step, the noisy input MVs are classified into the four distinct classes background, foreground, edge and noise. The classification is performed via thresholding of the MV magnitudes. Background MVs are for example defined as small MVs whose magnitudes are below a predefined background threshold, making the algorithm only applicable on fixed-view scenes. Liu et al. [52] segment the motion field into homogeneous regions and perform region merging with binary partition trees (BPT). Different moving objects are represented by selective nodes that are extracted by traversing the BPT.

An interesting approach that neither uses MVs nor transform coding coefficients was published by Poppe et al. in [53], which was extended to the multi-view case by Verstockt et al. in [54]. Object detection is performed by analyzing the number of coded bits per macroblock (MBs). This information can be obtained very efficiently through parsing of the compressed stream, so no decoding steps are necessary. Based on these coded MB sizes, a background model is created during a training period. New images are compared to this model to yield MBs that correspond to moving objects. Subsequently, these MBs are spatially and temporally filtered to remove noise. Finally, the size of the sixteen 4x4 sub-blocks within one MB are evaluated to make a more fine-grained segmentation. Although the approach is very efficient and the results promising, it suffers from the same problems as other pixel-domain background modeling techniques in that it requires the background to be static, so no camera motion is supported.

3.3.2 Algorithm

Our goal is the development of an efficient compressed domain object detection method that works in various application scenarios, including indoor and outdoor scenes with static or moving cameras. Potential camera motion has to be detected and compensated, so global motion estimation as described in the previous section represents the first object detection step. The building blocks of the object detection algorithm are illustrated in Fig. 3.8.

The result of the GME is the vector θ, which contains the four camera motion parameters $pan, tilt, zoom, rot$. During the GME, outlier masks in sub-MB resolution are created

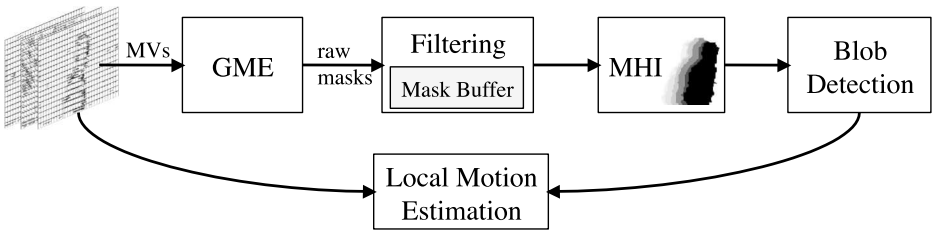


Figure 3.8: Object detection scheme

from vectors that do not follow the global motion. Outliers mainly originate from moving objects, so they represent a first, rough segmentation of the scene in background and foreground. However, outliers are also due to noise and due to MVs with random orientation and magnitude in low-textured areas, where the block-matching algorithm of MPEG-based video codecs delivers very noisy results that do not reflect the real scene motion.

In order to alleviate the impact of these unavoidable miss-detections and the resulting temporal inconsistency, spatio-temporal filtering along the MV trajectories is performed. We apply 2D+t morphological filtering, followed by a temporal median filter over a sliding temporal window the size of one GOP, which is usually set to 8-16 frames for real-time critical applications or video streaming. For temporal filtering operations, a mask buffer is created that holds the last N MV fields, where the default value for N is the GOP size. To limit the memory requirements, the buffer is created in the smallest sub-MB resolution, so one entry is stored for each sub-block covering 4x4 pixels in the original image.

The filtered outlier masks give a better segmentation of the scene in static background and moving foreground objects. Figure 3.9 and Fig. 3.10 show examples for the raw outlier masks of the GME module and the filtered masks that are used for object detection. A problem that arises from the application of two-dimensional global motion model can be observed in Fig. 3.10d. During camera operation, *near* foreground regions show stronger motion than *distant* regions, which is why near regions may be mislabeled as foreground in extreme cases. Spatio-temporal filtering removes most of the resulting false detections.

In the compressed domain, very small objects which are covered by only a small number of MBs are likely to be missed. A problematic example is for instance the pedestrian in the *parkrun* sequence shown in Fig. 3.10. In order to enhance the representation of small objects and to alleviate the impact of missed object MBs, we successively construct a motion compensated, so-called *Motion History Image* (MHI) [55] from the filtered masks. Motion history images store the foreground motion of multiple frames in one single channel image and are traditionally used for gesture recognition. The history image is constructed by overlaying the filtered mask image with the global MHI. At positions where the mask image is non-zero, the corresponding silhouette pixels in the MHI are set to the current time stamp. An

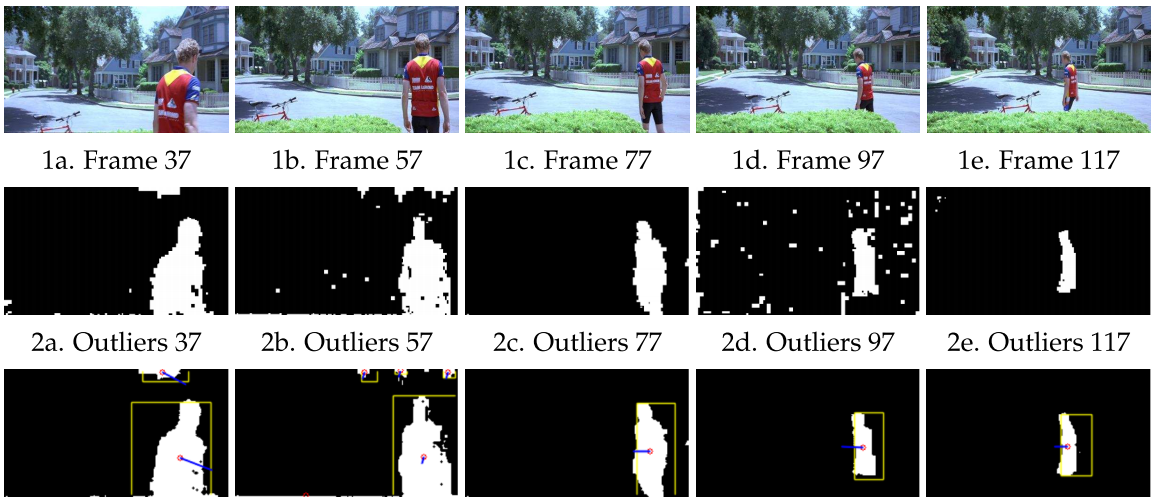


Figure 3.9: Example of raw and filtered outlier masks. 1a-e) Screenshots. 2a-e) Raw outlier masks. 3a-e) Filtered masks and detected objects. Local object motion is represented by a vector leaving the centroid, which is represented by a circle. Sequence *street with trees and bicycle ©Warner Bros. Advanced Media Services Inc.*

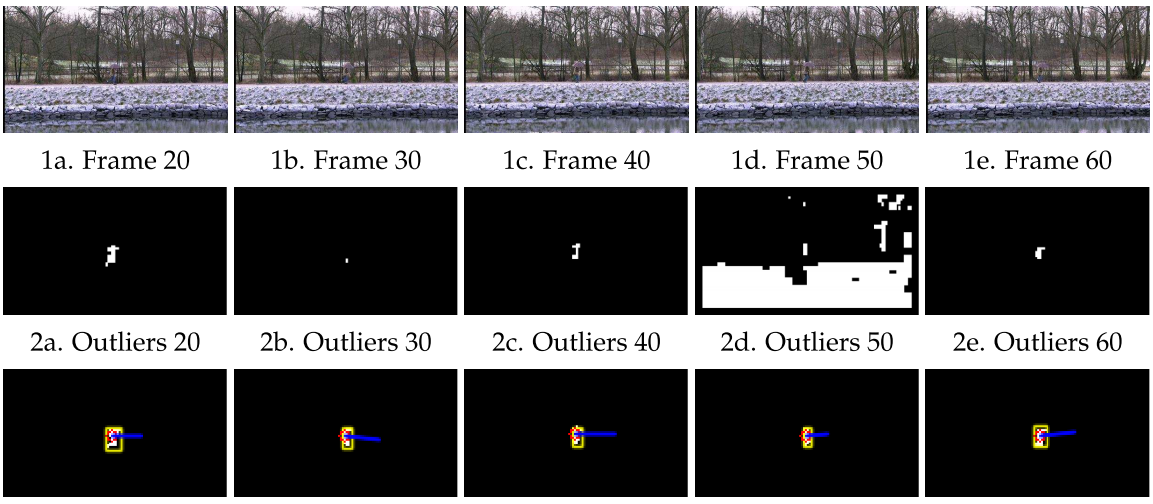
exemplary MHI is shown in Fig. 3.11. After each update step, we merge the regions with the two most recent time stamps to obtain a more solid representation of small and also of poorly detected objects. Poorly detected objects often occur in case of low-textured or very slowly moving objects, resulting in holes in the foreground masks.

For the time being, each connected region in the final foreground mask is considered as one single moving object as long as it is locally separated from others. Since no color or texture information is available, overlapping objects with merged silhouettes cannot be separated on the basis of a single frame. If the objects in question only overlap temporarily, they can be identified by analyzing the temporal evolution of the frame-wise detection results. More details on this issue are provided in Sec. 3.4 - *Object Tracking and Trajectories*.

As a final step of the detection stage, we calculate and store certain properties of all detected objects, namely the

- frame number of occurrence
- the binary object mask together with its position
- the local object motion.

The local object motion is estimated similarly to global motion (see Sec. 3.2), except that only MVs which are covered by the object mask are included in the estimation support. Furthermore, we only perform one iteration, resulting in the standard least-squares estimation of the parametric motion model. The estimated local object motion serves as the position



1a. Frame 20 1b. Frame 30 1c. Frame 40 1d. Frame 50 1e. Frame 60

2a. Outliers 20 2b. Outliers 30 2c. Outliers 40 2d. Outliers 50 2e. Outliers 60

3a. Filtered 20 3b. Filtered 30 3c. Filtered 40 3d. Filtered 50 3e. Filtered 60

Figure 3.10: Screenshots, raw and filtered outlier masks. 3a-e) detected objects. Local object motion is represented by blue vectors. Sequence *parkrun* ©*Sveriges Television AB (SVT)*

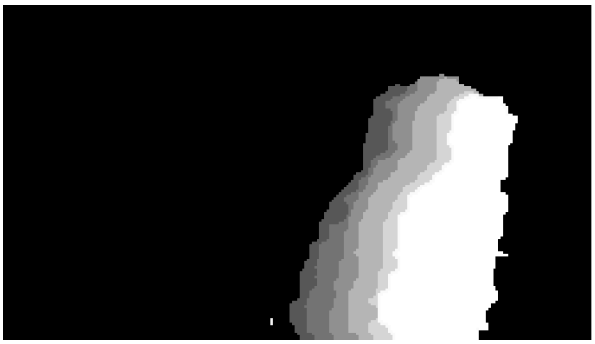
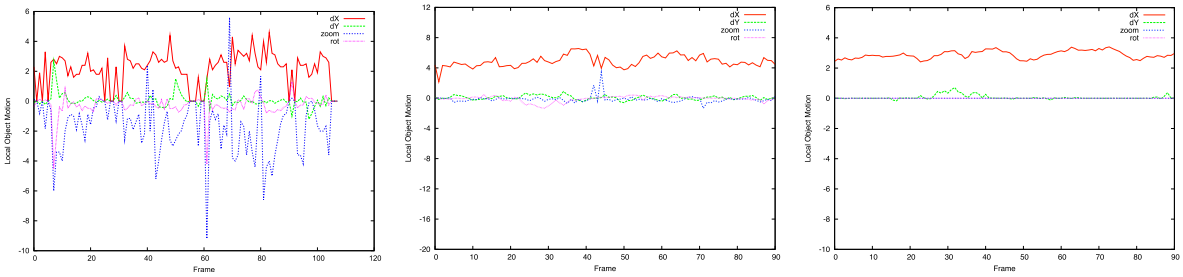


Figure 3.11: Example of MHI of sequence *street* with memory length 6 for better visibility

prediction during object tracking (see Sec. 3.4). Small objects are covered by only few MBs, so the local motion estimates are usually very noisy. In the case of spatially scalable SVC streams, and if higher spatial layers are available, we successively extract the MVs of higher spatial layers that are covered by the projected object mask in order to enhance the estimation support and thus to improve the estimation results. Figure 3.12 shows the achieved refinement of the estimated local motion of the pedestrian in sequence *parkrun*. The man is constantly walking to the right while maintaining his speed approximately constant. After an estimation refinement at doubled resolution, noise is significantly reduced and the horizontal motion dX remains more stable. For comparison, the manually obtained ground truth is shown in Fig. 3.12c. The local motion is greatly improved, resulting in more accurately predicted positions during object tracking.



(a) layer 480x272 (b) layer 960x544 (c) ground truth (480x272)

Figure 3.12: Local object motion of man in sequence *parkrun*. (a) shows the original estimation at spatial layer 480x272, (b) the estimation refinement after adding the local MVs of layer 960x544. (c) manually obtained ground truth

Sequence	Duration [frames]	Corr. detected objects	Missed	False positives
street	270	268/270 (99%)	2	22
parkrun	100	95/100 (95%)	5	3
surveillance	118	224/236 (95%)	12	3
kung fu	180	291/303 (96%)	14	0
hall monitor	300	404/455 (89%)	51	1
restaurant	310	288/310 (93%)	22	7
train	228	205/228 (90%)	23	2

Table 3.2: Object Detection Results

3.3.3 Results

Table 3.2 summarizes the results of the object detection stage for multiple test sequences. Figure 3.13 shows additional screenshots and the filtered foreground masks. The method performs well for various kinds of sequences. However, as it is solely based on the sparse and noisy motion fields, some problems are unavoidable, most notably that far away objects or objects that hardly move are often partially or completely missed, like for instance the man on the left in the well-known *hall monitor* sequence as he stops to drop his bag, and only his arm keeps moving for a couple of frames.

Furthermore, false detections may occur on unwanted moving objects like trees in the wind or occasionally in static, low-textured regions, like for example in the *street* sequence shown in Fig. 3.9.

Except for object detection issues caused by low-textured areas, the mentioned problems are common to most automatic moving object detection systems even in the image domain. The big advantage of image domain approaches is the availability of pixel-wise color information, which enables additional post-processing operations like shadow detection (e.g., [56]) or the incorporation of results from image segmentation algorithms (e.g., [57, 58]). The advantage of the presented compressed domain approach however is the support for camera

motion, no need for training to perform background construction and its simplicity, while delivering robust results under various conditions.

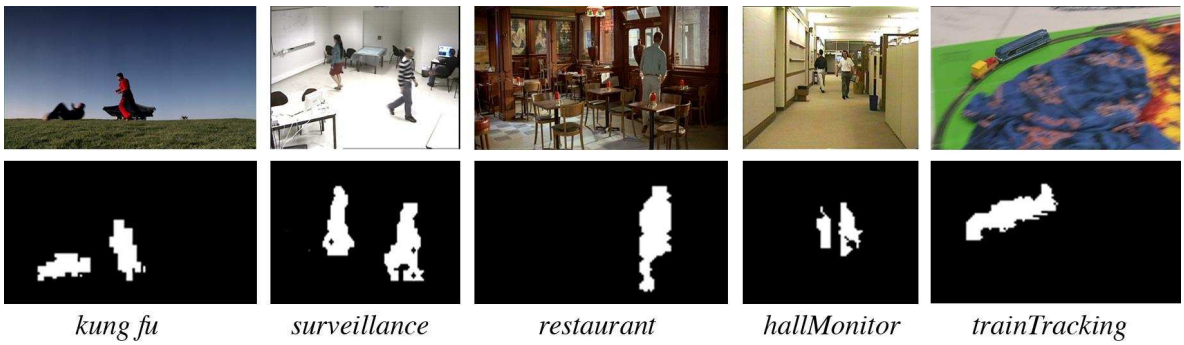


Figure 3.13: Example segmentation results

Some of the imperfections of the detection stage can be recovered by analyzing the independent, frame-wise object detection results over time. Methods for object tracking and 2D+t trajectory estimation are presented in the next section.

3.4 Object Tracking and Trajectories

The tracking of moving objects opens the doors to numerous high-level analysis tasks, ranging from person or vehicle counting to behavior analysis and anomaly detection. The major target application of tracking algorithms is video surveillance. Tracking results may also be used to assist the task of video copy detection (see Chap. 6) or to deduce information about the scene setup, such as the orientation angle of the camera (see Chap. 5 for more details on that topic).

3.4.1 Related Work

The detection and tracking of moving objects was extensively studied during the last decades. The majority of presented approaches are working in the image domain and employ a variety of different strategies, including extended Kalman filters (EKF) [59], Multi-Hypothesis Tracking (MHT) [60], particle filters [61], Mean-Shift tracking [62], Hidden Markov Models (HMMs) [63] or non-Markovian hypothesis selection [64]. Although such image domain algorithms usually deliver more robust and precise results, the necessary pixel-level grayscale and color information is not available in the compressed domain.

Similar to global motion estimation and object detection, pure compressed domain tracking approaches rely either on MVs, on residual information in the form of transform coding coefficients, or both. Most of the published methods exploit the information found in MPEG-

1/2 streams, where MVs and DCT coefficients are easily accessible. Hesseler et al. [26] perform the tracking initialization on decoded I-frames and use histograms of MPEG-2 MVs to track objects. The method does not support rotating objects and changes in size. Lie et al. [65] proposed a system that tracks single macroblocks (MBs) under consideration of residual information. The MB trajectories are afterwards merged to obtain a moving object segmentation. An MPEG-2 face detection and tracking system based on DCT coefficients was presented by Fonseca et al. in [66].

Numerous other object trackers specific to MPEG-1/2 have been proposed in the literature. However, the majority cannot be employed in the H.264 domain because some basic assumptions are no longer valid. Most notably, the often used DCT coefficients (e.g., [67, 68, 69, 70, 71, 72, 38]) are not available in H.264 streams, because intra-coded blocks in H.264 are transformed from *spatially* intra-predicted values instead of the original pixel values, so full frame decoding would be necessary to obtain meaningful values.

Regarding our goal of unsupervised object detection and tracking, all former MPEG-2 approaches that are purely based on motion could in principle be ported to the H.264 compressed domain, but have other shortcomings such as manual tracking initialization (e.g., [73]), no support for camera motion (e.g., [74]) and no support for multiple, occluding objects (e.g., [75]).

Only a few approaches specific to MPEG-4 and H.264 have been proposed in the literature. Sutter et al. [76] presented a lightweight tracking algorithm for MPEG-4/Visual. It is based on the motion vector field and works by the frame-wise translation of a pre-selected region of interest (ROI). No indication for the performance in the case of multiple occluding objects is given and the system has to be initialized by the user. You et al. [77] proposed a H.264/AVC based method that performs tracking of feature points selected by the user. The matching of these points uses the dissimilarity energies related to texture, form, and motion. Therefore, they partially decode the stream around the Region-of-Interest (ROI) back to pixel level and fully decode I-frames. Mehmood et al. [78] propose a light-weight tracking algorithm in the H.264 domain. Although the method can cope with camera motion, it has to be initialized manually and only supports single object tracking.

Rath et al. proposed an MPEG-2 approach in [79] which is solely based on motion vectors and aims at unsupervised segmentation and tracking of moving objects in real-time. The initial object segmentation and detection is based on an iterative rejection scheme that is incorporated in a global motion estimation process with iterative rejection [80]. For each detected object, the binary object mask is considered as a model and is updated at each occurrence. Tracking is performed by measuring the distance between MVs covered by the mask in two successive frames. Although the results for various test sequences are promising, the system cannot handle multiple objects or problems such as occlusions.

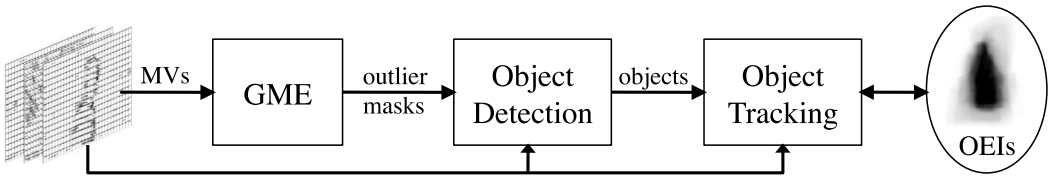


Figure 3.14: Block diagram of the motion based object tracker

In the next section, our approach to unsupervised object detection and tracking in the H.264 domain is presented. While the initial, frame-wise object segmentation step is similar to the one in [79], we propose a richer object model and use a modified object matching algorithm, enabling the detection and tracking of multiple, temporarily occluding objects.

3.4.2 Algorithm

The tracking algorithm is purely motion based in order to avoid full stream decoding. It builds upon the results of GME and the object detection stage presented in the previous sections 3.2 and 3.3, respectively. A block diagram of the object tracking processing chain is shown in Fig. 3.14.

Output of the object detection stage are frame-wise detected object together with their properties, namely the binary mask, the position, the mask moments up to the second order and the local motion estimates. Additional object properties like the object size and centroid are calculated from the mask moments to assist the matching process.

Denoting the intensity values of the binary mask image by $I(x, y)$, the calculation of the raw mask moments is given by

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y). \tag{3.12}$$

The object size M_{00}, measured as the visible surface projected onto the image plane, is assumed to equal the number of non-zero elements covered by the binary mask. The coordinates of the centroid $(\bar{x}, \bar{y})$ are determined by

$$\bar{x} = \frac{M_{10}}{M_{00}} \quad , \quad \bar{y} = \frac{M_{01}}{M_{00}}. \tag{3.13}$$

In the following, we explain how similar objects in adjacent frames are identified and how the object reference point, which is taken as reference for the object's current position, is calculated.

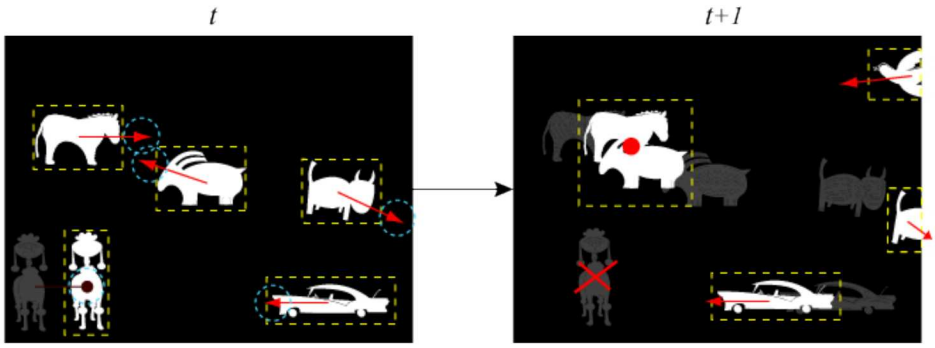


Figure 3.15: Some possible matching situations that may occur

Object Correspondence

Figure 3.15 illustrates some possible situations that may occur in two successive mask images. New objects can appear at the image borders or within the scene because they start moving, and already present objects may disappear because they stopped moving or left the scene. Common problems with respect to multiple objects are occlusions, which result in merged object masks that may split up again later in time.

Since no color information is available to resolve problems like occlusions, a temporal analysis of the object properties is carried out, which allows to draw certain general conclusions about what is happening in the scene. The following set of heuristics is used. It is kept very general, since we have no a priori knowledge of the scene or the appearing objects.

- **Mask position:** The mask represents the object silhouette. If it touches the image borders, a look at its local motion can tell if the object is leaving or entering the scene.
- **Mask size:** Assuming perfectly segmented objects, continuous and gradual changes in size are usually caused by objects leaving or entering the scene, by changes of the visible object surface, by occlusions with other static or moving objects, changes in perspective, zooming, changing distance to the camera, or by a non-rigid object that partially stops or resumes moving. Rapid and abrupt changes in the object size generally indicate split-and-merge situations.
- **Local motion:** The translational motion parameters *pan* and *tilt* indicate the moving direction and predict the position in the next frame (relative to the camera position). If the estimation support, i.e., the visible object surface, is sufficiently large, *zoom* may give an indication if the object approaches or moves away from the camera.
- **Centroid position:** Rapid jumps of the objects centroid position also indicate split-and-merge situations, while displacements that approximately correspond to the estimated local motion are a sign for normal, undisturbed motion.

The tracking initialization takes place when the first objects are detected at time t_i. Each object is assigned with a unique label and is kept in memory along with its properties. The expected position $(\hat{x}_{i+1}, \hat{y}_{i+1})^T$ in the subsequent frame is predicted by using the estimated, translational local motion parameters $(\hat{dx}_i, \hat{dy}_i)^T$.

$$\begin{pmatrix} \hat{x}_{i+1} \\ \hat{y}_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} \hat{dx}_i \\ \hat{dy}_i \end{pmatrix} \tag{3.14}$$

At time t_{i+1}, the algorithm searches within a threshold radius of $r_{max} = 2\|(\hat{dx}_i, \hat{dy}_i)^T\|$ MBs around the predicted position for new input objects.

In case only a single object is detected, the matching process is straight forward. If a new object is detected within the search area in the successive frame, we assign the same label to it. Otherwise we assume the object has stopped moving, mark it as *inactive* and store it together with its "last seen" position and frame number. If at a given moment, a new object appears "out of nowhere", i.e., one that is neither entering nor leaving the scene, we search for previously stored *inactive* objects in that region to reactivate them. If none is found, we assign a new label to the appearing object.

Merging. If an object of significantly larger size is found in the search area, we check if that new object coincides with the predicted position of another object. If this is true, the objects masks probably merged and we assign both labels to that joint object. Otherwise we check for inactive objects that have been lastly detected at the respective position. If no other near objects have been detected prior to an abrupt change in size, we however copy the same label to the new object and set a flag of uncertainty. Possible explanations include quickly moving objects that re-appear behind occluding obstacles, quick changes in the view perspective of non-symmetric objects, or merging with a previously static object, for instance a pedestrian that picks up baggage or a bike.

Split situations where multiple objects emerge out of one are discussed at the end of the next section 3.4.2.

Reference Point

At this stage, we have identified similar objects over time. Since moving objects are mostly non-rigid and often occluded by obstacles, we look for a reference point within the object that remains as stable as possible. We therefore chose the center of gravity. To give an example, the waiter in the sequence shown in Fig. 3.16a moves from one table to the next, stops to clean them and is often partially occluded. While wiping the table, the centroid of the mask moves away from the original one, which was located around the waistline.

(a) (b)

Figure 3.16: (a) Exemplary deformations of the same object at different moments in time. The red circle is the centroid of the overlaid, white object mask. (b) Final OEI of waiter. Sequence *restaurant ©Warner Bros. Advanced Media Services Inc.*

In order to stabilize the reference point over time and to resolve merged masks, we propose the construction of so-called *Object Energy Images* (OEI). We extend the idea from motion energy images (MEIs) [81] to object silhouette construction. The goal is to create a more stable representation and model of an object than the quickly fluctuating object mask. With OEI, we are able to obtain a measure of inertness.

At the first occurrence of an object, we initialize the OEI with the original mask. Each time a previously present object is detected in the current frame, we project the OEI to the position predicted by *pan* and *tilt*. We superimpose it with the new mask image and increment the value of the OEI at positions where mask values are non-zero. If the new mask does not entirely fit into the projected OEI, we enlarge it accordingly.

We keep one long-term OEI as a model for each detected object and continuously update it as long as the object is detected. The OEI represents a silhouette image of the object, where the most rigid regions appear brighter than parts like legs or arms. As the reference point we compute the center of gravity $\mathcal{R}$, defined as the average of all OEI positions r_i weighted by their pixel value w_i:

$$\mathcal{R} = \frac{\sum r_i \, w_i}{\sum w_i}. \tag{3.15}$$

Since more importance is assigned to higher values, darker zones in the OEI like moving arms have a lesser influence on the position of the reference point. Examples of OEIs are given in Fig. 3.17 and 3.16b.

The most problematic cases regarding the objects' reference points are merged object masks. In this situation, as the merging is detected, we project the OEIs of the involved objects at their predicted positions. The involved OEIs are now superposed with the merged,

(a) *street* (260) (b) *surv. 1* (42) (c) *surv. 2* (42) (d) *car* (223)

Figure 3.17: Final OEIs for different objects. Number in brackets shows the number of individual object masks used for constructing the OEI

binary mask. Per involved object, we now re-estimate the local motion using the MVs covered by its OEI, where the normalized OEI acts as a weighting function. The weighting accounts more importance to inert parts of the object that best represent the local object motion. As long as the merging lasts, we do not modify the OEI and shift it each frame to the position predicted by the new local motion estimate.

If the real objects overlap in the scene, the OEIs will also overlap, which leads to false prediction results for at least one of the objects involved, because MVs from the wrong object are used to estimate local motion. At an OEI overlap of more than 50%, we judge which of the objects is in the foreground by analyzing the continuity of the moving direction in terms of degrees of all involved objects. At each time step, we sum up the angle difference between the last and the currently estimated moving direction for each object. The object with the smallest sum of differences is considered as foreground and for this object, we continue to track it as described above. For background objects, we keep the previously estimated moving direction constant to predict its new position until the OEI overlap falls again under 50%.

In the case more than two objects merge, the labels have to be re-distributed correctly when the objects split up again. A decision has to be made which of the newly separated objects corresponds to which original object. In the two-object case, this is handled by the determination and tracking of the foreground object like described above. For more objects, we assume that the moving direction of all involved objects is hardly affected by the occlusion and we re-assign the labels according to the closest matches when comparing the difference in moving direction angles before and after the split. However, this assumption may lead to a false label switching if objects change their moving direction during the occlusion.

If the system gets initialized with merged objects that split later on, we only know after the split that the area contained multiple objects. We then reset the merged OEI and re-initialize a new OEI for each object. Trajectory results for sequences with multiple tem-

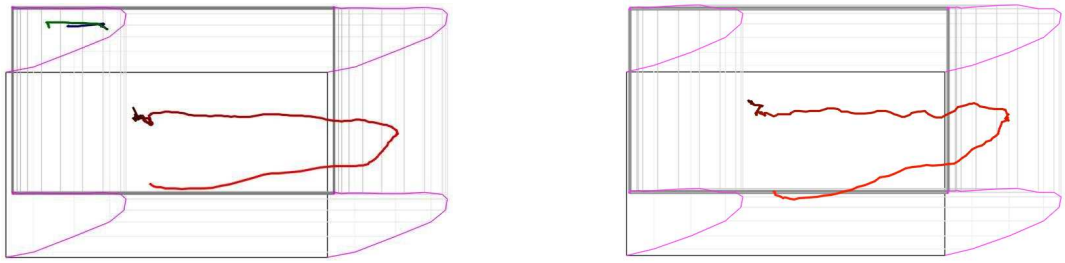
porarily occluding objects are given in Fig. 3.19c-d (see next section for further explanation of the trajectory plots).

3.4.3 Results

At this point, we can draw the trajectories in the image plane as seen by the camera. Example results are provided in Fig. 3.18 and 3.19. The complete, global motion compensated trajectory for the man in the *street* sequence is shown as a red curve in Fig. 3.18a. For comparison, Figure 3.18b shows the ground truth, obtained manually by a user who was demanded to click on the middle of the mans' waistline in each frame.

The grey rectangles in the plots represent the viewport of the camera over time. One rectangle per second is drawn. The purple curves which connect the rectangle corners represent the estimated camera translation over time. It can be observed that the camera is following the moving object. It has to be noticed that the third dimension in this representation is time and not space. The estimated trajectory deviates only slightly from the ground truth and the centroid motion appears smoother in the estimated trajectories due to the stabilization through object energy images.

The short green and blue lines in the top left corner of image 3.18a correspond to tree branches moving in the wind.



(a) (b)

Figure 3.18: a) Estimated trajectory b) Manually obtained trajectory. The rectangles represent the viewport of the camera over time (1 rectangle per second is drawn). The brightness of the trajectory decreases with time. Sequence *street*.

Since our initial information is block-based motion, very small, unclear, or slow objects are difficult to detect and follow. For example, the small and motion blurred toy helicopter in sequence *trainHelicopter* (Fig. 3.19h, marked by a circle) is only detected notably after its take-off of the floor, and the slowly approaching pedestrian in sequence 3.19g is not detected at all. However, all normal sized objects are well detected and tracked, even the rather small pedestrian in the center of sequence 3.19b.

Up to now, the presented trajectories are in the dimension 2D+t. In the following section, we present an extension to the pseudo 3D case by estimating the relative distance of moving objects to the camera.

3.5 Distance Estimation

In this section, a single-view method is proposed to estimate the relative distance of moving objects to the camera. It builds on the results of the object tracker presented in the previous section and is based uniquely on compressed domain motion information, like all other algorithms in the proposed processing chain. We assume that moving objects have been successfully detected and tracked throughout the scene.

Since no a priori knowledge of the camera parameters, the scene geometry or the objects in the scene is given, the estimated positions of moving objects are always relative to the camera and no absolute distance values can be obtained. Nevertheless, a relative distance or depth measure can be introduced, which reflects if moving objects are approaching or moving away from the camera.

3.5.1 Related Work

The estimation of the distance of moving objects to the camera is similar to the problem of estimating 2D ground-plane or 3D trajectories. A large number of approaches has been proposed, most of them being tailored to a specific application environment. The existing work can be coarsely divided into single-view and multi-view approaches.

Multi-view algorithms for estimating the location of moving objects are most likely to deliver robust and precise results. Occlusions probably do not affect all views to the same extent and faulty measurements in one view may be corrected with the help of others. Turolla et al. [82] presented a multi-camera system for location estimation. It borrows the Joint Directorate of Laboratories (JDL) model from the data fusion domain in order to segment and track multiple occluding objects by using different features such as color, position and dynamics. Park et al. [83] use a parallel projection model which supports both zooming and panning of the imaging devices. The proposed algorithm is based on a virtual viewable plane for creating a relationship between an object position and a reference coordinate, which reflects the distance between object and camera. The reference point is initially a rough estimate which is then iteratively refined.

In monocular or single-view sequences, some kind of object distance measure has to be introduced and estimated. Most proposed algorithms operate in the pixel domain and are designed for specific applications in mind. Rosales et al. [84] apply extended Kalman

a. Sequence *Stefan*

b. Sequence *Parkrun* ©*Sveriges Television AB (SVT)*

c. Sequence *Surveillance* ©*LaBRI*

d. Sequence *Kung-Fu* ©*Sveriges Television AB (SVT)*

e. Sequence *Hall Monitor*

f. Sequence *Man in Restaurant* ©*Warner Bros. Adv. Media Services Inc.*

g. Sequence *Car* from *PETS00*

h. Sequence *TrainHelicopter* ©*LaBRI*

Figure 3.19: Image plane trajectory estimation results for some test sequences. From left to right: 1-3) Screenshots 4) Estimated trajectory 5) Manually obtained trajectory. The drawn rectangles represent the viewport of the camera over time (1 rectangle per second is drawn). The color of the trajectories becomes darker over time.

filtering to reconstruct relative 3D trajectories. Ono et al. [85] and Gil et al. [86] use defocus information to determine the distance. Rao et al. [87] propose the object size as a depth indicator in a system with an aircraft mounted camera. In [88], Qi et al. propose a method of object localization based on a model of the human height. Using the relationship between the projected position of the head of an object and the distance between object and camera, a man height model is constructed. By this model, the two-dimensional position of an object on the ground-plane is estimated. The approach of Pang et al. [89] also uses a priori information about the object being observed to infer its distance from the camera.

The mentioned methods rely on pixel domain information and can thus not be ported to the compressed domain. In contrast to other model-based 3D tracking approaches tailored to specific types of objects [90, 91], our goal is the development of a more general approach that can handle rigid and non-rigid objects of any kind. Although a general approach that is based on a sparse and noisy approximation of the optical flow is very likely to deliver only rough estimation results, it will be shown that the obtained relative distance measure approximates the evolution of the object's depth well.

In the following section, we propose a single-view compressed domain method to infer the object's relative distance to the camera. No a priori knowledge of the scene is assumed.

3.5.2 Algorithm

Since we work with single, uncalibrated cameras and compressed domain information, the distance of the moving object to the camera can only be observed indirectly through its position and size. Furthermore, no absolute distance values can be obtained due to the lack of scene geometry knowledge and camera parameters. We aim at estimating a relative distance measure that reflects if an object is approaching or moving away from the camera. We consider the following object properties as distance indicators:

- visible object area (mask size or area $A(t)$),
- object width ($w(t)$),
- object height ($h(t)$).

The object area, width and height are determined with the help of the second order central moments of the binary mask image. Using Eq. 3.12, the area A is given as M_{00}. In order to account for rotating objects like turning cars in surveillance sequences, the width and height are measured along the orientation Θ of the mask image, which can be calculated by

$$\Theta = \frac{1}{2}\arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right), \tag{3.16}$$

with

$$\begin{aligned} \mu'_{20} &= \mu_{20}/\mu_{00} &&= M_{20}/M_{00} - \bar{x}^2 \\ \mu'_{02} &= \mu_{02}/\mu_{00} &&= M_{02}/M_{00} - \bar{y}^2 \\ \mu'_{11} &= \mu_{11}/\mu_{00} &&= M_{11}/M_{00} - \bar{x}\bar{y}. \end{aligned} \tag{3.17}$$

The dimensions along the mask orientation and its perpendicular axis are considered as height and width, respectively.

We assume that the relation between the visible object surface and its relative distance $d_{rel}(t)$ is of quadratic nature. Due to occlusions, noise, perspective distortions and non-rigid objects, this assumption is violated in nearly all real world scenarios. Nevertheless, the application of a very strong temporal moving average filter (window ≥ 2 sec.) mitigates the mentioned effects in most cases and leads to a more confident measure $A_f(t)$ of the object area. It relates to the distance as $A_f \sim d_{rel}^2$. We also filter the object height and width in a similar way, which are related linearly to d_{rel} as $h_f(t) \sim d_{rel}$ and $w_f(t) \sim d_{rel}$.

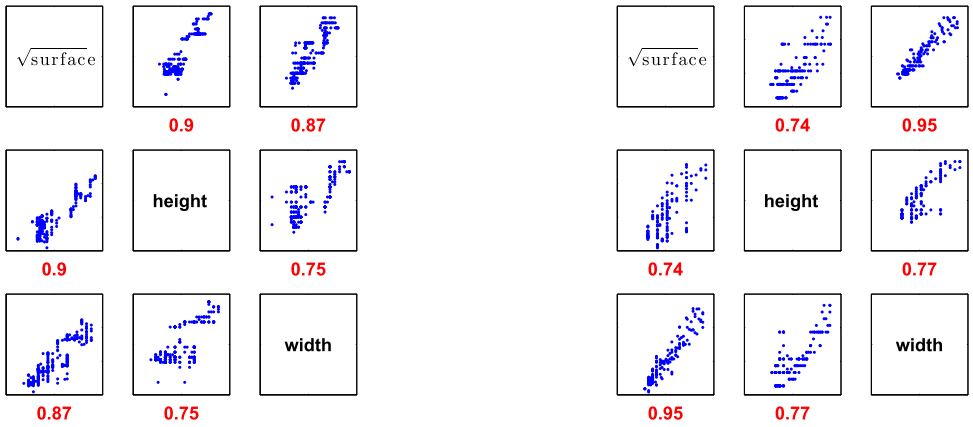
(a) *street* (b) *car*

Figure 3.20: Mosaics of two example test sequences

Matrix correlation plots of the considered indices are given in Fig. 3.21 for the two example sequences *street* and *car*, depicted in Fig. 3.20. The matrix plots show the distance indicators plotted against each other and confirm the assumed linear relationship between the visible surface, the object width and its height. Figure 3.21a refers to the man in sequence *street* and Fig. 3.21b to the vehicle in sequence *car*. The rigid nature of the vehicle results in slightly cleaner plots.

The three normalized indicator vectors are stacked into the matrix

$$\mathbf{I} = [A_f(t)^T h_f(t)^T w_f(t)^T], \tag{3.18}$$



(a) *street* (b) *car*

Figure 3.21: Matrix correlation plots. Numbers below are the covariance values.

which is multiplied by the weight vector $\mathbf{w} = [w_A w_h w_w]^T$. The weight w_i for each variable is chosen proportional to its maximal correlation coefficient r with any of the other two variable:

$$w_i \sim \max(|r(i,j)|); i \neq j; i,j \in \{A,h,w\}; \sum_{n=1..3} w_n = 1. \tag{3.19}$$

This way, more influence is assigned to the variables that correlate better. The weight vectors for *street* and *car* are thus $\mathbf{w} = [0.34\ 0.34\ 0.32]^T$ and $\mathbf{w} = [0.36\ 0.28\ 0.36]^T$, respectively.

The final distance estimation result is then obtained by

$$d_{rel}(t) = d_0 * \mathbf{I} \cdot \mathbf{w}, \tag{3.20}$$

where d_0 is an arbitrary reference distance at t_0, which was set to 100. Values smaller than d_0 reflect that objects are moving away from the camera, whereas for approaching objects the relative distance d_{rel} becomes greater than d_0.

Examples for the temporal evolution of the three distance indicator and the final depth estimation results for sequences *street* and *car* are shown in Fig. 3.22. The lower plots show the error in the measured variables compared to the ground truth in percent. The ground truth has been obtained manually by measuring the object height, width and surface in the video sequence and was further processed in the same fashion as the estimated variables. The error on all variables is moderate with $\pm 7\%$. The mean error in the relative distance for sequence *car* is about 1 % smaller than for sequence *street*, because the object is rigid and easier to detect.

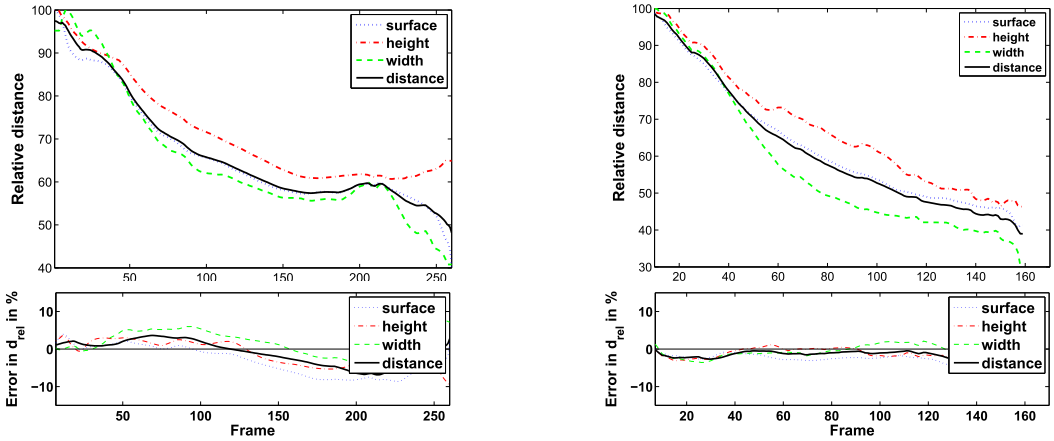
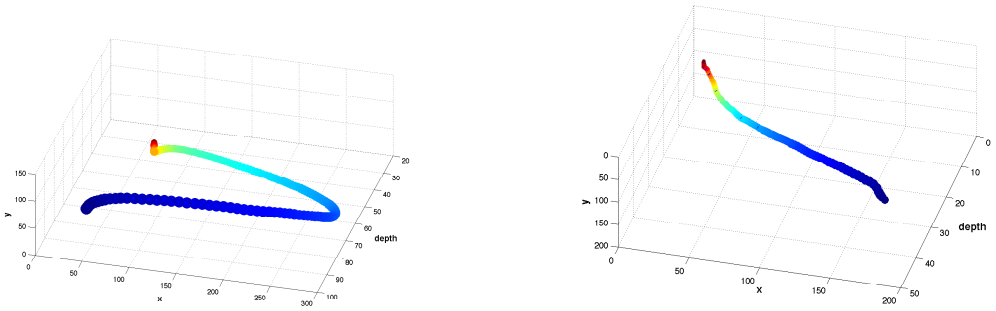


Figure 3.22: Top row: Normalized distance indicators and final, relative distance estimation result. Bottom: Error between ground truth and extraction results.

3.5.3 Results

Unfortunately, no ground truth for the real object distance is available for our test sequences, so the accuracy of the estimation results has to be evaluated by human judgement. Since plots of the estimated relative distance over time alone (see Fig. 3.22) are not very demonstrative, a more intuitive and illustrative representation is desirable. This can be achieved by weighting the formerly obtained, global motion compensated 2D trajectories with the estimated relative depth d_{rel}. This leads to a pseudo 3D representation of the object trajectories in a coordinate system spanned by the image width x, the image height y and the depth d_{rel}. These pseudo 3D trajectories for the sequences *street* and *car* are shown in Fig. 3.23. When compared to the mosaic images shown in Fig. 3.20, it can be observed that – despite the rough input information – the relative distance to the camera is well approximated for both sequences. Assuming a flat surface, 2D ground plane trajectories can be obtained by projecting the 3D trajectories on the plane spanned by x and d_{rel}.

The pseudo 3D trajectories for other sequences are provided in Fig. 3.24 (for screenshots and 2D+t trajectories see Fig. 3.19). Although the overall results reflect the real motion very well, two problems come apparent regarding Fig. 3.24e and 3.24f. Figure 3.24e shows the pseudo 3-D trajectories of the two men in the well-known sequence *hall monitor*. The man on the left enters the scene through a door in the hallway, so his surface, width and height steadily increase until full visibility. Since his silhouette does not touch the image border, he is not considered as an object that is newly entering the scene. The steadily increasing mask size leads to the false conclusion that he is approaching the camera during the first frames of appearance. A similar problem can be observed regarding Fig. 3.24f, depicting the trajectory



(street) (car)

Figure 3.23: Trajectory with relative object distance (depth) to the camera. Colors: blue $\hat{=} t_0$, red $\hat{=} t_{end}$

of the waiter in the *man in restaurant* sequence. At one moment he stops to wipe a table and only his arm keeps moving, which is erroneously interpreted as distancing.

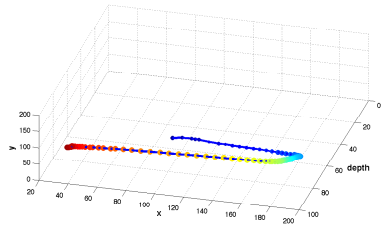
In both of the mentioned cases, a very sharp peak can be observed in the trajectories, whereas *normal* motion is usually more fluid. For future work, peak detection on the 3D trajectories can be performed to detect and correct such situations. If a specific camera view or application scenario is given and if a sufficient amount of data is available, another way of handling similar problems is the use of machine learning techniques to perform trajectory or behavior classification, like for instance in [92, 93, 94].

3.6 Summary and Conclusions

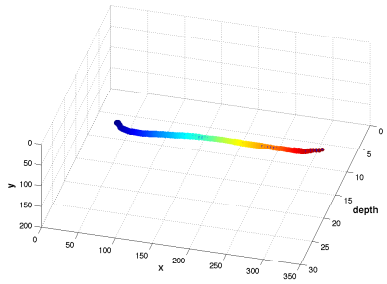
We presented efficient methods to index and analyze scalable or single layer video streams encoded by H.264/AVC or SVC. The algorithms rely entirely on compressed domain motion data, because block-based transform coding coefficients, which are often used in the processing of MPEG-2 streams, are not available in the H.264 domain due to more advanced intra-prediction schemes. All presented methods require no training, can handle camera motion and multiple objects, are fully unsupervised and allow computationally efficient analysis.

The first block in the proposed processing chain is the extraction of macroblock modes, partition sizes and motion vectors. A modified version of the publicly available reference decoder JSVM is used for stream parsing and entropy decoding.

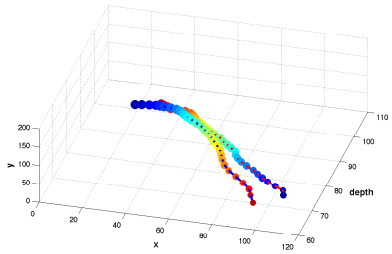
We use the extracted motion vectors to robustly estimate the global scene motion, which in most cases originates from camera operations like panning or tilting. This step has to be performed if the target application is not limited to sequences with static cameras. Furthermore, the robust estimation scheme with iterative rejection delivers valuable outlier masks as a by-product, which represent a rough segmentation of the scene in background and mov-



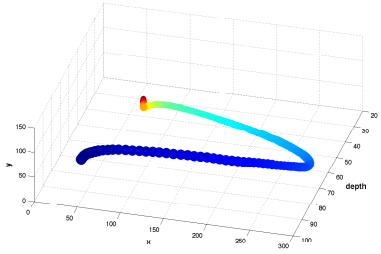
a. *Stefan*



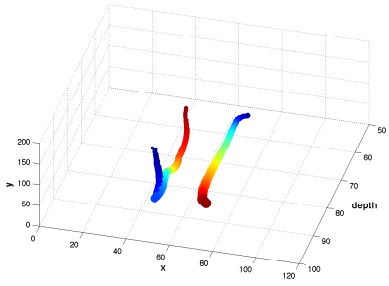
b. *Parkrun*



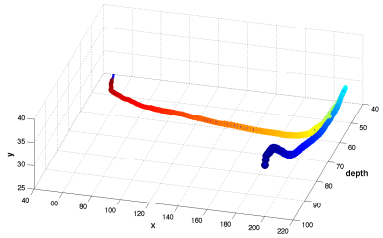
c. *Surveillance*



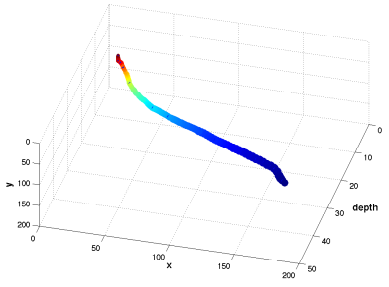
d. *Street*



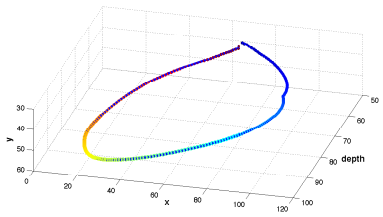
e. *HallMonitor*



f. *Restaurant*



g. *Car*



h. *TrainHelicopter*

Figure 3.24: Pseudo 3D trajectories for some test sequences. Depth is relative distance of the object to the camera. Colors: blue $\hat{=} t_0$, red $\hat{=} t_{end}$. For screenshots, see Fig. 3.19

ing foreground objects. Due to its proven robustness, we adopted an existing global motion estimation algorithm and modified it to handle single-layer as well as scalable H.264 streams.

Based on the outlier masks from the global motion estimation and the motion vectors from foreground regions, we developed novel methods to detect and track moving objects in the image plane seen by the camera. The most difficult tracking situations arise from multiple occluding objects, resulting in merged silhouettes in the binary mask image. In a general approach that is not trained on a specific type of object and where non-rigid objects may appear, the masks cannot be separated by looking at a single instant in time. We propose a temporal scene analysis and the construction of object energy images as dynamic models to resolve merged mask situations. The image plane tracking results are illustrated by 2D+t representations, which show the object trajectories together with the camera motion over time for a given video sequence.

From the temporal evolution of the object silhouette dimensions, we presented a way to infer a rough measure representing the relative distance of the object to the camera. The estimation of a depth measure allows to construct pseudo 3D representations of the object trajectories or via projection, the construction of 2D ground plane trajectories. Real-time video surveillance, scene summarization and copy detection are some of the possible applications that can benefit from the presented methods. The developed approach lead to the publication presented in [95].

The methods that have been presented in the first part of this work represent a set of general tools that can be employed in different application scenarios. Part II of this work focuses on how different types of applications can benefit from compressed domain processing. Video surveillance represents a major market for computer vision systems. Chapter 4 presents a framework that shows how the benefits of image domain processing can be combined with the efficiency of compressed domain methods in the context of a traffic surveillance application. In Chap. 5, a novel compressed domain method to roughly estimate the camera orientation through object motion in single view sequences is presented. Chapter 6 presents a critical evaluation of compressed domain features in a video copy detection framework.

Although the presented algorithms proved to deliver robust results for a variety of videos, different problems still need to be solved. Like also in this work, the segmentation of the scene in background and foreground objects is carried out via global motion estimation in most systems that face sequences with potential camera motion. As already stated in Sec. 3.2.4, global motion estimation fails under certain circumstances due to the noisy nature of MPEG motion vector fields. This affects the whole processing chain that builds upon on robust global motion estimates. In Chap. 7, we present an approach to mitigate these effects by performing joint indexing and coding on the encoder side in order to improve the quality of the motion vector field in terms of compressed domain analysis.

We end the first part of this thesis with a concluding remark. The trade-off between the computational efficiency of compressed domain approaches and the precision of pixel domain processing has to be evaluated during the design of a vision system. If multiple video streams are to be analyzed in parallel and real-time capabilities are a must, compressed domain processing offers great possibilities.

In the upcoming chapters, we demonstrate the capabilities of compressed domain processing in the context of different application scenarios, e.g., video copy detection and traffic surveillance. The limitations of pure compressed domain approaches are pointed out and we show how results can be improved by combining both image and compressed domain processing in an efficient way.

Part II

Applications

Introduction

While a more general introduction to compressed domain analysis was provided in Part I, the second part of this work – Applications – focuses on possible real-world applications of the compressed domain tools that have been presented in Chap. 3. The goal is to demonstrate by example how different types of applications can benefit from the proposed methods and from compressed domain processing in general. Depending on the target application, we present additional techniques that are tailored to specific domains. Furthermore, the limitations of pure compressed domain analysis are pointed out and combinations with traditional image domain analysis are proposed where otherwise no satisfying results are obtained.

Video surveillance is arguably the most important and obvious application of computer vision algorithms. The enormous amount of visual information that is generated by the thousands of installed surveillance cameras can not be handled by human observation alone. If real-time analysis and on-time alert triggering in case of anomalies is envisaged, a successful surveillance system heavily depends on the presence of robust and semi- or completely unsupervised detection methods. In Chap. 4, we present an approach that combines the advantages of image domain and compressed domain analysis in the context of traffic surveillance on highways.

In Part I – Sec. 3.5 we described a purely compressed domain method to estimate the relative distance of moving objects from the camera. Building on top of these results, we present an approach to roughly classify the camera orientation angle in Chap. 5. As demonstrated by Hoiem in [96] and [97], scene understanding and object detection/recognition results can be improved by taking information about the camera orientation into account.

The detection of copies in video data bases and archives is another interesting application that is suitable for compressed domain analysis. When dealing with large data sets, efficient feature extraction algorithms and fast queries are a must. Since most video archives are closed or at least centrally controlled systems, usually all of the videos are encoded by the same codec and in the same manner. This permits to deploy efficient compressed domain indexing or retrieval solutions that are tailored to the specific coding standard in use. In Chap. 6, a video copy detection system for H.264 coded video data bases is presented.

As stated before, the object segmentation and detection on H.264 coded video is solely based on the sparse and noisy motion vector fields of predictive pictures. Although this approach allows for efficient processing and delivers good results in a variety of scenarios, it fails under certain conditions because of arbitrary motion fields that occur due to low-textured areas or non-static background. In Chap. 7 we present an approach to partly overcome this problem by correcting the motion vector fields on the encoder-side.

Chapter 4

Traffic surveillance

Video surveillance in public places and on highways is becoming increasingly more popular and important. In the London area alone, several thousand surveillance cameras have been installed over the years. The huge number of deployed cameras necessitates robust, efficient and automated video analysis algorithms, since the enormous amount of generated data can not be handled by human observers alone.

Along with this flood of IP-based surveillance cameras, another issue that arises is the network performance. Video streams with acceptable quality, resolution and frame rate can easily occupy a considerable amount of bandwidth. The ongoing development and deployment of high resolution cameras requires the use of efficient source coding. A video codec that is suitable in such scenarios is *Scalable Video Coding* (SVC) [2], the scalable extension to H.264/AVC, which received final approval in 2007. The first commercial security cameras and systems that support H.264/SVC are already available by vendors like Aventura Technologies[1] or General Electric's intelligent video platform VisioWave[2]. For more details on SVC and H.264 in general, the reader is referred to Sec. 2.3.2 et seq.

Since traffic surveillance scenes can get very crowded at peak times, compressed domain analysis that relies only on motion vectors, like presented in Chap. 3, will deliver poor results with respect to object detection and tracking. In scenes with heavy traffic, considering motion only is often not sufficient because of the likelihood of overlapping object masks. Although image domain analysis suffers from similar problems, the obtained object masks are more precise and can be further refined by techniques like shadow suppression. Additionally, methods that consider local feature points can be employed to increase robustness. However, the computational complexity of robust image domain algorithms often not allows for real-time processing.

[1] http://www.aventuratechnologies.com/
[2] http://www.gesecurity.com/

In order to enable fast and efficient analysis, we propose a method that combines the advantages of both types of analysis. In order to minimize stream decoding time, we only decode I-frames to perform image domain object detection and feature point extraction. Between two I-frames, we solely rely on B- and P-frame motion information in the form of macroblock (MB) based motion vectors (MVs) to detect and track objects. The approach can be regarded as a combination of tracking-by-detection using local feature points, and recursive tracking based on the motion field.

In order to assist object detection and tracking, we initialize the system by a short, two-stage training period. We start by fully decoding I-frames during a training period and estimate the scene background using a mixture of Gaussians (Sec. 4.4). The B- and P-frame MVs during this training time are used to construct maps that represent the dominant moving direction per MB (Sec. 4.5). This information is later used to assist the object detection (Sec. 4.7) and tracking algorithms, which combine local, scale-invariant feature points with compressed domain object matching (Sec. 4.8). The results of automatic vehicle counting and trajectory estimation are provided in Sec. 4.9.

4.1 Related Work

A large number of compressed domain object detection and tracking approaches can be found in the literature. Some publications specific to the MPEG-2 coded videos include [49, 50, 45, 46]. Babu et al. [49, 50] proposed an accumulation of motion vectors over time, followed by K-Means clustering to determine the number of objects in the scene and subsequently using the EM algorithm for object segmentation. Mezaris et al. [45] use an iterative rejection scheme based on the bilinear motion model for foreground/background segmentation. Yu [46] uses a combination of MV clustering and background subtraction of DC images to segment moving objects. Sutter et al. [76] presented a lightweight tracking algorithm for MPEG-4/FGS, but no indication for the performance in the case of multiple occluding objects is given. Furthermore, the presented system has to be initialized by the user.

A few approaches specific to H.264 have been proposed. Zeng et al. [51] employ a block-based Markov Random Field (MRF) model to segment moving objects from the sparse MV field. You et al. [77] perform tracking of feature points the must be selected by the user. The matching of these points uses the dissimilarity energies related to texture, form, and motion. Therefore, they partially decode the stream around the Region-of-Interest (ROI) back to pixel level and fully decode I-frames.

Concerning our goal of unsupervised, compressed domain scene analysis, shortcomings of existing approaches include manual tracking initialization (e.g., [78, 73, 72]), no support for camera motion (e.g., [74, 72]) and no support for multiple, occluding objects (e.g., [51,

77, 78, 72, 75]). In the context of road traffic analysis, manual tracking initialization for each passing vehicle is not feasible and the support for multiple, potentially occluding objects is a must. The fact that all existing pure compressed domain approaches only support one or very few moving objects at the same time provides further evidence that compressed domain information alone is not sufficiently rich to perform object detection and tracking in very crowded scenes, which is typically the case in road traffic applications. We therefore combine image domain with compressed domain data to enhance the results at a moderate increase in the computational complexity.

You et al. [98] proposed an approach to pedestrian tracking in H.264 streams, combining partially decoded I-frames with compressed domain information. They apply probabilistic, spatio-temporal MB filtering and partial stream decoding to detect and track multiple objects in real time in sequences with stationary background. Occlusions are handled in the image domain by constructing hue color histograms on the partially decoded I-frames. In H.264, actual partial decoding of arbitrary regions is not possible due to spatial prediction and differential coding mechanisms. The authors propose to a replace the MBs around detected objects with a pre-estimated background image of the scene. However, this approach limits the method to indoor scenes with stable backgrounds, static cameras and without illumination changes or occluding objects.

4.2 Overview

An overview of the proposed system is given in Fig. 4.1. The first stage is unsupervised training, which happens simultaneously in the image domain on decoded I-frames to estimate the scene background (see Sec. 4.4) and in the compressed domain to segment the scene into regions with similar motion (see Sec. 4.5). Information from both processing domains is later combined during the object detection and tracking stage.

Before we explain our method in more detail, some information about the analyzed real-word test sequences is provided. They have been shot on a highway in the Bordeaux area under normal daytime conditions. The author would like to thank *Adacis Sarl*[3] for the provision of the sequences.

4.3 Test sequences

The test sequences used were shot at frame rates ranging from 15 to 25 fps and have been encoded with the SVC reference implementation JSVM, available at [19]. Two spatial lay-

[3] http://www.adacis.net/

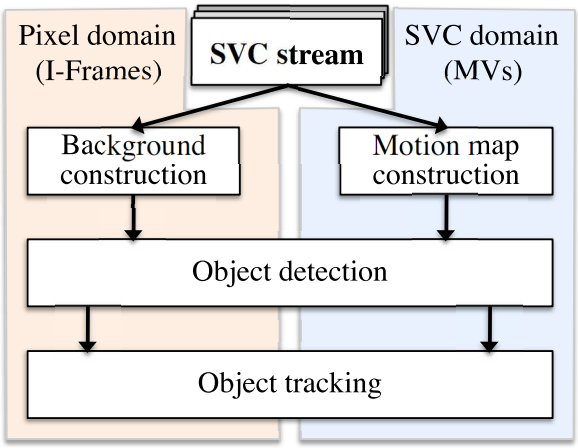


Figure 4.1: System Overview

ers (360x288, 720x576) have been coded under the *Scalable Baseline profile* with hierarchical B-picture prediction for temporal scalability. The *Group-of-Picture* (GOP) size is typically chosen sufficiently small for surveillance applications in order to add detection robustness for crowded scenes and was set to 8, resulting in the periodic pattern *IBBBBBBB*. The compressed domain analysis described below is based on the MVs present in the stream. Only the entropy coding of SVC has to be reversed in order to access the displacement values in quarter-pel precision per MB. An example of SVC MVs and MBs for a traffic scene is shown in Fig. 4.2. Each MB has a base size of 16x16 pixels and is sub-divided into smaller regions of minimal 4x4 pixels. Small MB partitions usually occur in high-textured areas in motion in order to increase coding efficiency.

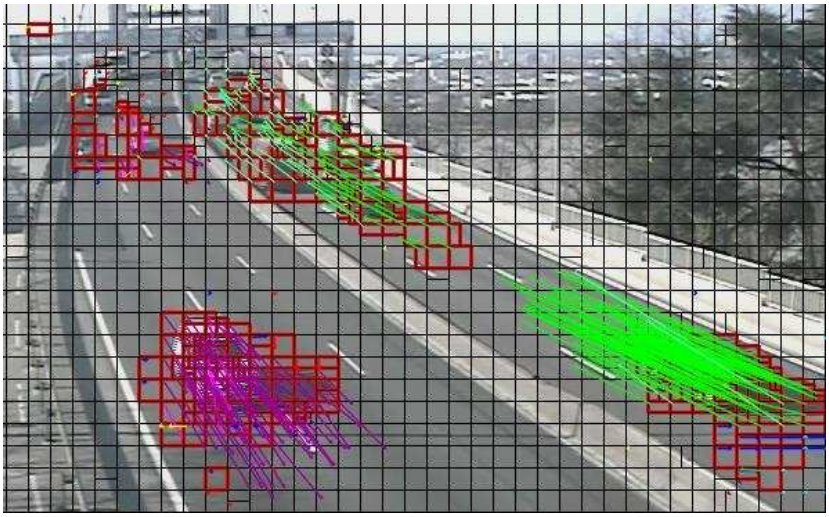


Figure 4.2: SVC MBs and MVs in a traffic scene. Outlier blocks of the GME are marked in red. The color of a MV corresponds to its angle (see Fig. 4.4).

For the construction of the scene background and pixel domain object detection, all I-frames, i.e., each 8th frame in this case, are fully decoded. MPEG-2 based approaches often make use of low-resolution DC images to avoid full decoding, but this is not applicable to H.264 because of the spatial prediction dependency on neighboring blocks [2].

4.4 Training Stage I - Background estimation

In the first training stage we estimate the background image of the scene. To do so, we apply the *Gaussian Mixture Model* (GMM) algorithm based on [39, 99]. It consists of a weighted sum of k Gaussian densities, which allows the color distribution of a given pixel to be multi-modal. Modeling the history of pixel values by several normal distributions helps the method to be more robust against local illumination changes and occlusions. Typically, $k = 3$ or $k = 5$ distributions are used, and the parameters of the mixture (weight w, mean μ, and covariance Σ) are updated dynamically over time. The probability P denotes the probability of occurrence of a color u at the current pixel s at time t and is given as

$$P(I_{s,t} = u) = \sum_{i=1}^{k} (w_{i,s,t} * \mathcal{N}(I_{s,t}, \mu_{i,s,t}, \Sigma_{i,s,t})). \tag{4.1}$$

$\mathcal{N}(I_{s,t}, \mu_{i,s,t}, \Sigma_{i,s,t})$ is the ith Gaussian model and $w_{i,s,t}$ its weight. The covariance matrix $\Sigma_{i,s,t}$ is assumed to be diagonal with $\sigma^2_{i,s,t}$ as its diagonal elements.

For each pixel, the first step consists of determining the closest corresponding Gaussian, i.e., the Gaussian for which the intensity of the pixel is within T_σ deviations of its mean. The parameters of the matched component are then updated by

$$\begin{aligned} w_{i,s,t} &= (1-\alpha)\, w_{i,s,t-1} + \alpha \\ \mu_{i,s,t} &= (1-\rho)\, \mu_{i,s,t-1} + \rho\, I_{s,t} \\ \sigma^2_{i,s,t} &= (1-\rho)\, \sigma^2_{i,s,t-1} + \rho\, (I_{s,t} - \mu_{i,s,t}), \end{aligned} \tag{4.2}$$

with α being a user-defined learning rate, with ρ being defined as

$$\rho = \alpha\, \mathcal{N}(I_{i,s,t}, \mu_{i,s,t}, \Sigma_{i,s,t}). \tag{4.3}$$

The weight of unmatched components is updated with

$$w_{i,s,t} = (1-\alpha)\, w_{i,s,t} \tag{4.4}$$

and $\mu_{i,s,t}$ and $\sigma^2_{i,s,t}$ remain unchanged.

If no component matches, a new Gaussian with mean $I_{s,t}$, a large initial variance σ_0 and a small initial weight w_0 is created to replace the existing Gaussian with the lower weight. Once each Gaussian is updated, weights are normalized and distributions are ordered based on the value $w_{i,s,t}/\sigma_{i,s,t}$. Only the B most reliable distributions are chosen, where

$$B = \underset{k}{\operatorname{argmin}} \sum_{i=1}^{k} (w_{i,s,t} < T_{bg}). \tag{4.5}$$

Pixels which are more than T_σ standard deviations away from any of the B distributions are labelled as foreground. Since the constructed background will be used to assist object detection and tracking, it is crucial to update it regularly to adapt to changing weather conditions. Once the initial background estimation is completed we continue updating it each incoming I-frame.

(a) Snapshot (b) Estimated background

Figure 4.3: Example for I-frame based background estimation of a traffic scene

An example snapshot from a test sequence together with its estimated background image is shown in Fig. 4.3. The method achieves very good results for the surveillance videos that we analyzed within this work. Further examples are shown in the appendix in Fig. 4.11.

4.5 Training Stage II - Motion Map Construction

The second learning stage takes place at the same time as the background image construction, with the difference that only B- and P-frames between I-frames are used. The goal is to segment the scene into regions of similar motion with respect to the moving direction. In the following, we refer to this segmented image as the motion map.

We accumulate B-frame MVs in order to obtain a MB based estimate of the dominant motion direction. The angles of all non-zero MVs are calculated and stacked in the 3-

dimensional tensor $\mathbf{M}$ of size $[w \times h \times T]$, where w and h are the image width and height in minimal sub-MB resolution (4x4 pixels/block) and T is the duration of the learning stage in frames. The learning duration T depends on the video sequence and varies for different cameras and times of the day. The busier the scene, the shorter T. We dynamically stop when both of the following conditions are met:

- At least ρ_{MV} non-zero MVs have been stored in $\mathbf{M}$.
- At least ρ_I I-frames have been accumulated for background construction (see Sec. 4.4).

The average value for T in the analyzed test sequences was approximately 10 seconds. $\mathbf{M}$ now contains the motion history of each MB during the learning stage. For each MB at position (x, y), the principal angle of motion α_{MB} is determined by the temporal median of all MV occurrences at the respective position. In order to visualize the map of median angles, we use a 12-bin angle color chart, shown in Fig. 4.4. Each angle in the final motion map is represented by its associated color. The resulting image visualizes the principal angle of motion of every block. Figure 4.5b shows an example for such a motion direction map. The different road lanes with similar direction of each scene become visible and are clearly separated by their color, i.e., by the dominant motion angle. Among other things, we use these motion maps later on in order to split merged object masks that span over two regions with opposing directions (see Sec. 4.7).

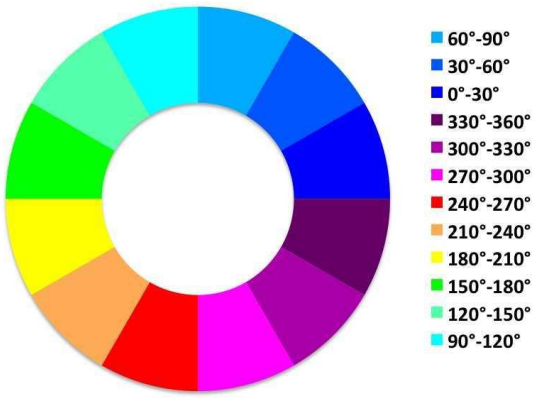
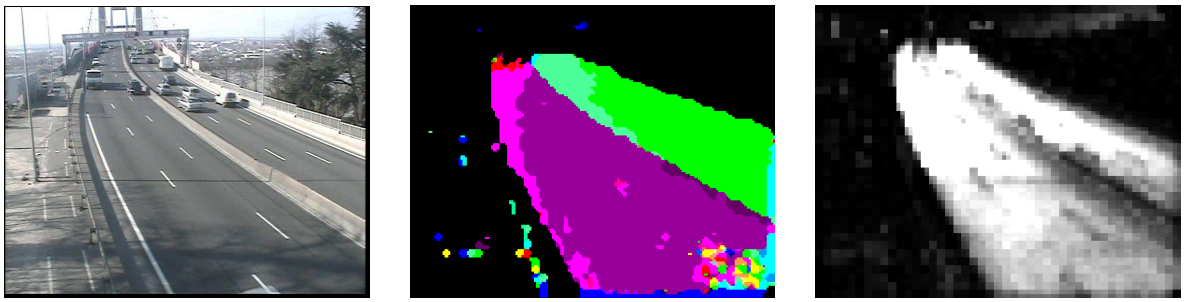


Figure 4.4: Angle color chart

Furthermore, gray-scale motion density maps of the scene are constructed by counting the number of non-zero MVs at all MB positions. At each MB position, we increment the gray value with each occurring MV and finally normalize by the duration of the learning stage T. An example of a resulting density map is shown in Fig. 4.11c. It can be observed that fast lanes are usually less occupied than others. Looking at the area beyond the horizon line, one can notice that the motion density becomes approximately zero and the motion

direction map shows no angle. In the context of scene geometry analysis, this observation will be used in Chap. 5 of this work to infer the position of the horizon line.



(a) Screenhot (b) Motion direction map (c) Motion density map

Figure 4.5: Example for motion direction and motion density map. The colors in (b) correspond to dominant motion angles (see angle chart in Fig. 4.4). Test sequence *seq5*

Further examples for backgrounds, motion direction and density maps are provided in Fig. 4.11 in Appendix A. We obtain good segmentation results for all test sequences and the estimated angle of motion per region is well approximated. Problematic are border macroblocks, because their motion vectors show random orientation due to newly appearing or disappearing objects. This problem can avoided by cropping one MB at all image borders.

4.6 Camera Motion Support

Although the majority of traffic surveillance cameras are static and show a fixed view, the deployment of Pan-Tilt-Zoom (PTZ) cameras is steadily increasing. These types of cameras are either manually controlled by operators or they follow pre-programmed, periodically repeating motion patterns to monitor multiple viewing directions alternately, e.g., at road intersections.

Due to background subtraction on I-frames, the proposed object tracking system necessitates a fixed view. However, the system automatically detects view changes by estimating the global camera motion for each B- or P-frame in the background. The motion vector field serves as input to the estimation process. In case of bi-directionally predicted MBs which have multiple MVs that point to different references frame in the future and in the past, we inverse the MV referencing a past frame and use the average of both vectors in order to attenuate noise. In order to obtain uniform results, we scale all MVs by the distance to its respective reference picture. We estimate the 2D 6-parameter affine motion model by the compressed domain method presented in Sec. 3.2.

The computation time for global motion estimation (GME) increases exponentially with the size of the estimation support, which grows proportionally to the video resolution. Experiments showed that robust results can be obtained at video resolutions equal or higher

than approximately CIF, so we only process the spatial base layer at 360x288 to save processing time. On a standard desktop PC with 2.16 GHz and 1 GB of RAM, we achieve a GME performance of about 95 fps. Figure 4.6 shows the estimation results for a surveillance camera that switches between two pre-defined, periodically repeating orientations, which are very well detected. The estimated global motion in static camera phases is close to zero, which indicates that the system robustly rejects MVs on moving vehicles as outliers.

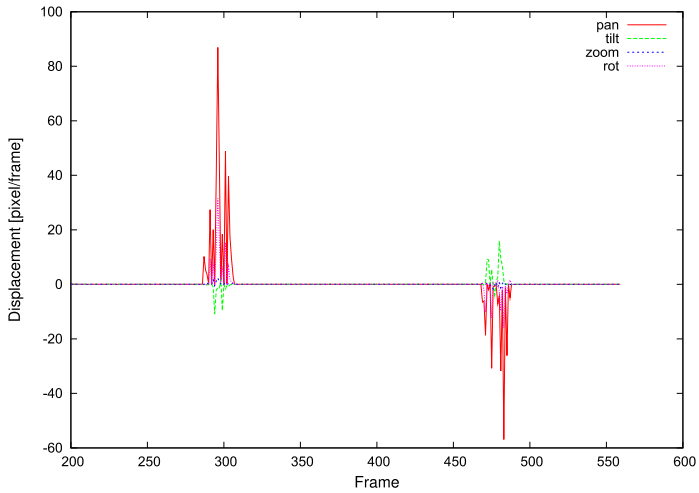


Figure 4.6: Estimated camera motion on traffic surveillance SVC stream. The peaks indicate rapid viewpoint changes of a PTZ camera

If the estimated displacement exceeds a pre-defined threshold of 2 pixels per frame for more than 5 successive frames, the system is re-initialized and the algorithm restarts with a new learning stage. At the given resolution, experiments showed that the estimated camera motion is mostly due to noise and wind below this threshold, which has to be scaled proportionally as function of the video resolution. In case of periodically repeating camera motion patterns, the GME results are used to detect view changes and detection and tracking can be restarted without repeated training.

An important by-product of the GME are outlier masks of all MBs that do not follow the global motion. Although the majority of outliers are caused by moving objects, noise is introduced due to low-textured areas and effects like shadows. Spatio-temporal filtering is applied to alleviate the impact of these effects. Figure 4.7 shows an example of an outlier mask at macro-block resolution after GME, where two common problems can be observed: Moving cast shadows are included in the masks and distant objects that appear small and hardly moving are missed.

Figure 4.7: Compressed domain outlier masks after GME. Bright regions represent foreground masks.

4.7 Object detection

Once the training stage is complete, we start the object detection and tracking algorithm. Foreground/background segmentation in the **compressed domain** is performed via the GME module on B- and P-frames, like briefly explained at the end of the previous section and in more detail in Sec. 3.2 of Part I.

Due to high vehicles like trucks and long cast shadows that are included in the object mask, the silhouettes of vehicles driving on opposing lanes may be merged. To overcome this problem, we split all foreground masks at positions where the motion map is either zero or where regions are touching whose dominant motion varies by more than 80°, i.e., on borders between opposing lanes. On the resulting, filtered foreground masks (see Fig. 4.7), blob extraction is performed using the algorithm proposed by Chang in [100], which traces contours and labels connected componenents in a single pass.

Object detection in the **pixel domain** – performed on decoded I-frames – consists of the following processing steps:

1. Background subtraction with continuously updated background
2. Suppression of noise in static regions
3. Suppression of moving cast shadows
4. Spatial filtering
5. Object/Blob detection.

The result of background subtraction is usually a rough foreground mask that includes moving cast shadows and noise in non-relevant areas of the image, so further post-processing

steps are necessary. In order to remove background (BG) noise in irrelevant parts of the image, we delete foreground (FG) pixels where the motion map and the motion density map are both zero (see Sec. 4.5, and Fig. 4.11 for examples).

In the next step, moving shadows are detected and suppressed. This is especially important when the sun is low, since the cast shadows of high vehicles like trucks may span multiple lanes (see Fig. 4.8). Moving shadows are most of the time detected as foreground in the pixel domain as well as in the compressed domain. The basic idea of shadow detection is to estimate how a cast shadow influences the value of background pixels. Many works have been published on this topic, including [101, 102, 103, 104, 105]. We apply a non-parametric detector based on the approach proposed by Horprasert et al. in [103], where measures for the brightness and color distortion in case of drop shadows are used to classify shadow pixels with respect to the background model.

Let

$$I_i = [I_R(i), I_G(i), I_B(i)] \tag{4.6}$$

denote the RGB color value of pixel i and

$$E_i = [E_R(i), E_G(i), E_B(i)] \tag{4.7}$$

the expected color in the background image. The brightness distortion α_i reflects the brightness of the current pixel in comparison to its expected value and is obtained by minimizing the function

$$\alpha_i = \underset{\alpha}{\operatorname{argmin}}(I_i - \alpha_i E_i). \tag{4.8}$$

The scalar value α_i equals 1 if the brightness of the given pixel is the same as in the background image, takes values less than 1 if it is darker, and greater than 1 if it is brighter than the expected value. The color distortion CD_i is defined as the orthogonal distance between the observed color and the expected chromaticity line, defined as the line which passes through the origin and the point E_i in the RGB color space:

$$CD_i = ||I_i - \alpha_i E_i|| \tag{4.9}$$

A pixel is considered to be shadow if it has similar chromaticity but lower brightness than the expected value of the background pixel, expressed as

$$\tau < \alpha_i < 1, \tag{4.10}$$

where τ is a user-defined threshold which we set to 0.45. For a detailed explanation of the algorithm, the reader is referred to [103]. An example of the shadow detection result is

shown in Fig. 4.8. It can be noticed that cast shadows are well captured, however a over-detection occurs, most notably on reflecting windshields.

Figure 4.8: Shadow detection result. Shadow pixels are marked in red

The detected shadows are subtracted from the foreground mask, which is morphologically filtered afterwards in order to obtain the final foreground mask image, as shown in Fig. 4.9a.

(a) (b)

Figure 4.9: (a) I-frame foreground mask after processing. Bright regions represent foreground mask. (b) Detected blobs. Each blob is shown in different color.

To extract and label connected components in the foreground mask image, we use the same blob extraction algorithm that was applied on the compressed domain foreground masks [100]. The detected blobs for the example sequence are shown in Fig. 4.9b, where different colors represent different objects. All detected objects having a minimal size are kept and used to initialize the tracking algorithm.

4.8 Object tracking

The proposed object tracking mechanism combines I-frame based detection with motion based results. Detection on I-frames is considered to be more reliable than in the compressed domain, because it can cope with shadows and static objects to a certain extent. However, it tends to over-detect, so objects are sometimes split and more blobs than real objects are detected. On the other hand, compressed domain analysis often suffers from under-detection, i.e., multiple objects are merged in one blob and small objects are missed. This occurs because shadows are usually detected as moving objects and because MVs are only available at MB level, so at a significantly reduced resolution.

Object tracking is performed on a GOP basis, including one I-frame and seven B-frames in our case. A validation of the GOP tracking is performed with the help of the I-frame of the subsequent GOP. The tracker is initialized with objects that have been detected in the first I-frame after training is complete. Within the bounding boxes of all objects, *Scale Invariant Feature Transform* (SIFT) [106] descriptors are calculated. SIFT has been chosen because of its proven robustness in respect to viewpoint and illumination changes. In a comparative study published in [107], SIFT outperforms other local descriptors. However, feature calculation and matching are very computation intensive. The computational overhead is greatly reduced by calculating features only on detected I-frame foreground regions.

After feature computation, all I-frame objects are labelled and their masks are projected onto the following B-frame foreground mask. Labels from superposing regions are copied and the intra-GOP tracking process repeats the following steps for each B-frame B_t:

- Estimate local motion per detected object
- Project masks to predicted positions (by local motion) in the following frame B_{t+1}
- Match overlaying regions and copy labels

If multiple I-frame objects overlay with a B-frame object mask, all labels are copied to it. If a compressed domain object appears throughout the GOP where no pixel domain object has been detected before, a new label is assigned. The local object motion which is used to predict the position in successive frames is estimated similarly to global camera motion (see Sec. 4.6), except that only the MVs covered by the mask are considered as active estimation support.

At the end of each GOP cycle, object label validation is performed by matching SIFT points between the objects of the two subsequent I-frames. Although SIFT is typically robust, some miss-matches cannot be avoided. We apply certain additional constraints to validate or reject matching feature points:

- Matches of features that lie on the borders of object bounding boxes are rejected
- Matches between two points that lie in regions of the motion direction map with opposing angles are rejected (see Sec. 4.5 and motion maps in Fig. 4.11)
- Matched feature points that are too far away from their position predicted by local motion are rejected

An example for accepted and rejected SIFT matches between two I-frames are shown in Fig. 4.10. The majority of false matches is eliminated by imposing the constraints mentioned above. Object masks that are touching the image borders are marked as entering or leaving the scene under consideration of local motion estimates. In case multiple vehicles had been initialized as one single object, as is often the case for far away objects that are close together, we only know after the splitting that it contained multiple objects and assign new labels as the split is detected.

4.9 Results

For relatively simple scenes with only few objects and little or only short occlusions, object detection in the compressed domain alone leads to satisfying results. In crowded scenes however, the additional use of pixel information is important in order to eliminate moving shadows or to separate partially occluding objects, which is typically the case in traffic surveillance videos.

Table 4.1 shows the results for unsupervised vehicle counting on the test sequences shown in Fig. 4.11, obtained by compressed domain only analysis without training [43] and by combining compressed and pixel domain information. The ground truth was obtained manually and the videos have an average duration of about 30 seconds. Each moving object that entirely left or entered the image on the bottom, left or right image border within the analyzed period has been counted.

The under-detection of the compressed domain only approach is mostly due to occlusions of multiple vehicles that are close together, resulting in merged object masks. This can be corrected to a certain extent by also taking pixel domain information of decoded I-frames into account (see Sec.4.8). However, multiple distant objects that are close to each other are very difficult to separate in either the image or the compressed domain. Due to better resolution and shadow suppression, image domain analysis allows to detect multiple objects earlier.

In the bottom rows of Fig. 4.11, the estimated trajectory results in the compressed domain domain alone and from the combined approach presented above are shown. The most

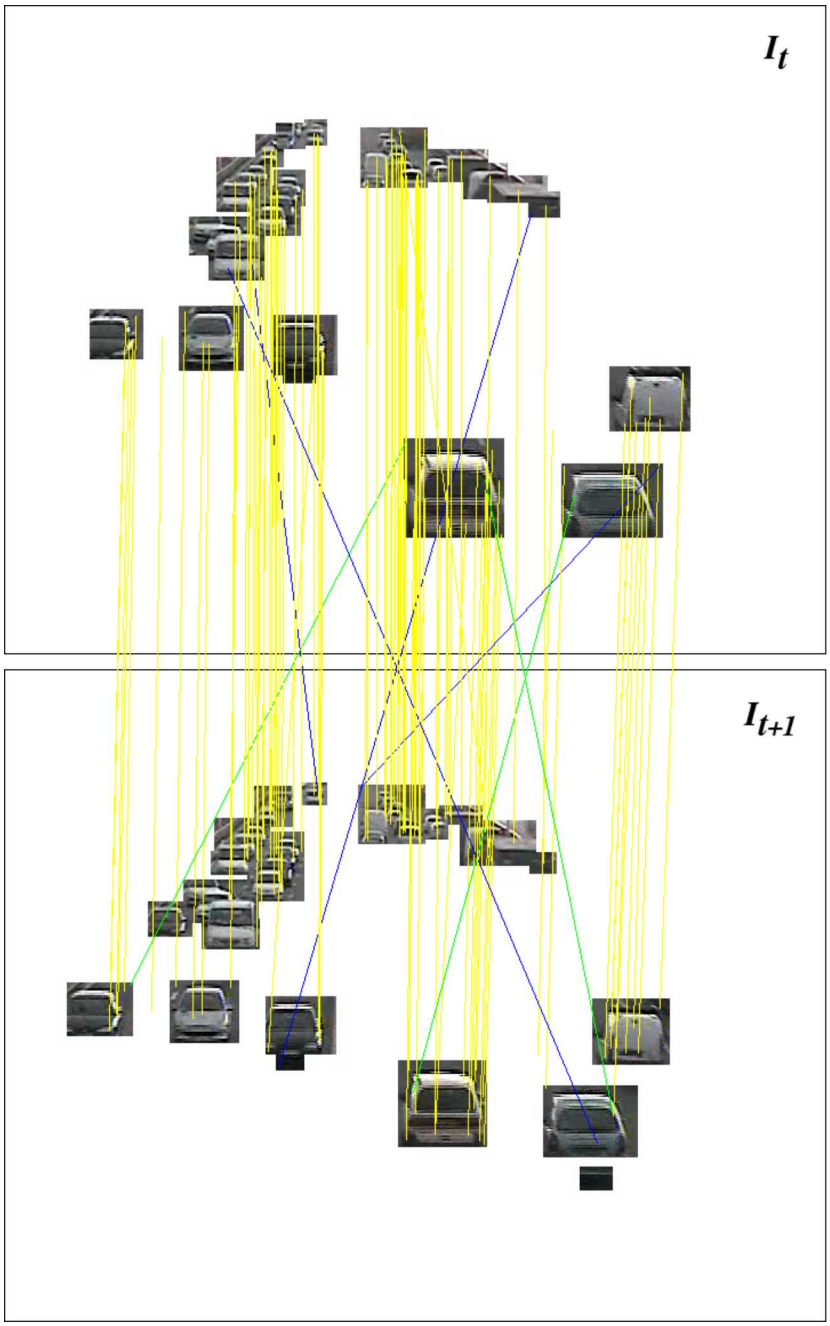


Figure 4.10: SIFT matches between I-frame objects. Accepted matches are shown as yellow lines, blue and green ones have been rejected.

	Ground truth, number of vehicles	Compressed domain only	Proposed approach
Seq 1	52	33 (−19)	47 (−3)
Seq 2	29	22 (−7)	27 (−2)
Seq 3	25	22 (−3)	26 (+1)
Seq 5	17	16 (−1)	17 (0)

Table 4.1: Vehicle counting results. Number in brackets denotes difference from ground truth

obvious difference can be observed when looking at the results for *Sequence 1*. The scene is very crowded and many objects are merged in compressed domain foreground masks, sometimes even affecting objects on opposing lanes. *Sequence 5* suffers from similar problems because of the street being a turning and the cast shadows of trucks spanning multiple lanes. Through SIFT object validation after each GOP, the results could be greatly improved.

4.10 Summary and Conclusions

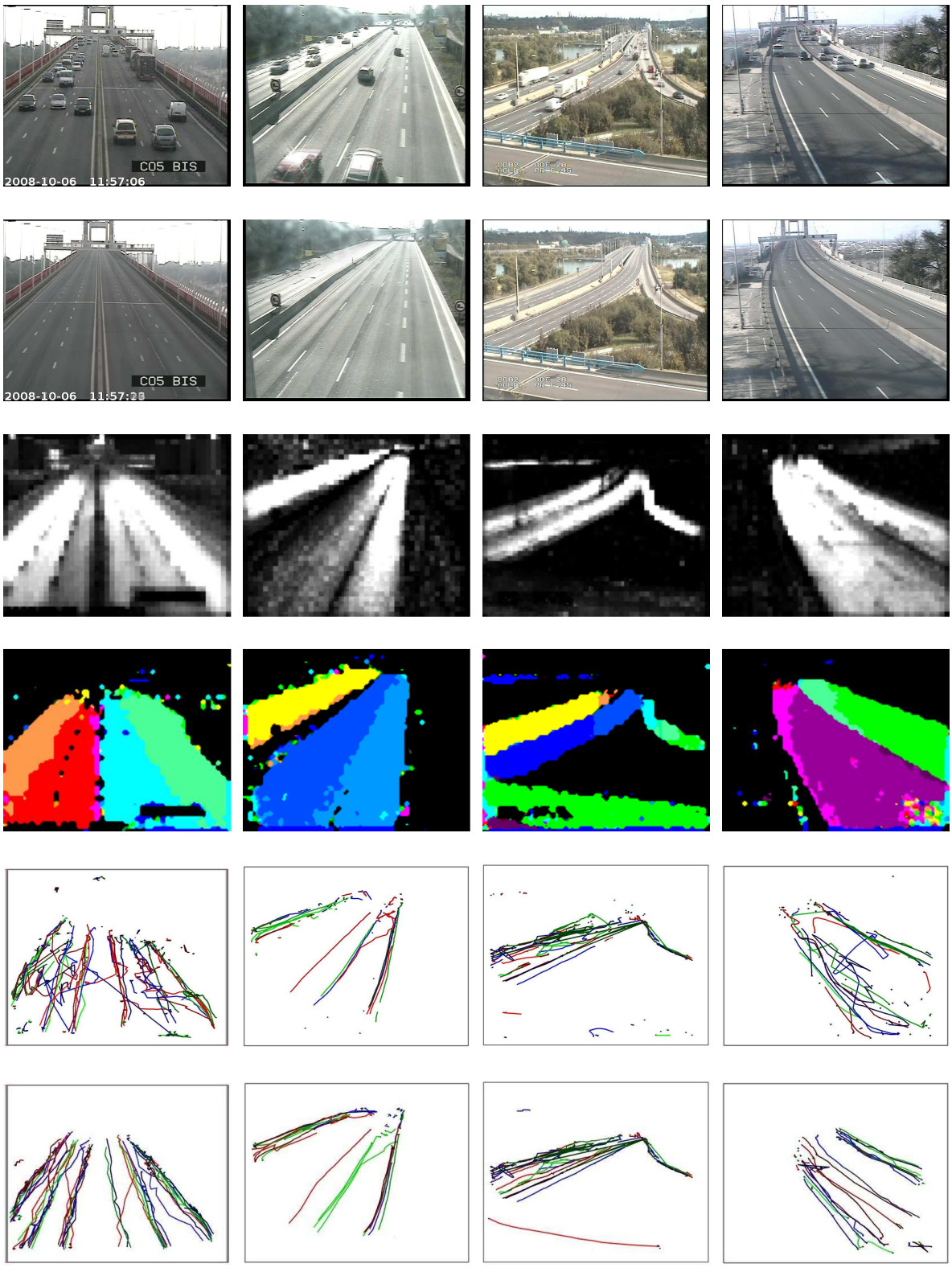
Traffic surveillance systems often employ a large number of cameras, hence the scalability of the whole system is an important design issue. We presented an efficient traffic analysis system that combines the precision of pixel domain analysis with the computational benefits of compressed domain analysis. The proposed approach lead to the publication proposed in [108]. The main contribution of this work lies in the combination of compressed domain and image domain analysis with a novel, unsupervised training stage in the compressed domain.

H.264/SVC is a suitable coding standard for surveillance networks due to its high coding efficiency and its scalability features on stream level. Within the scope of this work, the motion information that is present in SVC streams is exploited in different ways to enhance the overall system performance:

- Motion direction and density maps are constructed to increase the robustness of the detection and tracking algorithms
- Global motion estimation is performed to detect view changes in case of PTZ cameras
- Moving objects are detected on the basis of block-based motion vectors
- Object positions are predicted by their local motion estimates

The shortcomings of pure compressed domain analysis of crowded scenes have been pointed out. This lead to the proposition of a validation mechanism of the tracking results by using robust, local SIFT feature points on decoded I-frames. By limiting the feature point calculation to detected foreground regions, the increase in computational complexity is moderate. Experiments showed that the feature computation and matching process can be accelerated by factor of 5 to 7 at similar performance when replacing SIFT with *Speeded Up Robust Features* (SURF) [109].

The robustness of the proposed method was demonstrated by results for automatic vehicle counting and trajectory estimation in real-world test sequences. However, other applications and extensions of the proposed methods can be easily realized to perform further



(a) *Sequence 1* (b) *Sequence 2* (c) *Sequence 3* (d) *Sequence 5*

Figure 4.11: **Rows from top to bottom: 1st**: Screenshots ©*DIRCE/DIRA*. **2nd**: Backgrounds. **3rd**: Motion density maps. **4th**: Motion direction maps (for color correspondance see Fig. 4.4). **5th**: Raw, unprocessed trajectories without prior training. **6th**: Trajectories of proposed system.

analysis tasks. An example is the triggering of alarms when a motorist is driving against the traffic on highways. Without the need for object tracking, this can be very efficiently implemented by comparing the observed MVs with the dominant direction in the motion map.

Future work includes to account for approximated scene geometry information (see Chap. 5) to improve object detection and tracking results, as well as the incorporation of the object distance estimation scheme presented in Sec. 3.5. Further ideas also include the refinement of motion direction maps by replacing the block-based median values by circular *von Mises*[4] distributions, and the detection of lane separations through clustering or segmentation of the refined motion maps and trajectories. The presented system is also able to provide input for higher level analysis like behavior classification and the detection of unusual events, like demonstrated by Sas et al. [110], Wang et al. [111], Morris et al. [112], or Ivanov et al. [113].

[4]The equivalent in circular statistics of the Gaussian or normal distribution.

Chapter 5

Scene Understanding

Numerous computer vision tasks like object detection, tracking, scene segmentation and behavior analysis can benefit from information about the scene and its basic geometry. Perspective projection obscures the relationships that are present in the actual scene – an object that is nearby will appear much bigger than one that is far away, even when both are the same size in reality. Perspective plays an important role not only because it affects the size of the object's projection on the image plane, but also for the estimation of its speed for example. Approximate knowledge of the 3D scene geometry can provide very useful information during analysis tasks.

Another aspect that plays an increasingly important role in computer vision is the consideration of context. Although sophisticated object detectors, classificators and scene segmentation schemes are already available, such tasks still remain challenging research problems. Information about the context can deliver very useful information to enhance detection results. If for instance a small foreground object is detected and tracked within a region considered as sky, it is far more probable that it represents a bird, plane or helicopter than when it is detected on a region assumed to be road, even if their silhouette or color appear similar.

The work that is presented in this chapter is inspired by the excellent publications presented by Hoiem et al. [96, 97]. The authors propose a unified approach for modeling the contextual symbiosis between the three crucial elements that are required for scene understanding:

1. Object detectors
2. Approximate camera position and orientation
3. Rough 3D scene geometry

Although Hoiem's work is based on the analysis of still images, the findings are equally true for video. Unfortunately, information about the scene is usually not available a priori.

Our goal is the estimation of the rough scene geometry and setup based on moving objects, which are extracted from monocular sequences in the H.264 compressed domain.

Concerning the three required elements for scene understanding mentioned above, the first – low-level object detectors – was already covered in Chap. 3 of this work. Based on the compressed domain tools and results presented previously, we propose a method to approximate the orientation angle of the camera in Sec. 5.2. For the special case of traffic surveillance scenes, we present an approach to roughly estimate the basic scene geometry by approximating the position of the horizon line in Sec. 5.3.1, followed by a method to segment the road surface in Sec. 5.3.3. Results are provided within the respective sections for better clarity.

5.1 Related work

The estimation of geometrical properties like the position of the road, the orientation of the camera pose or the distance of moving objects can be approached in a variety of different ways. The largest family of methods is based on multi-view sequences or stereo vision [114, 115]. Except from specialized applications like robot vision or dense surveillance networks, the majority of real-world scenarios consists of single-view setups. The family of algorithms based on monocular sequences is grouped under the keyword *Structure-from-Motion* (SfM), where ego-motion and changes in perspective are used to infer the 3D structure of the scene or of moving objects [116, 117].

The existing methods are based on the rich information provided on pixel level for the sake of precision. However, as demonstrated in [97], approximate results can deliver sufficient information for certain tasks like object classification. The use of image domain information enables numerous analysis tasks like the pixel-wise recovery of depth maps (e.g., [118]) or the estimation of the 3D structure of an object given a series of input images taken from different angles [119]. Most methods rely on multiple cameras (multi-view) or a series of images taken from different perspectives (SfM). A review of these methods would go beyond the scope of this work and the interested reader is referred to the literature for further information on the well-developed field of SfM and multi-view geometry [114, 117, 115].

Only very few compressed domain attempts have been published on this or similar topics. Mbonye [120] uses MPEG-2 compressed domain data to adjust the camera pose by attentive visual servoing tailored to a road traffic application with car mounted cameras.

In this work, we go one step further and exploit single-view compressed domain analysis results to infer the orientation angle of the camera without a priori knowledge of the scene setup. For the special case of traffic surveillance, we propose simple methods to determine the position of the horizon line and to perform road segmentation. Estimates of the horizon

line and the camera orientation allow us to project imaginary equidistant lines from the ground plane onto image plane, hence adding depth information to the sequence.

5.2 Camera Orientation Estimation

We estimate the camera orientation angle by analyzing the evolution of the object's ground contact position. The vertical position of an object in the image plane usually changes as a function of its distance to the camera. The way it changes with varying distance depends mainly on the angle of the camera itself and on the geometry of the surface the object is moving on. For the sake of simplicity, we assume that the surface is flat in the following. If no lens distortion is present, the relationship between the object's distance z to the camera (modeled as pinhole) and the vertical bottom position on the image sensor y_S can be expressed as

$$y_S = \frac{d}{2} + f * \tan(\tan^{-1}(\frac{z}{h_{cam}}) - \alpha_{cam}), \tag{5.1}$$

where d is the vertical dimension / height of the image sensor, f is the focal length, h_{cam} denotes the camera height and α_{cam} the orientation angle of the camera, relative to the ground plane. An α_{cam} of 0° corresponds to bird's eye view and 90° means the camera is parallel to the ground. Figure 5.1 illustrates the basic scene setup of a possible surveillance scenario. As a reference point for z in the image plane we use the lowest point of an object's silhouette, which approximately corresponds to the foot-ground contact point at inclined camera angles, i.e., at $\alpha_{cam} \geq 0^\circ$. This of course assumes that the observed objects are not flying.

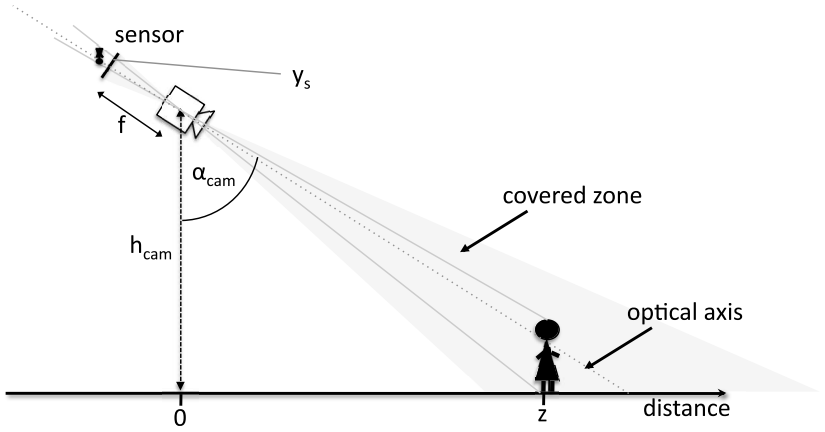


Figure 5.1: Possible camera setup.

Figure 5.2 shows theoretical examples of different parameter combinations and how they influence the relationship between the distance and the y-coordinate on the image. For the plots, we used the object's ground contact position as the reference point, fixed f at 35mm,

d at 24mm and varied the camera angle and height. The straight diagonal lines represent the linear case, which is only met at $\alpha_{cam} = 0°$. The values for f and d correspond to classic analogue cameras working with 35mm film[1]. For digital cameras with cropped sensors, the equivalent focal length is significantly smaller. A camera with a 1/3" CCD sensor for instance would only require a focal length of 4.8 mm to capture the same parts of the scene[2].

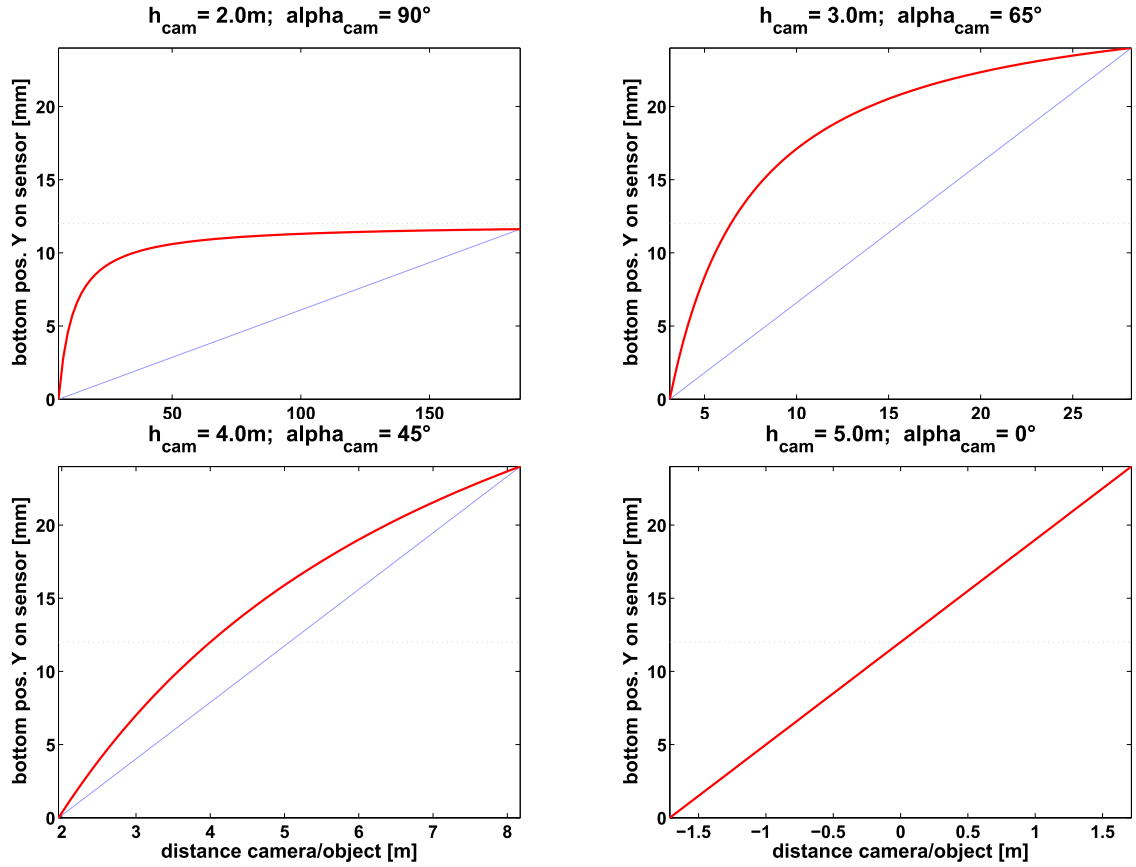


Figure 5.2: Relationship distance to vertical position in image

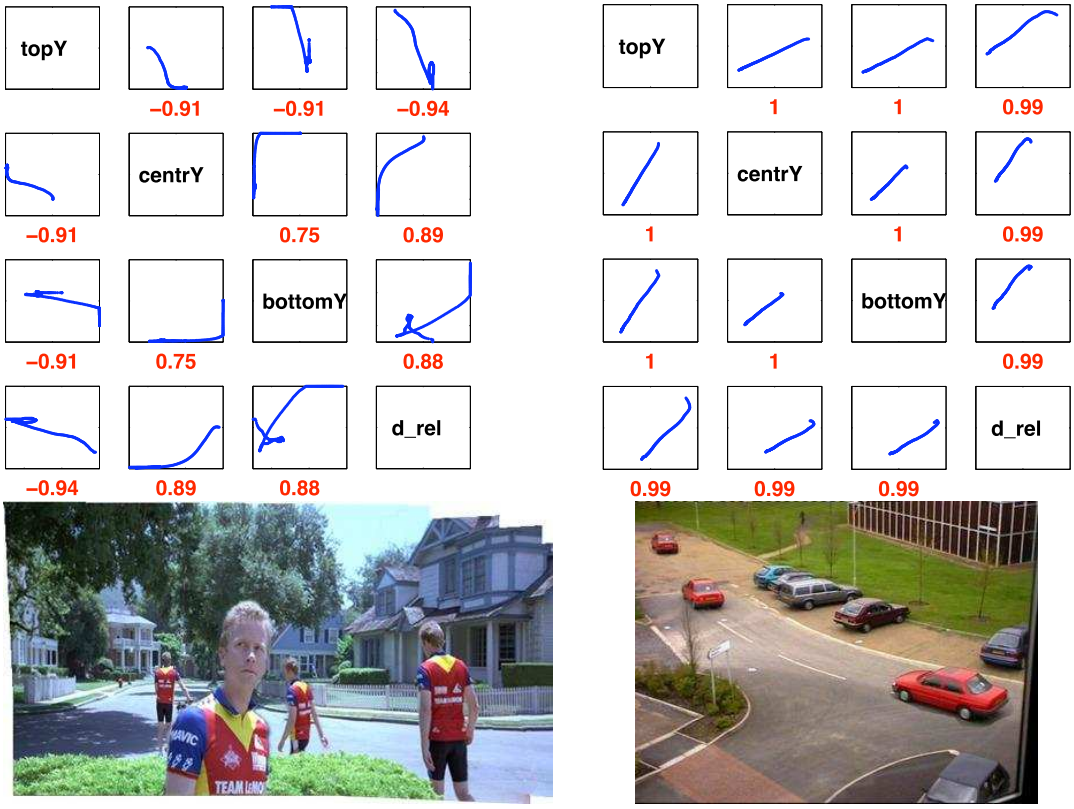
For horizontally orientated cameras with $\alpha_{cam} = 90°$, the corresponding plots for the top and centroid positions would differ from the ones shown for the bottom position. The Y-position of the object point located at the height of the optical axis ideally stays constant at changing distance. Reference points that lie higher than the optical axis ideally decrease in the Y-position with increasing distance. This changing behavior can be exploited to infer the basic relative geometry between object and camera.

Figure 5.3 shows matrix correlation plots of the relative distance versus the low-pass filtered Y-positions of top, centroid and bottom. When comparing the plots of both sequences *street* and *car*, it can be observed that the rigid nature of the car results in cleaner and more

[1]The dimensions of 35mm film are 36x24mm

[2]Applet for focal length conversion: `http://www.cambridgeincolour.com/tutorials/camera-lenses.htm`

sharply defined point clouds than for the non-rigid pedestrian in *street*. For the *car* sequence, it can also be observed that the plots for top, centroid and bottom vary in a similar way and always increase with growing distance, leading to all positive correlation values.

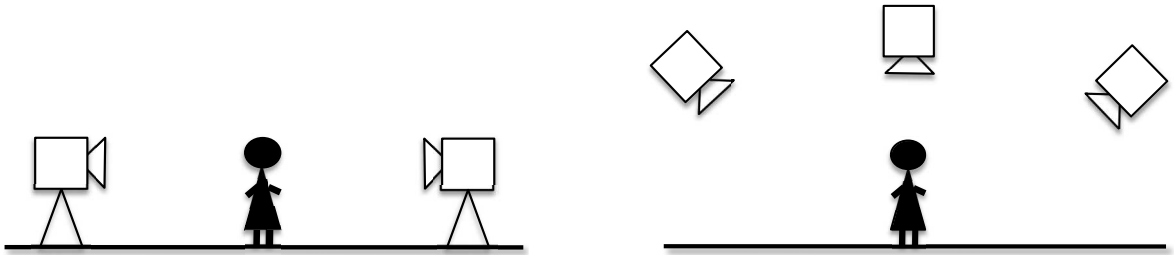


(a) *street* (b) *car*

Figure 5.3: **Top row:** Matrix correlation plots with estimated distance d_{rel} versus top, center and bottom Y-positions of moving object. The numbers below denote the covariance. **Bottom row:** Mosaic images of both sequences

The scatter plots for *street* are subject to more severe noise, most notably due to the non-rigidity of the object and due to the fact that the camera motion had to be compensated, which is another source for noise. The important subplot in Fig. 5.3a is the lower left one, showing the top position against the relative distance. The negative correlation value – opposed to the positive for centroid and bottom – indicates that the optical axis lies between the top and centroid and suggests that the camera is parallel to the ground.

We use this behavior to perform a first, rough classification of the camera orientation in *top view* (V_{top}) and *ground view* (V_{grd}). The two classes correspond to the basic situations depicted in Figure 5.4. We define reference covariance vectors for both basic orientations as $V_{top} \hat{=} [111]$ and $V_{grd} \hat{=} [-101]$. The values correspond to ideal values of the last row of



(a) Ground view (b) Top view

Figure 5.4: Basic camera pose classes

the matrix plot from Fig. 5.3. For the ground view model, the optical axis is assumed to be parallel to the ground and to point at an object somewhere between its feet and head.

For classification, we calculate the respective covariance vector for the moving object in the video scene and match it to the model with the smallest euclidian distance. Table 5.1 shows the euclidian distances for some of the sequences shown in Fig. 3.19. The bold values highlight the smallest distance and thus the classification results, which are correct in all cases. For sequences with multiple objects, we calculate the distances for all objects and average the results.

After this first classification, we fit the parameters of the model given in Equation 5.1 to the data given in the scatterplots d_{rel} vs. *Y-position*. If the object silhouette never touches the lower image border, we use the data from the bottom position, otherwise we use the less reliable top position. For fitting, a robust non-linear least squares algorithm is applied. This iterative estimation process consists of four steps:

1. Start with an initial estimate for each coefficient (f, d, h_{cam}, α_{cam}). Since the intrinsic camera parameters f and d are unknown, we assume a standard field of view and fix f at 35mm and d at 24mm. The initial values for h_{cam} and α_{cam} are chosen according to the pose classification results: For V_{ground}, we chose $h_{cam} = 1.8$m and $\alpha_{cam} = 90°$, corresponding to a shoulder camera setup. For V_{top}, we set $h_{cam} = 4$m and $\alpha_{cam} = 65°$, representing a possible surveillance camera setup.

2. Produce the fitted curve for the current set of coefficients. This involves the calculation of the Jacobian, which is defined as a matrix of partial derivatives taken with respect to the coefficients (h_{cam} and α_{cam} in our case).

3. Adjust the coefficients and determine whether the fit improves. For coefficient adjustment, the Trust-Region algorithm [121, 122] is applied.

4. Iterate steps 2 and 3 until convergence.

As a constraint, we limit the possible range of the variable α_{cam} to $0° \leq \alpha_{cam} \leq 90°$. Problematic during fitting is the unit-less nature of the relative distance measure, opposed to the

absolute distance values that are used in the model. We try to compensate for this by not limiting the range of the unknown camera height. This leads to unrealistic estimation results for h_{cam}, but within this work we are only interested in the camera angle α_{cam}, which is the determining factor for the curvature of the one-dimensional mapping function. The camera height h_{cam} does not change the curvature, but only the distance range.

5.2.1 Results

Figure 5.6 shows the fitted models over the scatterplots d_{rel} versus *Y-position* for *street* and *car*. The step-like behavior of the scattered data is due to the fact that we do not work at pixel level but on the coarse H.264 macro-block grid. At small object displacements in adjacent frames, the reference point is therefore more likely to stay in the same macro-block area. The fitted model for *car* matches better to the measured data because the object is rigid and always fully visible, whereas for the pedestrian in *street*, the more unstable top position was used due to the fact that his legs are cut off at the lower image boundary. Furthermore, the camera motion had to be compensated, which introduces additional errors, resulting in an overall poor fit for this sequence.

(a) *street* (b) *car* (c) *restaurant*

(d) *trainTracking* (e) *hallMonitor* (f) *surveillance*

Figure 5.5: Screenshots of sample sequences

The estimated camera angles α_{cam} for all sequences shown in Fig. 5.5 are given in Tab. 5.1. Since no ground truth is available, the estimated angles can for now only be evaluated by

	street	*car*	*restaurant*	*train*	*hall*	*surv*
V_{top}	1.95	**0.02**	2.22	**0.09**	1.19	**0.69**
V_{grd}	**0.90**	2.21	**0.24**	2.16	**1.10**	2.07
α_{cam}	90°	51°	83°	65°	78°	69°

Table 5.1: Camera pose classification and results for α_{cam}

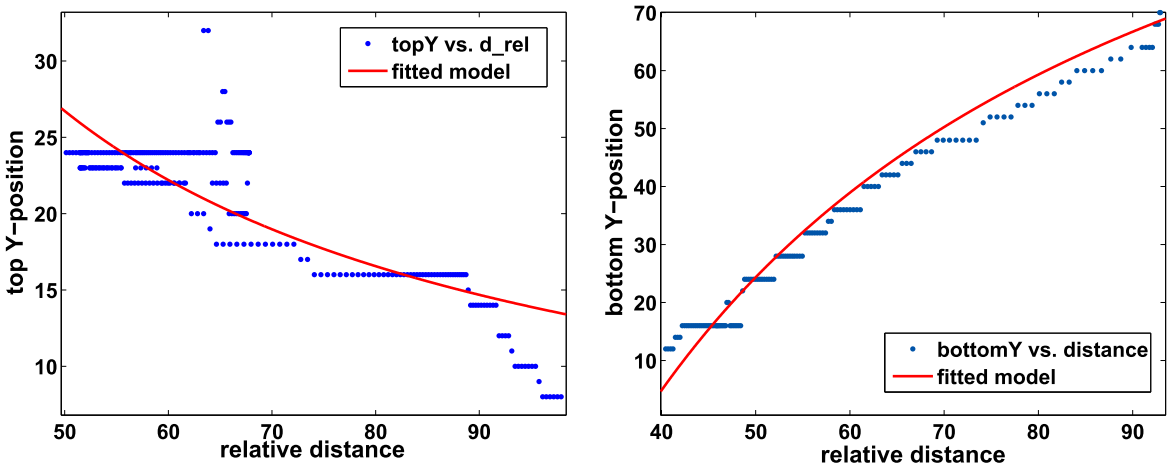


Figure 5.6: Scatter plots vs. fitted model. left: *street*. right: *car*

human judgement. Although the results are not very precise, they reflect the approximate camera pose better than the coarse classification in top and ground view.

5.3 Scene Geometry Approximation

The estimation of the camera orientation, presented in the previous chapter, is not restricted to certain types of videos and can be applied to any sequence where object detection and tracking delivers valid results. In this section, additional tools are introduced that are applicable when working in constraint environments like video surveillance. Focus is put on the analysis of traffic surveillance videos on highways, so this section can be regarded as an extension to the work presented in Chap. 4. Our goal is to demonstrate that rough, compressed domain motion information from monocular video sequences and uncalibrated cameras are sufficiently rich to deduce certain geometric properties that enhance the overall scene understanding.

In the following, two methods are presented. The first method approximates the position of the horizon line and estimates the camera angle. The second method segments the road in the scene, which represents the Region-of-Interest (ROI) in traffic videos. Both methods are based on the motion direction and density maps introduced in Sec. 4.5. The processing chain

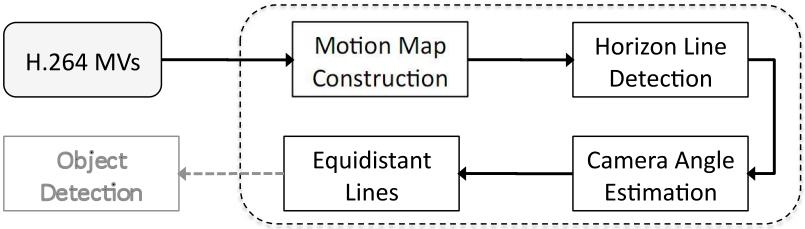


Figure 5.7: Overview of scene geometry recovery

is illustrated in Fig. 5.7. To recall, these motion maps are obtained during a preliminary, unsupervised training phase. The motion direction map reflects the median direction or angle of motion per macro-block, and the density map represents the average motion intensity per macro-block. Examples of motion maps are shown in Fig. 4.11.

5.3.1 Estimation of Detection Limit and Horizon Line

When working with outdoor sequences like traffic surveillance, information about the position of the horizon line can deliver important knowledge about the scene geometry, which on the other hand can be incorporated in detection and object classification schemes to increase the overall system robustness [97]. Assuming the camera is mounted parallel to the ground so that the horizon can be approximated as a horizontal line, its vertical position can be efficiently approximated by analyzing the motion maps. To achieve this, one descriptor $I(y)$ is formed per motion map that represents the line-wise intensity as a function of the vertical position in the image.

After binarizing the motion direction and density maps, this can be efficiently realized by summing up all X-values at fixed Y-positions in the image. With $(x, y) \in \{0, 1\}$ denoting the element at position x and y in the motion map, the intensity descriptor which is calculated for the angle ($I_A(y)$) and the density map ($I_D(y)$) is obtained by

$$I_{A,D}(y) = \sum_{i=0}^{width} (x_i, y). \tag{5.2}$$

To alleviate the impact of noise and to increase robustness, we perform moving average filtering and take the mean value of both characteristics as estimation basis. Examples of filtered intensity curves for the motion maps of *seq5* are shown in Fig. 5.8a. We now determine two distances that have an impact on scene analysis:

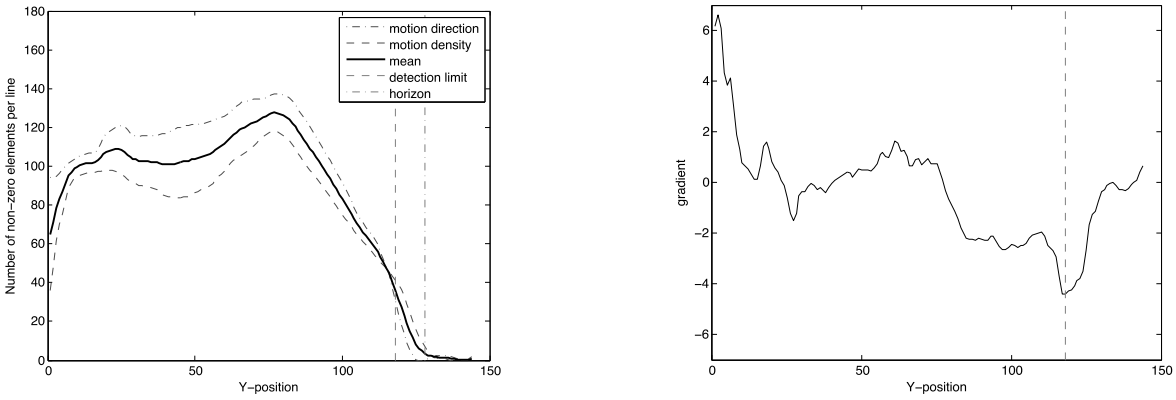
1. The **detection limit**, defined as the distance beyond which no reliable object detection can be performed.

2. The **effective horizon line**, defined as the horizontal line that separates sky and ground assuming a flat world. In the case of highway surveillance, we approximate this line as the maximal vertical position of the visible road.

We define the vertical position $LimY$ of the detection limit as the position where the line-wise intensity drops most significantly, i.e., where the gradient ∇ has its global minimum:

$$LimY = \min\left(\nabla\left(\frac{I_A(y) + I_D(y)}{2}\right)\right), \tag{5.3}$$

as illustrated in Fig. 5.8. The horizon line is estimated to be where the intensity function drops under 5% of its maximal value. Ideally this threshold would be zero, but 5% was determined empirically to account for noise and miss-detections.



(a) Vertical evolution of the intensity (b) Gradient and its minimum

Figure 5.8: Example of detection limit and horizon line estimation

For the traffic surveillance sequences that were used in Chap. 4, the estimated horizon lines are depicted in Fig. 5.9. All horizon lines are very well approximated, even for the rather complex sequence 5. It has to be noted that we do not necessarily aim at estimating the true horizon that marks the border between visible sky and ground, but at approximating the effective horizon, which represents the maximal distance up to where moving objects can be sensed by the system. The detection limit on the other side represents a empirical limit up to which objects can be successfully detected and tracked in the compressed domain.

The method delivers good horizon approximations on all test sequences. However, this simple way of inferring the position of the horizon line only works in constraint environments. The training period for the construction of the motion maps has to be sufficiently long so that a certain number of moving objects occurred throughout the scene. Furthermore, the camera has to point in the direction that follows that course of the road, so the presented method does not work for cameras that are oriented orthogonally to the road.

(a) sequence 1 (b) sequence 2 (c) sequence 5 (d) sequence 5

Figure 5.9: Estimated detection limits (red) and horizon lines (blue) on traffic surveillance sequences.

Alternatively, the position of the horizon in the image plane could be determined a posteriori using the results of the object detection and tracking stage. A straight forward way would be to take the maximal vertical position of detected objects as an approximation, or to use the results of the relative distance estimation and to keep the maximal Y-positions of the most far away objects. However, the advantage of the approach presented above is that the horizon line can be determined very efficiently right after the training stage, i.e., once the construction of the motion map is complete, without relying on potentially noisy or faulty results from former processing steps.

5.3.2 Camera Orientation Estimation - Revisited

In Sec. 5.2 we presented a general approach to estimate the camera orientation based on tracking results of moving objects. Since the estimation itself is based on the results of a processing pipeline with multiple stages, many error sources are introduced, which leads to very rough approximations of the camera orientation. In this section, we propose another way to estimate the camera orientation in constraint environments, where the only condition is that the horizon line is visible and its position is known. A simple approach to determining the vertical position of the horizon in traffic surveillance videos was presented in the previous section.

As in Sec. 5.2, the method to determine the camera angle presented below is also based on the relationship between the distance to the camera z of a point on the ground and the vertical position of its projection on the image sensor y_S. This relationship was already provided in Eq. 5.1, but we reproduce it here for better readability:

$$y_S = \frac{d}{2} + f * \tan(\tan^{-1}(\frac{z}{h_{cam}}) - \alpha_{cam}).$$

Recall that d denotes the height of the image sensor, f the focal length of the camera lens and h_{cam} the height of the camera with orientation angle α_{cam}.

	seq 1	*seq 2*	*seq 3*	*seq 5*
α_{cam} based on objects	86.26°	71.11°	67.43°	82.30°
α_{cam} based on horizon	79.48°	75.84°	75.58°	77.64°

Table 5.2: Camera orientation results of traffic surveillance sequences

Under the flat ground assumption, the position of the horizon line on the image sensor is given at $z \rightarrow \infty$. Since

$$\lim_{z\to\infty} \tan^{-1}(z/h_{cam}) = \frac{\pi}{2} \triangleq 90°, \tag{5.4}$$

the vertical position of the horizon line $horY$ is given as

$$horY = \frac{d}{2} + f * \tan(\frac{\pi}{2} - \alpha_{cam}), \tag{5.5}$$

what approves that the position of the horizon line in the image plane depends only on the orientation angle of the camera, the dimension of the image sensor and the focal length, but not on the camera height [123]. For cameras that are parallel to the ground ($\alpha_{cam} = 90°$), potential lens distortions and the focal length also loose their influence, because the horizon always appears at $d/2$, i.e., in the center of the image. Equation 5.5 can be rewritten as

$$\alpha_{cam} = \frac{\pi}{2} - \tan^{-1}\left(\frac{horY - d/2}{f}\right), \tag{5.6}$$

so α_{cam} can be determined if the intrinsic camera parameters (d, f) and the vertical position of the horizon line $horY$ are known. Since the intrinsic parameters are unknown for our test sequences, we assume a standard lens setup equivalent to $d = 24$mm and $f = 35$mm and use the $horY$ we estimated in the previous section. For the four shown test sequences, this yields the results given in Tab. 5.2.

Unfortunately, no ground truth for our test sequences is available, but since the presented method bypasses any potentially erroneous object detection and tracking steps, the results are more accurate and stable than based on moving objects. A visual verification of the results can be carried out by drawing equidistant lines in the image plane, as illustrated in Fig. 5.11. Each segment should ideally contain the same number of lane separation markers, which is approximately the case.

5.3.3 Road Segmentation

Knowledge about the position of the road in the image plane can increase system performance and robustness. Performance gains can be achieved by limiting the processed ROI to the visible road surface, which also increases robustness because miss-detections outside the ROI loose their influence.

The motion direction and density maps already represent rough approximations of the frequented ground surface. We obtain a single, binary mask of the road in a straight-forward way by thresholding the average over both maps. Holes in the resulting binary mask are then filled through morphological operations and very small connected components are filtered out. A last post-processing step consists of rejecting all blobs that are situated entirely above the horizon line. A visualization of the road segmentation results is provided in Fig. 5.11. All important parts of the road have been captured; only the security lane in sequence 2 is cut off because no vehicle used it during the training period.

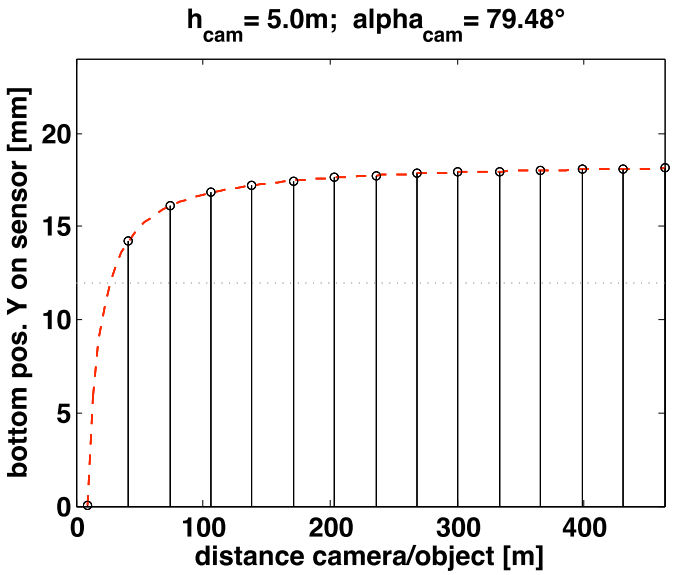
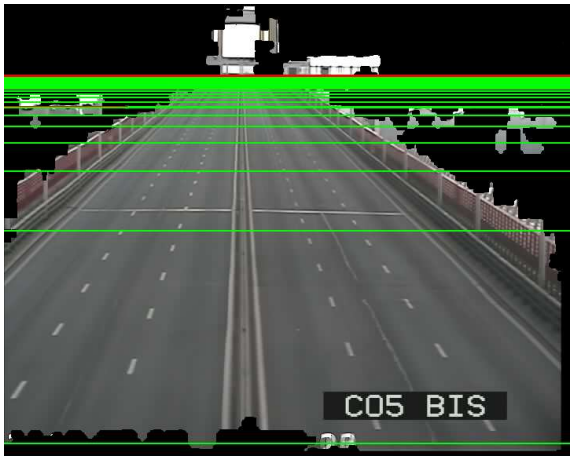
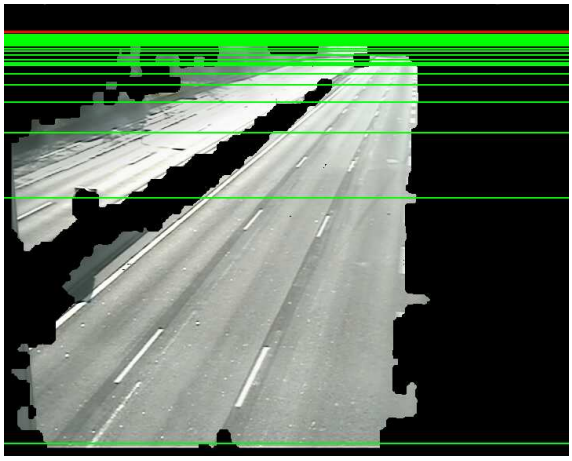


Figure 5.10: Determination of equidistant lines based on camera angle for *sequence 1*

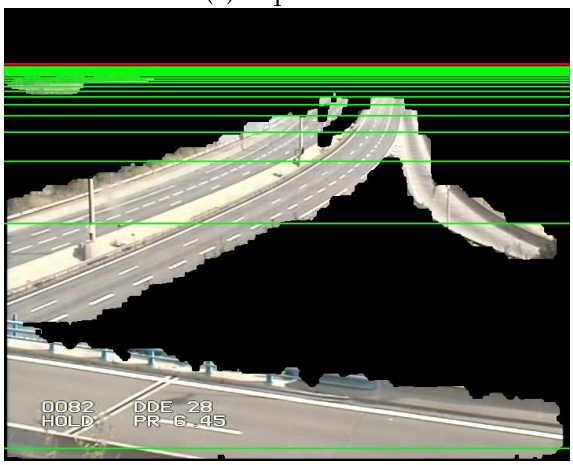
The green lines in Fig. 5.11 correspond to equidistant lines on the ground plane. They have been obtained through Eq. 5.1 at the estimated camera angle. An example for the calculation of these lines is given in Fig. 5.10 for sequence 1, where the distance mapping function for the given camera angle is sampled at evenly spaced positions. The parameter camera height can be set arbitrarily and does not influence the calculation, since it only rescales the x-axis but does not change the curvature of the function. Equidistant lines enhance two-dimensional image plane representation by a depth measure that can be used to assist object tracking and detection. If lane markers are visible, the lines can also be used to evaluate the quality of the camera angle estimation if no ground truth is available. Each segment between two equidistant lines should contain the same number of lane markings.



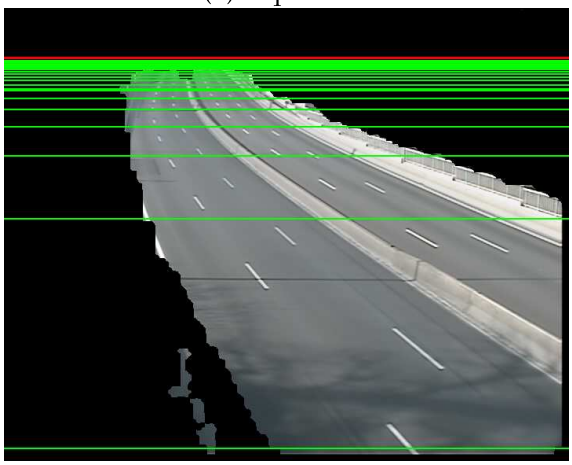
(a) sequence 1



(b) sequence 2



(c) sequence 5



(d) sequence 5

Figure 5.11: Compressed domain road segmentation results with equidistant lines in green

5.4 Summary and Conclusions

We proposed a simple, two-class camera pose classification scheme that uses only motion vectors extracted from H.264 streams. It it based on the presence of moving objects, where the temporal evolution of tracking results and the estimation of the relative depth serve as input. The vertical displacements of the object's top position, its centroid and ground contact point are analyzed and compared, which allows to draw conclusions about the basic camera pose. The algorithm robustly classifies the camera pose in ground or top view.

The idea was further extended to estimating the camera orientation angle by fitting a model function to the observations of the vertical position versus the estimated relative depth. Although being not very precise, the resulting angles represent rough approximations of the real orientations. The approach lead to the publication presented in [124].

For the specific use-case of highway surveillance, we presented an extension to the work presented in Chap. 4 and proposed simple yet effective methods to determine the vertical position of the horizon line and to segment the road portion in the image plane. Based on the position of the horizon line, a more robust approach to estimating the camera orientation was provided. The proposed method to determine the camera angle is not restricted to compressed domain processing. If image domain information is available, the horizontal position of the horizon line can for instance be determined by estimating the vanishing point.

Regarding future work, the unconstraint angle estimation algorithm may be refined in different ways. The relationship between the distance of an object and its Y-position in the image plane (see Equation 5.1) has to be corrected in the case of strong lens distortions resulting from wide angle lenses, which are frequently employed in surveillance cameras. Furthermore, the obtained geometrical information can be used to add robustness to the object detection and tracking module.

Chapter 6

Video copy detection

Content-based copy detection (CBCD) in video databases is an important and interesting research field. It differs from content-based video retrieval (CBVR) in that the goal is not to find *similar* videos but *exact* copies of the query video and transformations of it. The two major target applications of CBCD are video retrieval in databases and the protection of digital rights, where it represents an alternative to digital watermarking. In contrast to watermarking approaches, video copy detection regards the media itself as the watermark, where the task is to form unique video signatures that are robust to multiple types of transformations.

Large video collections in the form of local or web based archives necessitate efficient and scalable retrieval solutions. Scalability in the system context refers to the overall ability to cope with a varying number of managed content and/or users. In the algorithmic sense with respect to video retrieval, scalability refers to the ability to cope with videos of different spatial and temporal resolution in an efficient way. This aspect is becoming increasingly important regarding the continuously increasing variety of video distribution networks and devices, ranging from broadband High-Definition (HD) television to handheld devices. Techniques like scalable video coding (SVC) face this issue on the media side itself and will play an important role in the future media landscape.

A crucial aspect concerning the usability of the whole system is the computational complexity of the feature extraction process and the needed storage space for the obtained descriptors, most notably when dealing with large data collections.

Our aim is to efficiently extract lightweight video features that are temporally and spatially scalable and robust to multiple types of transformations. The compressed domain signatures presented in the following are either encoding- or motion-based, since in both cases, no or only little stream decoding is necessary. In general, signatures are supposed to be fast to obtain, easy to compare, small to store and robust to transformations. Although we concentrate on SVC compressed video, the proposed algorithms are also applicable for pure H.264/AVC streams.

The present chapter is organized as follows. An overview of previous work in this area is given in Sec. 6.1, followed by the introduction of the analyzed video features in Sec. 6.2. Before presenting the results of single feature copy detection in Sec. 6.4 we provide information about the used HD test data base in Sec. 6.3. Section 6.4 also includes studies on the scalability and codec interoperability of chosen features. How the detection results can be improved by combining multiple features is shown in Sec. 6.5. Finally, concluding remarks and some general system design recommendations are provided in Sec. 6.8.

Our main contributions to video copy detection are: i) The evaluation of compressed--domain features in comparison to popular image domain features in terms of copy detection performance on a SVC encoded data base, ii) a study on the scalability of different features, and iii) the proposal of search schemes that combine multiple compressed domain and pixel domain features in order to enhance the retrieval results.

In the following, the terms *feature*, *descriptor* and *signature* are used interchangeably.

6.1 Related Work

Concerning video copy detection and sequence matching, a number of previous efforts on CBCD have been published. They can be divided into two groups, pixel domain and compressed domain approaches. In both groups the extracted signatures are either of local or global nature. Local features only represent a small part of an image or video, while global features aim to represent the image/video as a whole.

In the **pixel domain**, color histograms [125, 126], feature points like SIFT [127] or Harris feature points [128, 129, 130] and object trajectories [131] are often considered features in CBCD. Indyk proposed to use shot boundaries to identify pirated videos in [132], which is a very lightweight descriptor but fails on short sequences. Mohan [133] adapted the ordinal measure for video retrieval applications. It is formed by comparing and sorting the mean brightness of defined regions in an image. It represents an image-based feature-vector that was later enhanced to the temporal case by Chen [134]. A comparative study on pixel domain CBCD has been published by Law-To et al. in [135]. The authors conclude that methods with local features have more computational costs but present robust results and that the ordinal temporal measure by Chen et al. [136] works very efficiently for minor transformations.

Regarding CBCD in the **compressed domain**, encoding data, motion information and transform coding coefficients build the basis for most frameworks, where motion delivers the most distinctive and important information. Hampapur et al. [137, 138] determine the dominant motion direction per frame through motion vector histograms. The correlation coefficient is used as similarity measure. Ardizzone et al. [139] base the search on the size

and the average motion of dominant regions, which are obtained by a sequential labeling method and clustering of the Motion Vectors (MVs). Kobla et al. [140] perform searches on a Global Motion Estimation (GME), which is determined by the largest bin in a directional MV histogram.

Naphade [141] proposes a MPEG-1/2 based video signature consisting of DCT coefficient histograms of the YUV components. The article focuses on computational efficiency and no indication is given regarding the performance on scaled or transformed versions. Taurun et al.[142] proposed an approximation of the color coherence vector (CCV) [143] by MPEG-1/2 DCT coefficients. Other approaches that rely on DC or AC block coefficients of MPEG-1/2 video are summarized in [144]. As stated before, transform coding coefficients are not available in H.264 without extensive decoding operations, hence only features that are available through stream parsing or that require only minimal decoding are considered in the following.

Babu et al. in [145] proposed a retrieval system for MPEG coded videos based on global motion and local object features. Global motion activity is characterized by the standard deviation of the magnitudes of MVs for each frame, and object segmentation is achieved through a combination of K-means clustering and the EM algorithm. Object trajectories are represented by two second order polynomials. Their system is designed to retrieve video sequences with similar local object trajectories and not for exact copy detection. Furthermore, the system does not capture complex camera motion and is not robust against transformations like rotation or flipping.

6.2 Features

This section presents the features we studied and shows how they are obtained. The order in which they are listed goes from low-level to high-level. Except from keyframe-based features, they are calculated frame by frame and are stacked into vectors along time. The longer these vectors are, the more unique and discriminative they become. The used similarity measure for sequence matching with each individual feature is also provided in this section. When comparing temporal feature vectors for two sequences of unequal length, we convolve the shorter vector with the longer one, calculate the similarity at each position and keep the highest similarity. Retrieval results of the different features are provided in Sec. 6.4. We also describe and provide results for features that turned out to deliver poor results for the sake of completeness and to demonstrate the limits of compressed domain video retrieval.

The presented compressed domain features are divided into three classes: encoding based, motion based and object based. Finally, we briefly review two popular keyframe

based approaches that are also evaluated for the task of video copy detection on the used video corpus.

6.2.1 Encoding based features

Bit Rate

The first video signature we analyze is the temporal evolution of the bit rate, i.e., the number of coded bits per frame (bpf) used by the encoder. This global low-level feature can be extracted very efficiently by simply parsing the compressed stream, so no decoding is necessary. For streams with spatial scalability, we extract the information for all layers. An example for a random test video is provided in Fig. 6.1. The Group-Of-Picture (GOP) size, i.e., the interval between two successive I-frames, becomes clearly visible when looking at the distance between two peaks, which correspond to I-frames that require more coded bits than inter-predicted B- or P-frames.

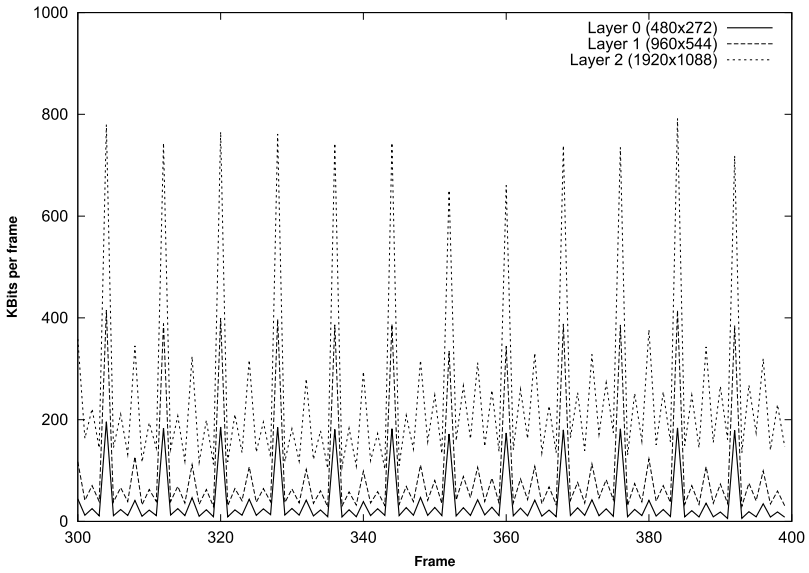


Figure 6.1: BPF per spatial layer of a test video with $GOP\_Size = 8$.

The periodic peaks caused by the GOP structure have to be compensated, because the absolute positions of I- and B-frames of corresponding images is probably different for two versions of the same video. We therefore average the BPF over each GOP.

The similarity S_{rate} between the length n feature vectors A and B of two sequences is represented by the correlation coefficient, given by

$$S_{rate} = r = \frac{\sum_{i=1}^{n}(A_i - \bar{A}) * (B_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \bar{A})^2} * \sqrt{\sum_{i=1}^{n}(B_i - \bar{B})^2}}, \tag{6.1}$$

with $\bar{A}$ and $\bar{B}$ denoting the mean values of vectors A and B, respectively.

$\oplus$ Easy and very fast to extract by parsing the stream. Lightweight: 1 integer per frame or per GOP, depending if values are averaged over GOPs.

$\ominus$ Codec- and implementation-dependent (see Sec. 6.7).

MB Partition Size Histograms

Beginning with H.264/AVC, macro-blocks (MBs) span 16x16 pixels and may be partitioned into smaller, independent sub-MB partitions as a function of the coding efficiency. In H.264/AVC and SVC, seven different MB partition sizes are possible: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 (see Chap. 2). Usually, MB partitions get smaller in well-textured and high-contrast areas that are in motion, like moving trees or object borders.

We construct and store frame-wise MB size histograms with seven bins, corresponding to the possible partition sizes listed above. The similarity S_{hist} between two videos is determined via the frame-wise sum of histogram intersections:

$$S_{hist} = \sum_{i=1}^{n} \sum_{k=1}^{bins} \min\left(h_A, h_B\right), \tag{6.2}$$

where h_A and h_B are the normalized histograms of the two input sequences A and B, n is the number of frames and $bins$ is the number of bins.

$\oplus$ Easy and fast to extract by parsing the compressed stream for macro-block and sub macro-block modes. Seven integers per frame have to be stored.

$\ominus$ Codec- and implementation-dependent (H.264/AVC and SVC only).

6.2.2 Motion Based Features

Since no color or pixel data is available in the compressed domain, motion is the most important information found in the stream. For block-based codecs from the MPEG family, motion is represented in the form of MVs that are associated with macroblocks (MB) in predicted frames of type B or P. I-frames are intra-coded, hence no MVs are available in the stream. As an approximation for I-frames, we use the mirrored LIST_0 MVs of the succeeding B-frame in coding order. As a reminder, LIST_0 contains MVs that point to past reference pictures, so in this case, they point to the preceding I-frame (see Chap. 2). We scale each MV by the distance to its reference frame in order to get vectors whose magnitudes are independent of the GOP structure.

Motion Activity

A very simple yet powerful global feature is the frame-wise, overall intensity of motion, sometimes referred to as pace of action. The feature we use is very similar to the MPEG-7 descriptor *Intensity of Motion I* [146, 147], given by the frame-wise, average MV magnitude:

$$I = \frac{1}{N} \sum_{i=1}^{N} \sqrt{dx_i^2 + dy_i^2}, \tag{6.3}$$

where N is the number of MVs per frame.

Different from MPEG-7, we weight the magnitudes of the MVs of a frame by a 2-D Gaussian to assign more importance to the center region. Motion vectors on the image borders are less reliable due to parts that are entering and leaving the image, usually resulting in random and noisy prediction vectors. Furthermore, the Region-of-Interest (ROI) is usually located in the image center. Since the MV magnitude depends on the temporal distance to the reference frame, we correct the GOP structure by dividing the MV magnitudes by the the distance to the respective reference frame.

The Gaussian weighting function is depicted in Fig. 6.2 and is given by

$$w(x,y) = e^{-\left(\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right)}, \tag{6.4}$$

where x_0 and y_0 denote the image center, $\sigma_x = w_I/2$ and $\sigma_y = h_I/2$, with w_I and h_I being the image width and height, respectively.

The frame-wise weighted motion activity MA thus is given as

$$MA = \frac{1}{N} \sum_{x=1}^{w_I} \sum_{y=1}^{h_I} w(x,y) \; ||MV(x,y)||. \tag{6.5}$$

The similarity between two sequences is determined by the correlation coefficient between two motion activity vectors (see Eq. 6.1).

$\oplus$ Easy and relatively fast to extract. Only the entropy coding has to be reversed in order to access the motion vector values. Can also be calculated for all other block-based video codecs and is robust to transformations like flipping and rotation.

$\ominus$ The query video has to be sufficiently long and contain motion for MA to be discriminative. Retrieves all sequences with similar motion, so the longer the descriptor, the more unique the motion fingerprint becomes.

Global Motion

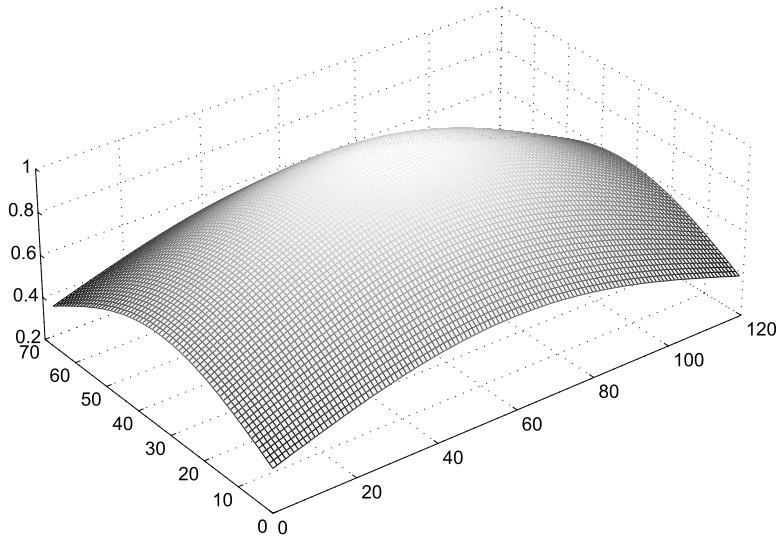


Figure 6.2: 2-D Gaussian weighting function for motion vector magnitudes.

Camera operation usually causes a global and dominant motion, which is an important feature in video indexing. In order to estimate the global scene motion, we use the algorithm presented in Sec. 3.2 to estimate the parameters $a_1, \ldots, a_6$ of the six-parameter affine motion model.

This multi-resolution approach directly exploits the spatial scalability of the stream. To save computing time, we only re-estimate the global motion until a certain threshold resolution is reached, which was empirically determined as 480x272 pixels. At higher spatial resolutions, the estimation results do not change significantly anymore (see Sec. 3.2 for more details).

The model parameters $a_1, \ldots, a_6$ are not robust to transformations like rotation or flipping per se. We derive the the two values m_{trans} and m_{ratio} from the model parameters to overcome this problem. m_{trans} corresponds to the magnitude of the total translational motion and m_{ratio} corresponds to the amount of zoom and rotation:

$$m_{trans} = \frac{|a_1|+|a_4|}{2}; \quad m_{ratio} = \frac{|a_2+a_6|+|a_5-a_3|}{2}. \tag{6.6}$$

These two values are robust to transformations like flipping and rotation and are stored for each frame of the video. Similarity is also calculated with the correlation coefficient between the vectors m_{trans} and m_{ratio} of two sequences.

The global motion estimation itself becomes less reliable when large, low-textured areas are present or when dominant motion is caused by a large object. Nevertheless, even if the estimation result does not reflect the real motion, it still forms a robust video descriptor. The estimation time depends heavily on the size of the estimation support, thus, the video di-

mensions.

⊕ Efficient estimation is possible and the features are robust to various transformations and distortions.

⊖ Query video has to be sufficiently long and contain global scene motion for GME to be discriminative.

Outlier Motion

Outlier motion is the complementary feature to global motion. During global motion estimation, all blocks that do not follow the dominant motion in the scene are marked as outliers. These outlier vectors are processed in exactly the same way as for global motion estimation (see above).

⊕ Captures object motion, which is most important for scenes with static cameras. Robust to various transformations and distortions.

⊖ Depends directly on the results of global motion estimation. In case the GME fails or no moving objects are present, outlier motion is not discriminative.

An example for all three motion based descriptors is shown in Fig. 6.3 for test sequence number 46 (see Sec. 6.3 for details on data base). It can be seen that the scene begins with high global and high local motion and calms down to the end. The translational global motion in this case results from camera panning and the outlier motion is caused by a pedestrian and moving tree branches. The outlier motion curve is not very stable because the estimation support, i.e., the number of MVs is small and due to noise introduced by the moving tree.

The unavoidable disadvantage of all motion-based descriptors is the fact that still scenes without significant global or object motion result in all-zero feature vectors. Regarding video retrieval in this case, we can only discard all sequences containing motion when performing a query.

6.2.3 Objects

Most notably for sequences that contain no global motion, moving objects can provide useful information for retrieval purposes. We detect and track objects in the compressed domain using the algorithms described in Sec. 3.3–3.4. As a result, we obtain the individual objects and their trajectories.

Number of Objects

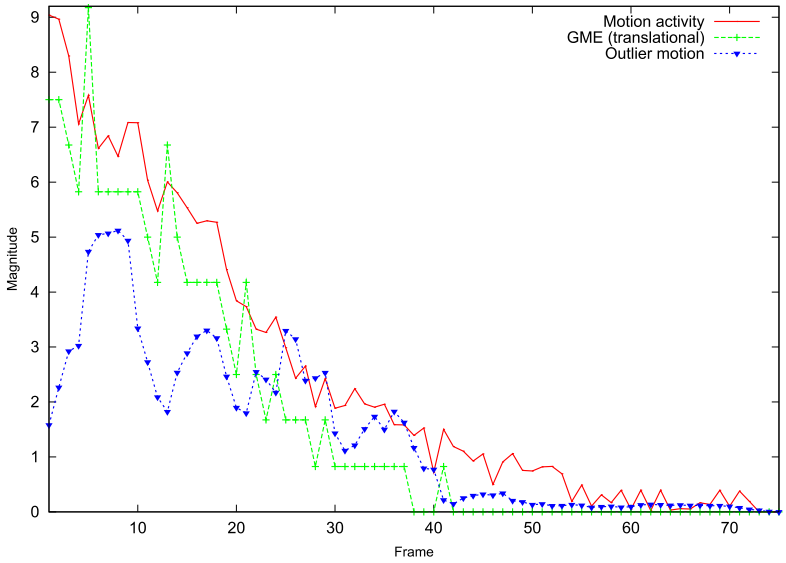


Figure 6.3: Example for motion based descriptors, extracted from test sequence number 46.

We discard any object that appears for less than 5 frames and store the remaining number of moving objects per frame as the first object based descriptor of the video sequence.

Trajectories

We represent object trajectories in a differential manner for retrieval tasks. Beginning with the second occurrence of an object, we store its speed, i.e., the distance the centroid travelled since the last frame, and the angle difference of the moving direction. Per video clip in the data base, we store as many differential trajectories as detected objects.

For comparing two sequences based on objects, we at first sum up the frame-wise difference in the number of detected objects and keep all sequences where the difference lies below a threshold. For all remaining sequences with a similar number of objects, we calculate the similarity between a query trajectory T_Q and data base trajectory T_{DB} by summing up the differences between the piece-wise differential trajectory vectors, given as the difference in speed Δv and the difference in angle $\Delta\alpha$, or in cartesian notation, $(dx, dy)^T$. We calculate the similarity for the temporal duration N of the query trajectory by

$$S_{traj} = \sum^{N} \left| \begin{pmatrix} v_Q \cos\alpha_Q \\ v_Q \sin\alpha_Q \end{pmatrix} - \begin{pmatrix} v_{DB} \cos\alpha_{DB} \\ v_{DB} \sin\alpha_{DB} \end{pmatrix} \right| = \sum^{N} \left| \begin{pmatrix} dx_Q \\ dy_Q \end{pmatrix} - \begin{pmatrix} dx_{DB} \\ dy_{DB} \end{pmatrix} \right| . \tag{6.7}$$

$\oplus$ High-level and thus robust to multiple transformations.

$\ominus$ For videos where the GME delivers erroneous results or for videos with many occluding objects, the number of objects and the resulting trajectories may be false.

$\ominus$ Short trajectories are often very similar, hence not very discriminative.

6.2.4 Keyframe Based Features

In order to evaluate the retrieval performance of the compressed domain features, we compare them with two common keyframe based methods: global 3-D color histograms and Lowe's popular local *scale-invariant feature transform* (SIFT) [106]. We chose these two features on the one hand due to their popularity and proven retrieval performance, on the other hand due to their robustness against affine transformations.

Color Histograms

We calculate global 3-D color histograms in the RGB color space for one representative keyframe of a video scene. In order to avoid full stream decoding, we extract the closest I-frame to the middle of the video clip. Per keyframe, a histogram with 125 bins (5 bins for each RGB channel) is constructed.

Histogram intersection is used as the similarity measure between two histograms.

$\oplus$ Fast to calculate and robust to various geometric transformations.

$\ominus$ Also retrieves different videos that are similar in color or shot in the same scenery. Prone to errors in case of large incrustations (logos, etc.).

SIFT

SIFT [106] describes local key-points that correspond to maxima or minima in Difference-of-Gaussians (DoG). The DoG is obtained by differencing multiple Gaussian-blurred versions of the original image at grayscale. For each detected key-point, its basic orientation is calculated. The descriptor vector itself is calculated on a 4x4 pixel region around the detected point and consists of a 128 bin 3-D histogram of gradient locations and orientations. In the comparative study presented in [107], SIFT proved to be the most robust local interest point descriptor.

As similarity measure S_{SIFT} between a query image I_Q and another image I_n, we use the ratio

$$S_{SIFT} = \frac{c \cdot \mathbf{NofMatches}_{(I_Q \cap I_n)}}{\mathbf{NofSIFTpoints}_{I_Q}}, \tag{6.8}$$

where $c = N_Q/N_n$ is a scaling factor, accounting for queries at different spatial resolutions. N_Q and N_n denote the number of pixels in the query image and the comparing data base keyframe, respectively. For feature matching itself, we use Lowe's *Best-Bin-First* (BBF) algorithm [148], a modification of the kd-tree algorithm to efficiently find the *approximate* nearest neighbors. Regarding future work, further improvements in matching time may be achieved by performing approximate nearest neighbor searches based on *Locality-Sensitive Hashing* (LSH) [149], as demonstrated by Auclair et al. in [150].

⊕ Invariant to a number of transformations like scaling, rotation and slight changes in viewpoint. It is a local descriptor, hence it is robust to post-production effects like cropping or logo insertions.

⊖ The feature calculation and matching process is computationally very expensive, most notably on high resolution frames. Furthermore, it is not robust to flipped versions of the original image.

6.3 Test Data Base

Before showing the retrieval results of the presented features, we provide information on the data set that was used for the experiments. It consists of a set of SVC-compressed, scalable high-definition videos. The corpus was created from 47 original video clips in Full-HD resolution (1920x1080) at 25 fps. Screenshots from all 47 base sequences are shown in Fig. A.1-A.2. For each of these 47 clips, 12 different versions are stored in the database:

- The original version (1920x1080),
- cropped from center to half resolution (960x540),
- resized to half resolution (960x540),
- resized to quarter resolution (480x270),
- flipped horizontally (1920x1080),
- flipped vertically (1920x1080),
- and six rotated versions (10°, 20°, 30°, 40°, 45°, 190°), continuously deformed to fit in the original 1920x1080 frame, where the emerging free space on the corners is filled with solid white.

Figure 6.4 shows sample screenshots of these transformations, taken from the base sequence number 16 (*Kung-Fu-2*).

The clips in the corpus are short at an average duration of 3 seconds. They contain four temporal and up to three spatial layers, depending on the resolution. The content of the different sequences greatly varies and includes indoor and outdoor shots with moving persons, objects and all types of camera motion. Figures A.3-A.4 show the translational global motion of each base clip in order to give the reader a better understanding of the used sequences. The shown global motion has been obtained with Motion2D [4].

Figure 6.4: Example of transformations per video clip present in the data base. Sequence number 16, *Kung-Fu-2* ©*Thomson*

Parameter	Value
GOP size	8 (fix)
GOP structure	IBBBBBBB
No. of ref. pictures per B-frame	2
Base layer mode	1 (AVC compatible)
Spatial layers at	1920x1080 ∣ 960x540 ∣ 480x270
Temporal layers at	25 fps ∣ 12.5 fps ∣ 6.25 fps ∣ 3.125 fps
Use of MCTF	no
Motion search	block search

Table 6.1: JSVM Encoder Settings

The data base was created in the course of the French national research project ICOS-HD, dealing with indexing of High-Definition (HD) content. More information on the corpus can be found on the official project website[1] [151]. For encoding, we used the SVC reference implementation JSVM in version 9.8, available at [19]. Some important encoding parameters that have been used for encoding are listed in Table 6.1.

Some of the clips in the corpus have been extracted from the same base video at different instances in time. Furthermore, some of the clips have been shot in the same environment with the same objects. For this work, the target application is the retrieval of exact video copies and transformations of the original clip.

[1]Menu: SPs → Sous-Projet 4 → Corpus HD (website in French)

6.4 Single Feature Detection Results

In this section we present the copy detection results with single feature queries, i.e., queries that are performed by using each of the presented descriptors alone. The results are measured in terms of precision and recall. Let TP denote the true positives of a query, FP the false positives and FN the false negatives. Precision and recall are then given by

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \tag{6.9}$$

Since the given task is to find copies of the query video and all transformed and distorted versions, recall equals 1 if all 12 versions of the clip are among the results, and precision equals 1 when all of the retrieved videos are correct matches. The task is not to find visually similar videos but copies of the exact query video, so the perfect system would retrieve exactly and only the 12 transformed versions, in which case both recall and precision equal 1.

6.4.1 Compressed Domain Feature Results

Figure 6.5 shows the precision-recall curves for each of the compressed domain descriptors presented in Sec. 6.2. Each precision/recall value pair has been obtained by averaging the retrieval results over all of the 47 original clips as queries. The curves represent the evolution of precision and recall at different threshold values of the similarity measure.

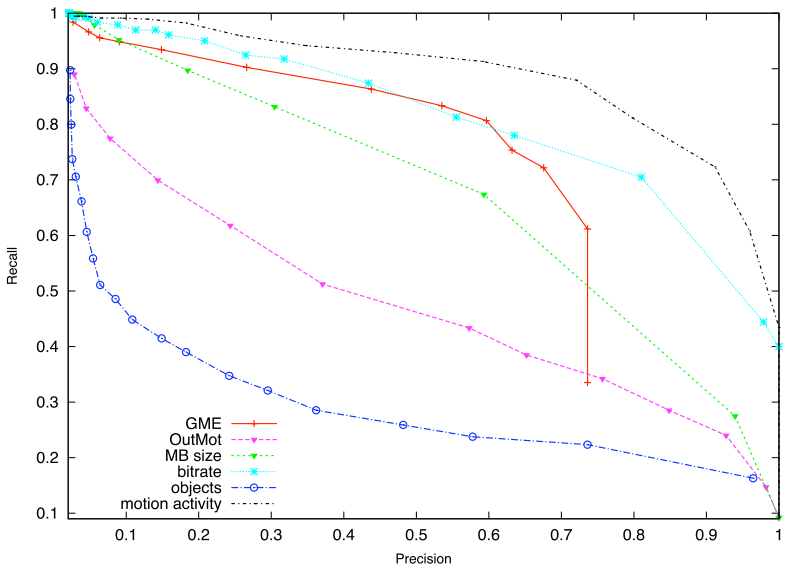


Figure 6.5: Single feature detection results: Compressed domain features

The performance of the local motion based features alone is poor on the given data set. The most high-level object feature – describing the number of objects and their **trajectories** – fails because the number of moving objects is very similar for most videos in the data base. Furthermore, the trajectory-based similarity has little discriminative power on the given data-set due to the short duration of the clips, resulting in short and near straight trajectories. Copy detection based on the object descriptor **outlier motion** shows increased performance because of its low level nature and because it can not be corrupted by erroneous tracking results. However, the main problem of the outlier motion descriptor is that it relies on the presence of moving objects and on the robustness of the outlier rejection scheme during global motion estimation.

The global encoding-based features both outperform the local object descriptors. With **macro-block size** histograms we achieve fair results, but it tends to retrieve a relatively high number of false positives and often misses geometric transformations like rotations or crops. This is due to the fact that the MB grid stays fixed regardless of the underlying content. Due to its limited performance and since this feature is H.264 specific and highly depends on the specific implementation, we do not study it further. Performance-wise more interesting is the **bit rate**. It can be extracted very easily and fast by parsing the video stream and achieves good retrieval results. Since the frame-wise bit rate over time is very similar for different clips, the correlation threshold (see Eq.6.1) that determines similarity has to be set close to 1. The main drawback, which is common to all encoding based features, is the dependance on the specific encoder implementation and the encoding parameters. If the entire test set is encoded in the same manner, the bit rate feature provides the best compromise between computational complexity and robustness. In Sec. 6.7 we provide a study of the interoperability of this feature between the two major coding standards MPEG-2 and H.264.

The retrieval performance of **global motion** (GME) is very good until it drops drastically with increasing precision. This is due to the fact that the precision-recall curves represent average values over all queries and not all clips contain global motion. If a query is performed with an all zero global motion vector, we can only discard any clips in the data base that do contain global motion but we can not make a further ranking between the remaining zero-motion clips.

With respect to single feature queries, the low-level **motion activity** clearly outperforms all other features for the task of video copy detection. Since no distinction is made between foreground and background motion, unlike global motion, it does not suffer from the constraints resulting from still cameras. The only case where this descriptor would fail completely is if neither any global nor any object motion is present, which is very rarely the case even for the short sequences in our test corpus.

6.4.2 Keyframe Feature Results

We calculated keyframe based features on one central key-frame of each sequence. In order to save decoding time, we extract the temporally most central I-frame of each sequence. Due to the given high resolution videos we only process one frame per sequence in order to limit the computational complexity, which is one of the drawbacks of SIFT. On a 1080p (1920x1080) screenshot of a well-textured scene, up to 15.000 interest points have been found, making it impractical for real-world use. A popular remedy is to down-scale the input images to reduce the number of features. For keyframe based extraction and matching, we processed all images at half of their original resolution for the sake of reduced computational complexity.

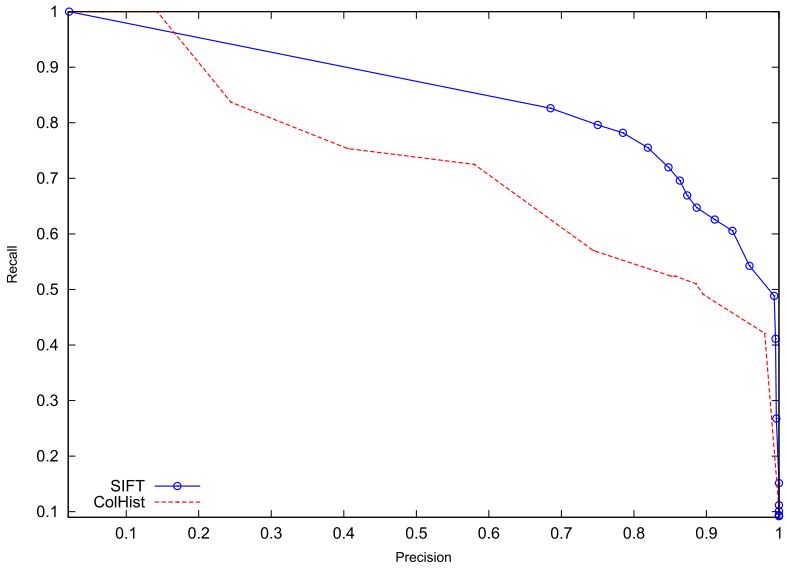


Figure 6.6: Single feature detection results: Keyframe approach

Figure 6.6 shows the performance of key-frame based retrieval on the same data set for SIFT [106] and 3-D color histograms in the RGB space. It can be observed that both approaches deliver reasonable results, but the local **SIFT** clearly outperforms the global color histograms. Nevertheless, even SIFT does not reach the performance of the motion activity feature on the given task.

This is due to the nature of the data base and the general properties of feature point / color based methods. Some of the clips are different, but have been shot in the same environment as others or with the same objects, so we obtain a significant amount of matches with sequences that are not a copy of the original query sequence. Furthermore, the data base also contains flipped versions, and SIFT fails in this case. The global color histograms suffer from similar problems and additionally failed on rotated versions, since the resulting gaps on the borders of the image after the rotation are filled with white, black or the border

pixel values, which becomes worse with increasing rotation angle. These rotated versions are also comparable to simple video-in-video type transformations and logo insertions.

6.4.3 Analysis of Retrieval Results

In this section we analyze and discuss the single feature retrieval results in more detail. The retrieval results for each base clip and for each specific transformation are provided, giving more insight in the strength and weaknesses of each descriptor.

Fig. 6.7 shows the retrieval results per transformation of chosen descriptors in terms of recall. The values represent average recall over all query videos and equal 1 if all specific transformed versions have been found for all query videos. Note that the plot only covers recall and aims at showing which descriptor is suited to retrieve which type of transformation.

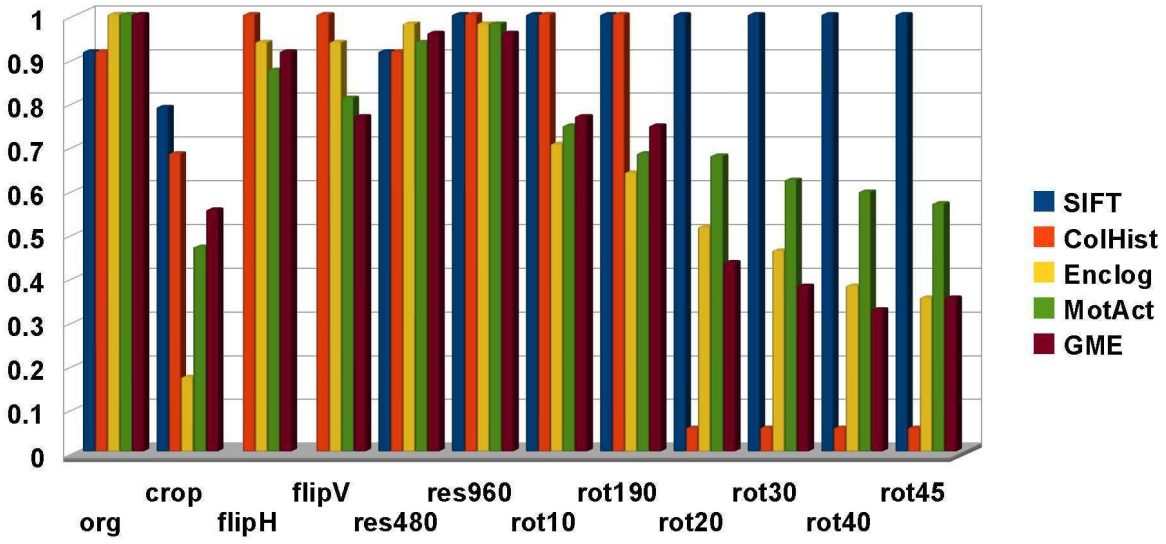


Figure 6.7: Average recall results for different transformations.

Except for the flipped versions, **SIFT** achieves the most consistent results over different type of transformations, with the highest recall values for cropped versions and near perfect results on rotation and scaling. However, it completely fails on both flipped versions, where all other features perform considerably well. One solution is to perform a second query with SIFT descriptors that have been obtained on the flipped input image, which however doubles the computational cost.

Color histograms also perform well on most transformations but suffer from severe problems on rotated versions. This is due to the high white ratio that is introduced in the image corners (see Fig. 6.4). In this particular case the problem could be overcome by ignoring

white values in the histograms, but since usually no a priori knowledge of the transformations is given, we did not correct the histograms. The problem is similar to large logo insertions or video-in-frame type transformations which also vastly affect the color distribution.

Bit rate performs well for all transformations where the visual appearance of the sequence is hardly affected (flipping, resizing), but average recall decreases continuously with changing visual content (rotations, cropping).

The same applies for the motion based descriptors **motion activity** and **global motion**, which deliver approximately similar average recalls, except for rotations where motion activity clearly outperforms GME. GME is slightly more robust on cropping, which can be explained by its more high-level nature.

To show how the retrieval quality depends on the actual video content, Figure 6.14 shows the true positive rates (recall) for each of the 47 query videos (for screenshots see Fig. A.1-A.2). It can be noticed that SIFT hardly depends on the visual content or nature of the sequence and achieves relatively constant recall values of about 0.8 over the whole range of queries. The recall for color histograms varies slightly more but still remains fairly stable. All analyzed compressed domain features show higher variability and lower consistency, which stems from the fact that they are temporal descriptors and mostly rely on frame-wise motion. Under unfavorable circumstances, i.e., no significant scene motion or object motion, the formed descriptors are not discriminative enough. This is the case for sequence number 27 and 34 for example.

Regarding recall only, keyframe based methods seem to be superior to compressed domain techniques with respect to copy detection. However, looking at the precision of the query results, the advantage of the presented compressed domain descriptors becomes visible, illustrated in Fig. 6.15, which shows the absolute number of false positives per query video. Both SIFT and color histograms retrieve a lot of clips that are visually similar but not copies of the query video. The problems of GME only based retrieval also becomes clearly visible. For all videos that contain no global motion, the retrieval results contain all other sequences without global motion in random order. Motion activity performs best and retrieves only few false positives, resulting in the overall highest average precision-recall curves.

6.5 Combined Feature System Design

Up to now, we only evaluated simple single feature queries. A more sophisticated way to perform queries is to combine multiple features with different properties to enhance the results. Ideally, the used features should be as complementary as possible to cover different types of transformations and the retrieval system should adapt to the specific properties of

the query video. A priori knowledge of the existing transformations in the data base can be used to influence the choice of descriptors. For example, if it can be assumed that no flipped video copies are present and retrieval time is not an important issue, standard SIFT [106] or SURF [109] will probably deliver very good recalls.

In the following we propose two different retrieval schemes that are tailored to the presented data set and the given transformations. The first scheme operates in the compressed domain only and aims at fast processing and retrieval. The second scheme is a combination of keyframe based features and temporal compressed domain features and targets high-precision applications. The general system design is depicted in Fig. 6.8. It is assumed that all features are pre-calculated and available for all videos in the data base.

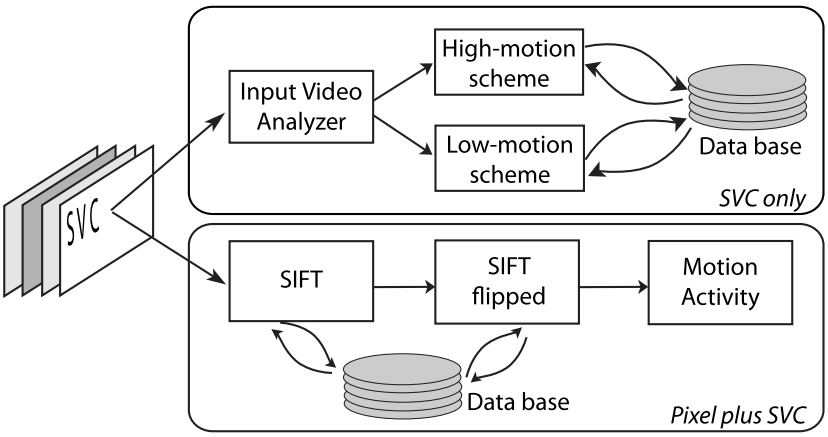


Figure 6.8: Building blocks of the system and overview of all proposed search schemes

The first step of each query is the feature extraction on the query video itself. The query module then dynamically choses a certain search scheme based on the properties of the input video and the user preferences. User preferences are the choice between the two main search schemes and a selection of transformations which the system should be able to cope with. The two search schemes we will introduce below consider each type of transformation that is present in the given test data base (see Sec. 6.3).

6.5.1 Compressed domain only scheme

For the detection of copies with compressed domain features only, we propose a dynamic and progressive search using a combination of different compressed domain features. This is motivated by the fact that motion based descriptors, most notably global motion, are not suited to retrieve clips that do not contain global motion. This becomes evident in Fig. 6.15. All sequences where the global motion descriptor retrieves a large number of false positives do not contain any global motion. However, for other sequences the number of false posi-

tives is close to zero. It is desirable to dynamically chose features that are better suited to the given query sequence.

As a function of the input video properties, one of two sub-schemes is chosen: the so-called high-motion or the low-motion scheme. If the mean value of the feature *global motion* (GME) is higher than a fixed threshold ρ_{GME}, the high-motion scheme applies. Otherwise, the low-motion scheme is used. The threshold is easy to determine since GME usually fails if global motion is constantly close to zero or zero, so the threshold has to be small. During our experiments, ρ_{GME} was set to an average motion of 8 pixels per frame (ppf) at full resolution (1920x1080) and is scaled according to the resolution of the query video. The exact value is not very critical and experiments showed that all values $4ppf < \rho_{GME} < 40ppf$ resulted in the same, correct classification results, i.e., all clips where GME retrieval fails (17-18, 23-24, 31-33, 38, 40-42, 44 and 47; see Fig. 6.15) are classified as low-motion.

High motion scheme:

1. Reject all clips classified as low motion
2. Refine the retrieved subset with GME and a low correlation threshold (≈ 0.5)
3. Further refinement with motion activity and a moderate correlation threshold (≈ 0.7)

Low motion scheme:

1. Reject all clips classified as high motion
2. Refine the retrieved subset with bit rate and a low correlation threshold (≈ 0.99[2])
3. Further refinement with motion activity and a moderate correlation threshold (≈ 0.7)

The precision-recall curve of this dynamic combination of compressed domain only features is shown in Fig. 6.9. On the given data set, a slight performance gain of about 6% in average can be achieved by using multiple features. Nevertheless, the complementarity of compressed domain (CD) features is limited and for certain sequences, all of them deliver poor results, e.g., for sequences number 27 and 34 of the data base (see Fig. 6.14) due to flat, low-textured backgrounds.

6.5.2 Pixel plus compressed domain scheme

For applications where high-precision is more important than computational complexity, we propose a combination of keyframe and compressed domain features for optimal performance. When a query is carried out, this scheme begins by extracting a first result subset

[2]0.99 may appear very high at first glance, but the bit rate feature is very similar for all sequences. The lowest correlation between two clips in our data set is about 0.97.

	SIFT ▷	flipped SIFT ▷	Mot.Act.
Prec	0.69	0.64	0.95
Recall	0.83	1.00	0.94

Table 6.2: Evolution of mean precision and recall at different stages of the combined scheme

that ideally contains all copies of the query sequence, i.e., recall equals one. A refinement run is then performed on this subset to filter out as many false positives as possible, while keeping the good matches.

Looking at Fig. 6.7 and 6.14, SIFT achieves the most stable and robust results regarding recall when flipped versions are not taken into account. In order to also obtain all potentially flipped versions, a second SIFT run is carried out with features that have been calculated on the flipped query image. Both SIFT result sets are joined together to form the first, high-recall subset. The high-recall at this stage comes at the price of lower precision, because SIFT usually retrieves some false sequences due to either miss matching or the presence of similar objects or background in other sequences. The similarity threshold for retrieval, i.e., the ratio of SIFT features in the query image and the number of matches with another keyframe (see Sec. 6.2.4) is set relatively low at 0.05 for both the regular and the flipped SIFT run. This way, very high recall is assured at a still reasonable amount of false positives.

The second stage consists of a refinement run on the high-recall subset, with the goal to keep all true positives while rejecting as many false positives as possible. Motion activity is chosen for this refinement run with a relatively low correlation threshold (0.6) due to its low false positive rates (see Fig. 6.15), but good precision. The refinement run is very fast because it works in the compressed domain and only a small subset of all stored videos has to be compared with the original query.

The evolution of mean precision and recall during this multi-stage retrieval is shown in Tab. 6.2. The given values have been obtained by setting the threshold on the similarity measure for SIFT and flipped SIFT to 0.05 and to 0.6 for motion activity. The first run with standard SIFT achieves a mean recall of 0.83, which increases to 1 when including the results from SIFT on the flipped version. The precision slightly decreases from 0.69 to 0.64 due to the introduction of some more false positives. After the refinement run with motion activity, we achieve near perfect results of 0.95 and 0.94 for precision and recall, respectively.

Fig. 6.9 shows the precision-recall curve of the proposed combined scheme when varying the threshold parameter of the refinement run. The curves for all involved single features are also shown for reference. We achieve near perfect results on the given data set.

To decrease the computation time at similar performance, SIFT can be replaced by *Speeded*

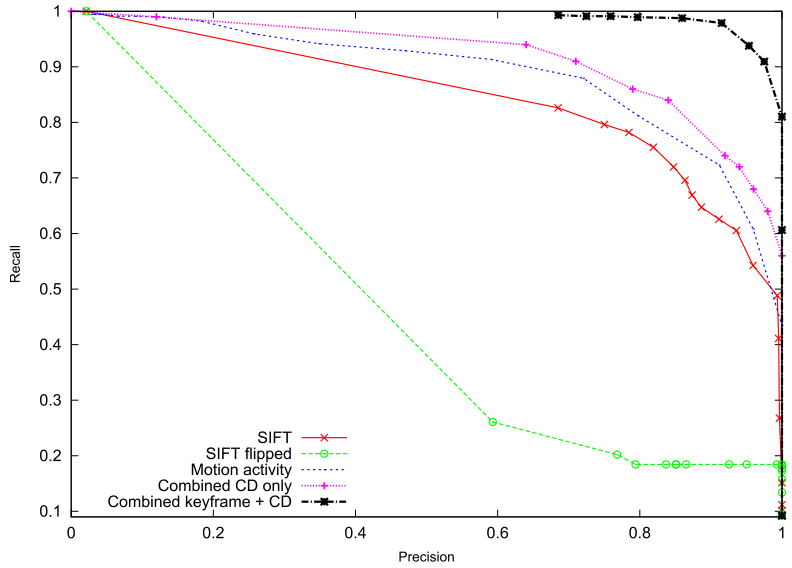


Figure 6.9: Precision-recall curves of all single features used in the combined scheme and curve of combined schemes (CD $\hat{=}$ Compressed domain)

Up Robust Features (SURF) [109]. During test runs with a standard implementation of SURF we achieved processing times of up to 7 times faster at similar results.

6.6 Feature Scalability

All presented compressed domain feature vectors are **temporally scalable** by a simple re-sampling process to the appropriate frame-rate. Temporal scalability in SVC is enabled by the *hierarchical prediction structure* of SVC. Lower temporal layers are obtained by simply discarding the last B-frames in coding order, cutting the frame-rate in half at each temporal level. Since the system relies on MVs, the lowest temporal layer which can be indexed is reached with a GOP structure where one predicted B- or P-frame is left between two intra-coded I-frames (IBIBI...).

Spatial scalability of the descriptors can be easily achieved by normalizing or scaling by the scale factor in size between the query video and the video to be compared. The features that are invariant to spatial scalability are the number of objects and the angle difference of the trajectories. An advantage of the correlation coefficient regarding scalability is the invariance with respect to scale changes of the feature vectors. Correlation is invariant to scalar multiplication, so spatial scalability is inherently given for all features where similarity is measured in terms of correlation.

The temporal and spatial scalability results for the best compressed domain descriptor - motion activity - are shown in Fig. 6.10. In order to demonstrate the spatial scalability,

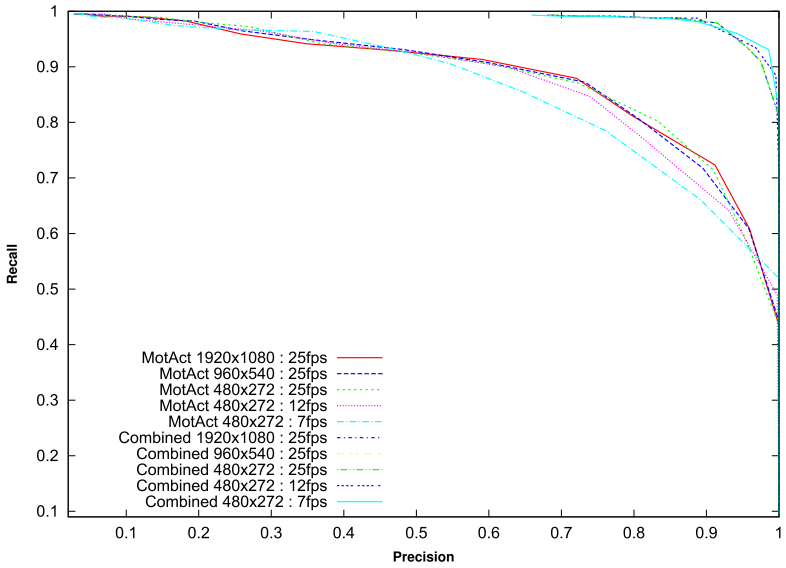


Figure 6.10: Spatial and temporal scalability of motion activity

queries have been performed with the original clip at full spatial and temporal resolution (1920x1080 @Â 25 fps), at full temporal and half spatial resolution (960x540 @ 25) and finally, full temporal and quarter spatial resolution (480x272 @ 25). For temporal scalability, we queried with descriptors that were calculated on the quarter resolution input video at half temporal resolution (480x270 @Â 12.5 fps) and at quarter temporal resolution (480x272 @ 7.25 fps), which also represents the last temporal layer that still contains predicted B-pictures with motion information.

For queries with lower spatial resolution and full temporal resolution, the retrieval results are almost identical, so spatial scalability in the tested range of resolutions (1920x1080 down to 480x270) does not affect the retrieval performance. Regarding temporal scalability, a slight performance decrease can be noticed, but considering the short duration of the sequences, the performance is still very good. The majority of sequences lasts only 3 seconds, so at 7 fps, the clip is entirely described with 10 integer values since after reduction, only every 2nd frame is a B-frame containing motion vectors.

Fig. 6.10 also shows the scalability of the second combined scheme (SIFT plus motion activity). The precision-recall curves have been obtained in a similar fashion as for the motion activity feature alone and clearly demonstrate the retrieval robustness towards spatially as well as temporally downscaled query videos.

6.7 Codec Interoperability

We performed all experiments on the H.264 encoded version of the data set, but since MPEG-2 is still in wide use, it is interesting to know if the presented compressed domain descriptors allow for cross-format queries. H.264 and MPEG-2 share the same basic coding principles. Images are cut into macro-blocks, block-based motion compensation takes place, transform coding is applied and entropy coding assures small file sizes. Since macro-block size histograms are not applicable to MPEG-2 and outlier motion did not perform very well, we focus on the two features bit rate and motion activity.

In order to evaluate the performance of cross-format queries, we created a second version of the whole corpus in MPEG-2. Like for the H.264 version, the GOP size was set to 8. A modified version of libmpeg2 [152] was used to extract the number of coded bits per frame and the motion vectors.

Bit rate

Before evaluating cross-format queries, we calculated the precision-recall curve for the bit rate descriptor on the pure MPEG-2 data base. Subsequently, we performed queries with H.264 bit rate vectors on the MPEG-2 data base. The resulting curves are depicted in Fig. 6.11.

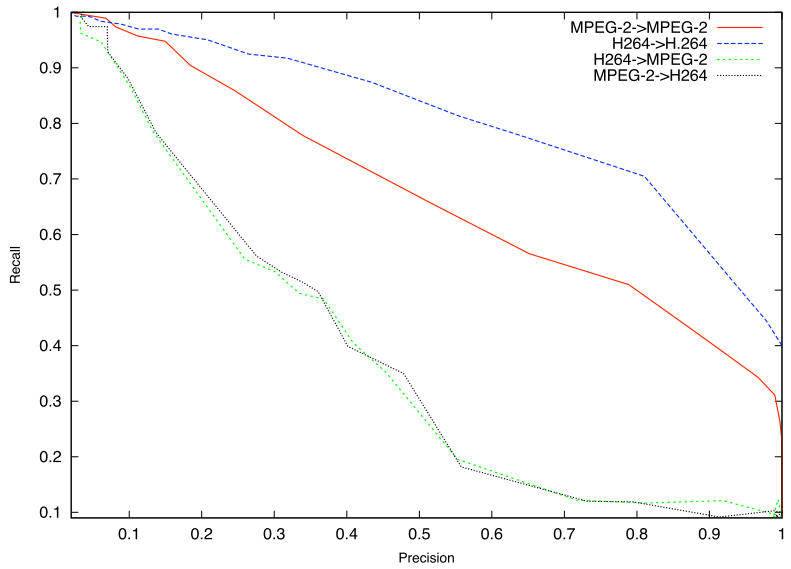


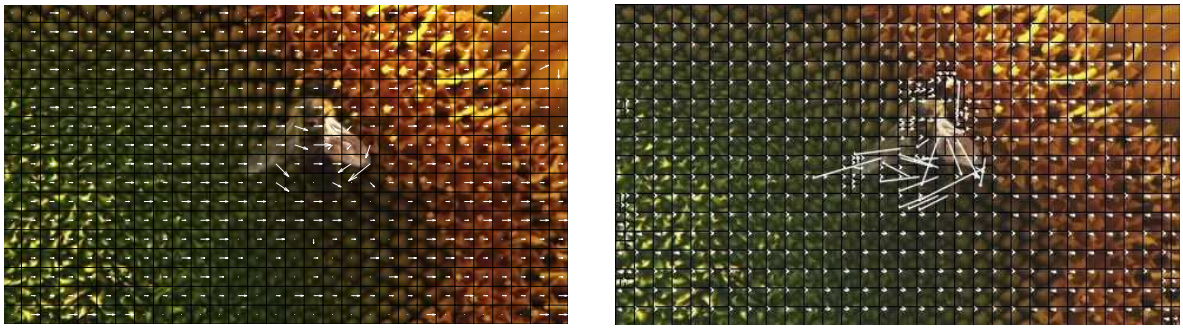
Figure 6.11: Comparison between bitrate feature in the MPEG-2 and H.264 domain

It can be noticed that the bit rate feature works better on the pure H.264 version than on MPEG-2. The reason for this behavior is that for MPEG-2, the amount of coded bits per frame depends less on the actual video content than in H.264, where more advanced pre-

diction and coding schemes are applied, resulting in better defined bit rates. Due to this property, cross-format queries on bit rate do not lead to satisfactory retrieval results. However, on pure H.264 data sets it performs considerably well and is extremely fast and easy to extract by parsing the bit stream.

Motion activity

Motion activity, i.e., the weighted sum of all motion vectors per predicted frame, can be computed for MPEG-2 in the same fashion as for H.264. The biggest difference concerning the two formats are the number of MVs per frame. Although the macro-block sizes of both standards are 16x16 pixels, a MB in H.264 may be cut down further in up to 16 sub-MB partitions. Each of these partitions has its proper MV assigned, so predicted H.264 frames usually contain more motion vectors as MPEG-2 streams, resulting in greater variability in the descriptor over time. An example of how the two standards process the same sequence is given in Fig. 6.12, which shows the MB grid and the associated MVs for the same frame of the MPEG-2 and the H.264 version of the clip.



(a) MPEG-2 (b) H.264

Figure 6.12: MB grid and MVs of the same frame from MPEG-2 and H.264/SVC. MVs are scaled for better visibility. Sequence *sunflower*

In order to obtain precision-recall curves of a motion activity based MPEG-2 only system and for cross-format queries, we proceed in the same fashion as for the bit rate feature. The results are provided in Fig. 6.13. We achieve the best performance on the MPEG-2 only data base. This is supposedly due to the coarser and evenly spaced MB grid of MPEG-2, whereas the MB configuration in H.264 is likely to change when coding transformed versions of the same video. The H.264 only system also delivers good results, but the retrieval quality drops significantly for cross-format queries. However, we achieve better results with cross-format queries using motion activity than with bit rate, e.g., average recall of about 0.5 at precision 0.7.

The two major reasons for the performance drop of cross-format queries are different implementations of the block matching algorithm in both standards and the sub-partitioning

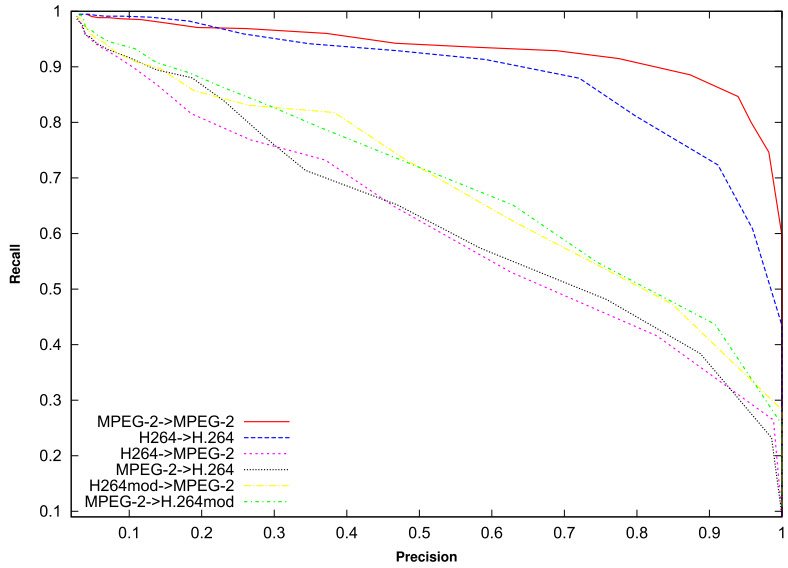


Figure 6.13: Comparison between motion activity feature in the MPEG-2 and H.264 domain

of MBs in H.264, which results in more MVs per frame on average. This leads to a motion amplification with respect to MPEG-2. For cross-format queries, we implemented a slightly modified version of the motion activity calculation for H.264. In order to simulate MPEG-2 behavior in H.264, we calculate an average MV for each 16x16 pixel MB. The cross-format queries on the given test set improve by up to 10 % due to this simple modification. The respective precision-recall curves are also provided in Fig. 6.13.

6.8 Summary and Conclusions

We proposed different compressed domain features and evaluated their performance in a video copy detection framework. Since neither pixel nor color information is available in the compressed domain, motion delivers the most significant and discriminant information. Encoding based features like bit rate over time or histograms of MB partition sizes deliver solid results but are largely codec dependent, what can pose problems in mixed format data bases. Among the analyzed descriptors, motion activity provides the overall most robust results for a variety of videos and transformations. However, a search that incorporates multiple features and that adapts to the properties of the query video can be applied to enhance the retrieval results. If high retrieval performance is needed in terms of precision and recall, a combination of keyframe based and compressed domain analysis is proposed, delivering near perfect results on the given data base.

We conclude with some general observations and system design recommendations regarding sequence matching in the compressed domain:

- If available, prior knowledge of the present transformations should be respected when designing a retrieval system.
- Regarding the detection of copies, compressed domain analysis allows for fast and robust analysis for a variety of videos. However, image domain methods are better suited for very static scenes and retrieval of *similar* videos.
- When using only a single video descriptor, motion activity is performing best for the task of video copy detection.
- For matching of 1-dimensional temporal features, the correlation coefficient is a simple yet robust similarity measure that inherently provides scalability in magnitude.
- For computational efficiency, down-sampling (or processing of lower spatial layers in the case of scalable content) may be performed to approximately CIF without affecting the overall retrieval results.
- A combination of local, key-frame based and temporal, motion based descriptors delivers the best results.

Our research on compressed domain video copy detection lead to the publication presented in [153]. For future work, we want to validate the results on larger video collections of scalable high-definition content, including more transformations like logo insertion and combinations of different transformations.

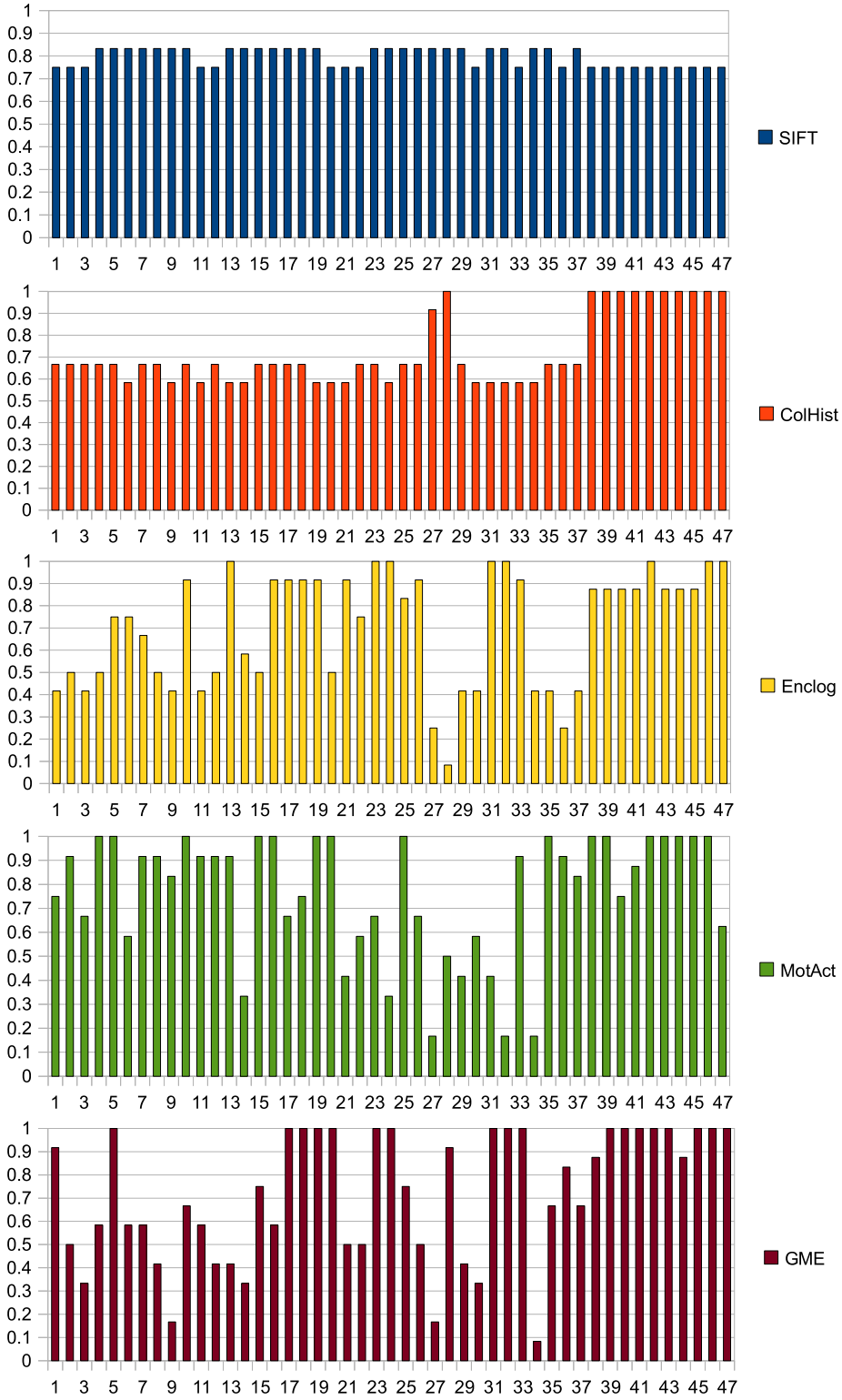


Figure 6.14: True positive rate per query clip.

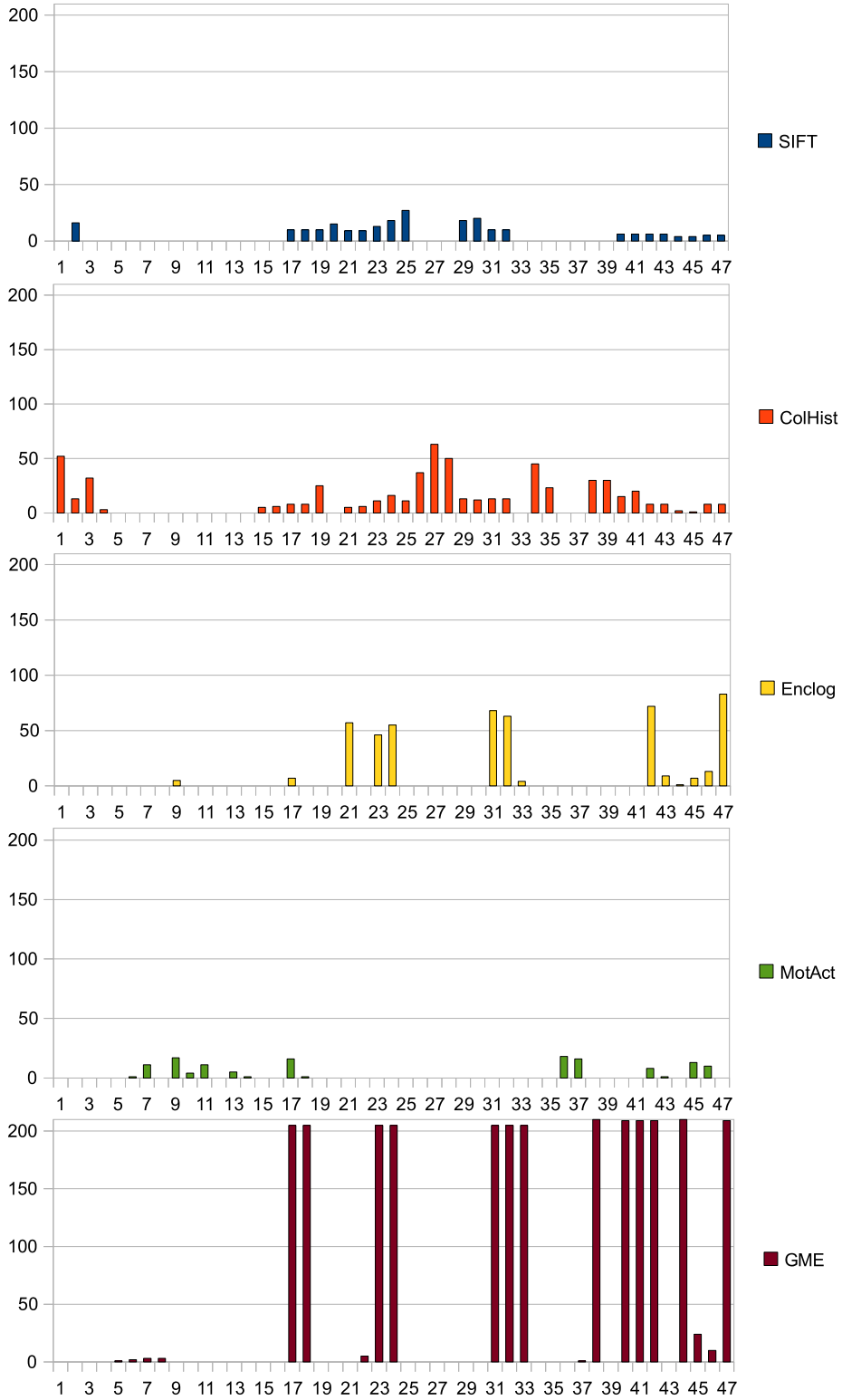


Figure 6.15: Absolute number of false positives per query clip.

Part III

Extensions and Conclusions

Chapter 7

Joint indexing/coding

The most valuable type of compressed domain information is motion in the form of block-based displacement vectors. As pointed out before, transform coding coefficients, which are often used in MPEG-2 video analysis, are not available in the H.264 domain due to spatial intra-prediction mechanisms.

Concerning H.264 compressed domain processing that relies on other information than motion, we showed in Chap. 6 how encoding based features like the frame-wise bit rate or macroblock partition size histograms can be exploited for video copy detection. Poppe et al. demonstrated in [53] that the number of coded bits per macroblock can be used to perform object detection and tracking in sequences with static cameras. However, motion is by far the most versatile information found in compressed streams. It can serve a variety of different purposes, including mosaic construction [37], camera motion estimation or object detection and tracking, as demonstrated in Chap. 3.

A limiting factor for the achievable performance of compressed domain analysis is the quality of the motion vectors (MVs). The MV fields found in the stream are optimized in terms of coding efficiency and do not necessarily represent the real motion in the scene. They are generated by a block-matching algorithm during encoding and can be regarded as a noisy and sparse version of the true optical flow. MV fields that contain severe noise can compromise the whole processing pipeline. Noise manifests most of all in vectors with random magnitude and orientation, most notably due to the following reasons:

- non-static, textured moving background regions like water or trees
- large, low-textured areas such as flat white walls or blue sky.

While the first problem also poses severe problems to image domain analysis using common techniques like background subtraction, the second problem is more pronounced in the compressed domain. Regarding MPEG-2 streams, a detection of problematic, low-textured

zones can be performed on DCT coefficients to assist scene segmentation and object detection [38]. When working with motion only, post-processing like spatio-temporal filtering of the MV fields can alleviate the impact of noise to a certain extent, but in situations where the MVs are too noisy and random, compressed domain algorithms fail.

Arguably the most obvious and general approach to compensate for rough and potentially erroneous compressed domain information is to partially or even fully switch to image domain processing. Examples of how information from both processing domains can be combined was shown in Chap. 6 in the context of a video copy detection framework, and in Chap. 4 regarding the analysis of traffic surveillance videos. While image domain processing increases the system robustness in most cases, the computational complexity on the analysis side also increases.

In this chapter, we present an approach to joint indexing and coding which shifts the increased computational complexity back to the encoder side by correcting the MVs from the original block-matching algorithm of H.264. The higher quality in terms of indexing and analysis of the new MV fields then allows for low-complexity, compressed domain only processing on the decoder/analysis side. The drawback is reduced coding efficiency of the resulting video streams as a trade-off for higher quality analysis results.

7.1 Related Work

Compressed domain analysis is mostly based on video streams that have been encoded with common block-based standards like MPEG-1/2 and H.264, which have been designed with respect to coding efficiency but not with indexing and vision tasks in mind. An approach to video coding that facilitates analysis is object-based video coding, where the coded entities are the visual or semantic objects in a scene rather than macroblocks.

Hakeem et al. [154] presented an object-based video coding framework for surveillance applications with static cameras. The method detects and tracks objects in the scene and learns the appearance model of each object using a principal component analysis (PCA). Each object is then coded using the coefficients of the most significant principal components of its learned appearance space. The rigid component of the object's motion is coded in terms of its affine parameters. Similar to the approach presented by Babu and Makur in [155], foreground segmentation is assumed to be given.

Other approaches include the one from Nishi and Fujiyoshi [156], which is not based on the coding of whole objects but on a pixel state analysis, or the method from Iglesias et al. [157] which represents an entire frame using its projection on the eigenspace computed from a reference frame. Schwartz et al. [158] proposed a hybrid approach that combines traditional, block-based MPEG-4 encoding with eigenspaces for the compression of fixed-view

surveillance videos. The eigenspaces are initially learned from a subset of sampled frames, pixels within a block are then projected to the corresponding eigenspace and the reprojection error is measured. If the reprojection error is high, meaning that the block is not modeled properly by the eigenspace, the block is encoded with traditional MPEG-4 mechanisms. To facilitate post-encoding vision tasks, the location of blocks containing moving objects is encoded, where the locations are obtained from the reprojection error of the eigenspaces.

MPEG-4/Part 2, also known as MPEG-4/Visual, provides mechanisms for object-based coding [159], but the standard had little success and H.264 became the de-facto industry norm due to its superior coding performance. We propose an approach to facilitate video analysis by modifying the original H.264 encoder's motion compensation (see Sec. 7.2). The resulting streams are still standard compliant, but enable fast analysis at moderate decreases in coding efficiency.

7.2 Motion Vector Correction

As stated before, the H.264 block-matching algorithm does not aim at finding motion vectors which represent the true optical flow between two frames, but instead searches for the displacement of a macroblock in a limited neighborhood which minimizes the prediction residue.

Our idea is to provide H.264 streams that are conform to the standard and still nearly as efficiently coded as the original ones, but with MV fields optimized for precise indexing tasks. To achieve this, we extend the H.264/SVC encoder implementation JSVM [19] with an additional reference motion estimator and decide on a MB basis if the original MV is kept or replaced. Figure 7.1 shows an overview of the experimental setup.

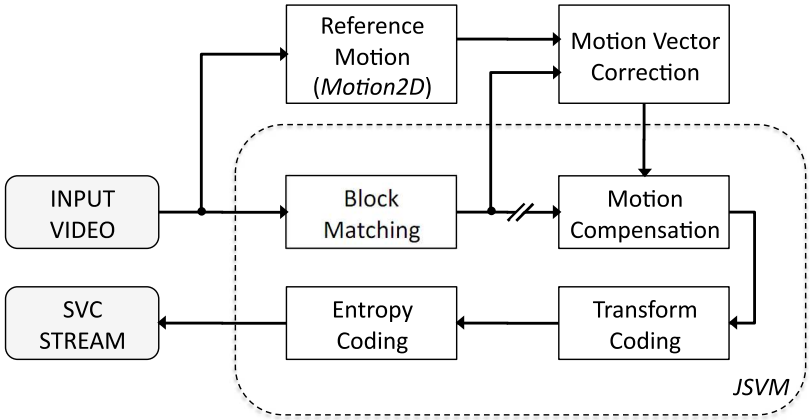


Figure 7.1: Motion vector correction scheme

The reference motion estimation is performed with Motion2D [4], a publicly available motion estimation library, which is applied to the input image sequence in coding order. Motion2D exploits a robust, multi-resolution and incremental estimation method exploiting only the spatio-temporal derivatives of the intensity function [36]. Output of this step are the frame-wise global motion parameters $\theta_{ref} = (a_1, \ldots, a_6)$ of the 6-parameter affine model, the weights W_{ref} which determine which pixel contributes to the global motion estimation, and the motion field MF_{ref} containing the reference displacement vectors per pixel.

The output of Motion2D is fed to the motion correction module, which is inserted between the motion estimation algorithm of H.264 and its motion compensation counterpart. Its second input are the original AVC/SVC motion vector fields from the block matching algorithm, denoted by MF_{org}. Global motion estimation like described in Sec. 3.2 is performed on these original motion fields, resulting in $\theta_{org} = (\hat{a}_1, \ldots, \hat{a}_6)$ and outlier masks in MB resolution.

The decision if the original H.264 motion field is changed is based on the summed, element-wise mean squared error (MSE) between θ_{ref} and θ_{org}:

$$\sum_{i=1}^{6}(a_i - \hat{a}_i)^2 \begin{cases} \geq \lambda_{GME} & \Rightarrow \text{ correct } MF_{org} \\ < \lambda_{GME} & \Rightarrow \text{ next frame} \end{cases} \tag{7.1}$$

where λ_{GME} is a user-defined threshold which controls the number of frames that will be corrected. If the MSE is below λ_{GME}, the original MVs are passed through, so the original stream is left untouched.

If it exceeds the threshold, a subset of all original MVs are corrected if their MSE with respect to the reference motion is higher than a second threshold λ_{MV}, which determines how many MBs per motion field will be changed. Denoting the components of motion vector $MV = (dx, dy)^T$, the decision rule per MB is given as

$$(dx_{ref} - dx_{org})^2 + (dy_{ref} - dy_{org})^2 \begin{cases} \geq \lambda_{MV} & \Rightarrow MV_{org} = MV_{ref}\,. \\ < \lambda_{MV} & \Rightarrow \text{ next MB}\,. \end{cases} \tag{7.2}$$

In the case of bi-predicted MBs with multiple reference frames and motion vectors, the decision to correct is made independently for each MV.

In order to limit the impact on coding efficiency and since the most problematic regions are low-textured areas, we only correct MVs which are considered as background. The pixel-wise weight images W_{ref} from the iterative reference motion estimator are used to decide if a macroblock is considered foreground or background. Examples for weight images are shown in Fig. 7.2, where white corresponds to a weight of 1 and black to a weight of 0. To determine if a MB is considered background, we calculate the mean value of the region in W_{ref} that is covered by the MB in question. If the mean value is greater than 0.5, we consider

the MB to be foreground and do not correct its motion prediction. As it can be seen in Fig. 7.2, low-textured parts within objects such as unicolor shirts are also considered as background and will thus be corrected if the reference motion significantly differs from the one obtained by block-matching.

(a) *redDress* (b) *camMotion* (c) *zoom*

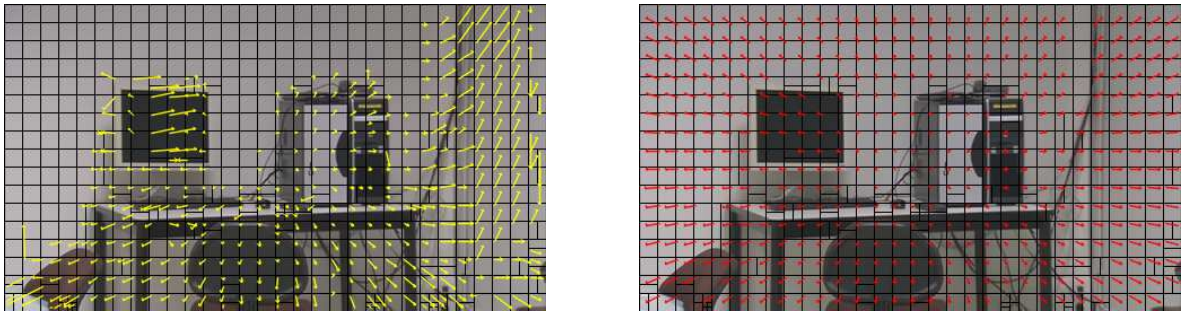
Figure 7.2: Example screenshots and pixel-wise, global motion estimation weight images W_{ref} for three test sequences

The new values for the corrected MVs are taken from the reference motion field MF_{ref}, which is defined at pixel level for background pixels only. It has to be noted that MF_{ref} is not calculated as the pixel-wise optical flow, but is reconstructed from $\theta_{ref} = (a_1, \ldots, a_6)$, so the reference displacement vector $MV_{ref} = (dx_{ref}, dy_{ref})$ at position (x, y) is given as

$$\begin{aligned} dx_{ref} &= a_1 + a_2 * (x - x_c) + a_3 * (y - y_c) \quad \text{and} \\ dy_{ref} &= a_4 + a_5 * (x - x_c) + a_6 * (y - y_c), \end{aligned} \tag{7.3}$$

where (x_c, y_c) denotes the image center coordinate. The corrected MV is thus calculated by Eq. 7.3 at the center of each macroblock.

If a motion vector is corrected, the rest of the H.264 processing chain is not affected, i.e., motion compensation is performed as usual. The prediction residual after compensation with the corrected MV is usually greater than the original one, so the coding efficiency is decreased. An example for an original H.264 motion field in a problematic scene and the corrected version is shown in Fig. 7.3. The scene contains large regions showing white walls, which have been originally coded in SKIP mode with zero motion. While this behavior increases coding efficiency, it poses sever problems to compressed domain analysis. The reference field shown in Fig. 7.3b enables accurate global motion estimation, which builds the basis for many further vision tasks such as object detection.



(a) *Original MV field* (b) *Reference motion*

Figure 7.3: Original H.264 MV field in comparison with reference displacement vectors. Sequence *camMotion* during zoom out

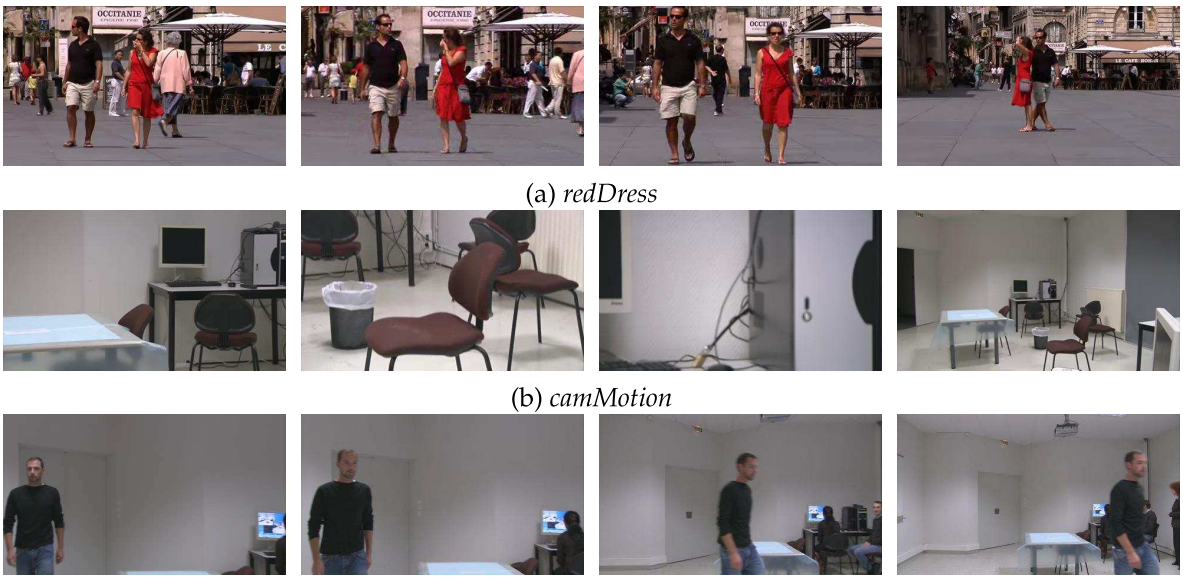
The mentioned thresholds λ_{GME} and λ_{MV} have to be chosen so that a minimal decrease in efficiency leads to indexing gains defined by the target application. Here, we consider global motion estimation and the detection of the presence of camera operations like panning, tilting and zooming as the target application because the quality of the results largely affect the performance of further vision tasks. For the time being, we set both thresholds manually (see next section).

7.3 Results

In this section, we present the global motion estimation (GME) results for the corrected streams in comparison to those obtained by the original streams and to those obtained by Motion2D [4] on the raw sequence. We show the results for three representative test sequences, *redDress*, *camMotion* and *personZoom*, which are part of the high-definition LaBRI video corpus and contain neither cuts nor transitions. Screenshots of the three sequences are shown in Fig. 7.4.

7.3.1 Example 1 - *redDress*

The sequence *redDress* is taken as example for standard situations. It was shot outdoor, is well textured and contrasted in most regions and contains camera panning, tilting and zooming. Although the scene shows multiple moving objects, the background remains the dominating region. In scenarios similar to this, compressed domain processing is likely to deliver robust and reliable results. Figure 7.5 shows the global motion estimation results for the sequence *redDress*, obtained by analyzing the raw video with Motion2D (Fig. 7.5a) and by analyzing the original, uncorrected SVC base layer motion vectors (Fig. 7.5b). The



(a) *redDress*

(b) *camMotion*

(c) *personZoom*

Figure 7.4: Example sequences for joint indexing and coding approach

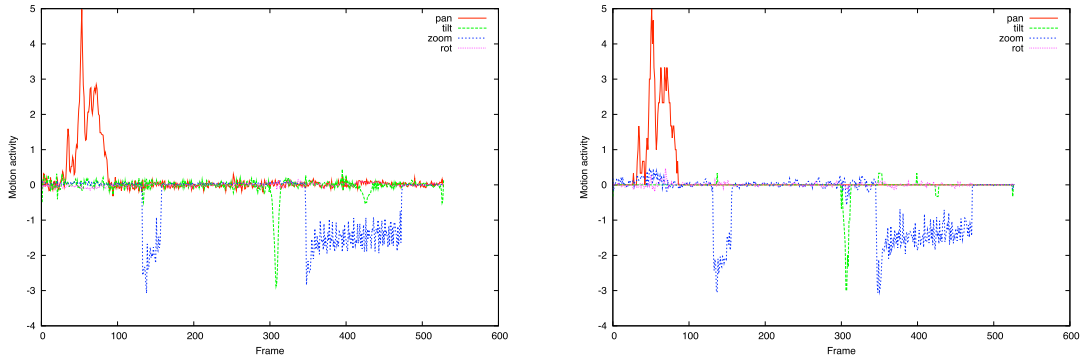
compressed domain results are similar to those in the image domain and no correction of the MV fields is necessary.

The fact that the compressed domain GME algorithm is delivering robust results implies that the H.264 motion vectors approximate the true optical flow well and that outlier blocks due to noise and moving objects are successfully rejected. Figure 7.6 on the left shows an example of an outlier mask, where MB that are excluded from the GME support are framed in red. Although the mask is not perfect, it manages to capture most object blocks in a difficult zooming situation and with object motion that is less pronounced since they are moving towards the camera. Spatio-temporal filtering of the outlier masks, as shown on the right, alleviates the majority of miss-detections and allows for further processing like unsupervised object detection and tracking.

7.3.2 Example 2 - *camMotion*

We created sequence *camMotion* to test the compressed domain detection of different types of camera operation in low-textured environments. It contains tilting/panning in different directions, zooming out as well as zooming in. At times the camera is out of focus, resulting in low-contrast images.

The GME results for this sequence are depicted in Figure 7.7. Here, the detection results obtained with the original SVC stream are partly degraded, mainly due to the presence of large, flat zones in the image, resulting in very noisy and arbitrary MV fields for certain frames. To enhance indexing results, the MVs of these frames have been corrected. The



(a) Image domain (Motion2D) (b) Compressed domain

Figure 7.5: Motion estimation results for sequence *redDress*

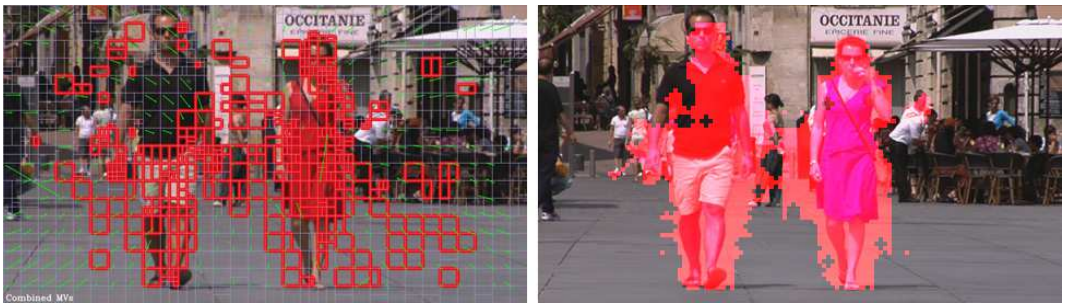
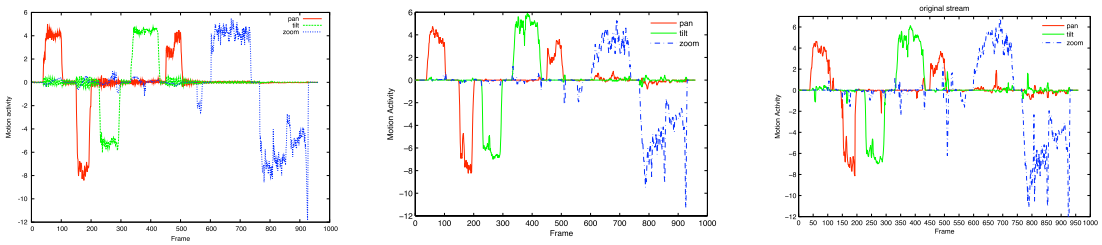


Figure 7.6: Raw (left) and filtered (right) outlier mask of frame 134 of the sequence *redDress*.

threshold λ_{MV} was set to 15 and λ_{GME} was varied between (A) 2 and (B) 1. For $\lambda_{GME} = 2$, this caused the correction of 5 percent of all frames (48 frames), and per modified frame an average number of 46 MBs, representing on average 30 percent of the estimation support and affecting about 23 percent of all pixels. For $\lambda_{GME} = 1$, the motion vector fields of 9 percent of all frames had been corrected. The average luminance PSNR of both decoded, corrected sequences stayed nearly the same (-0.2 dB), at an increase in bit rate of 1.7 percent for $\lambda_{GME} = 2$ and of 3.1 percent for $\lambda_{GME} = 1$.



(a) Motion2D (b) Corrected SVC (c) Original SVC

Figure 7.7: Motion Analysis with (a) Motion2D [4] of raw video at base layer resolution, (b) of the corrected SVC stream with $\lambda_{MV} = 15$, $\lambda_{GME} = 1$ and (c) of the original SVC stream.

	Orig.	Corr. (A)	Corr. (B)	Motion2D
Recall	0.90	0.93	0.96	0.98
Precision	0.83	0.92	0.94	0.97

Table 7.1: Results for recall and precision at $\lambda_{det} = \pm 0.7$. Sequence *camMotion*.

Figure 7.8 shows the qualitative global motion detection results of all three approaches. The detection threshold λ_{det} for significant motion was empirically determined and set to ± 0.7 for all 3 approaches. Below this threshold, the variations of the parameters a_1 to a_6 is considered as noise. λ_{det} is resolution dependent and has to be scaled with resolution. The recall, i.e., the percentage of all frames that really contain camera movement and that are correctly classified by the system was improved by 6 percent. More notably, the proposed correction mechanism reduced the number of false detections, i.e., improved the precision by 11 percent. Values for precision and recall are summarized in Table 7.1 considering the camera motion classification results for each frame. The ground truth has been obtained manually.

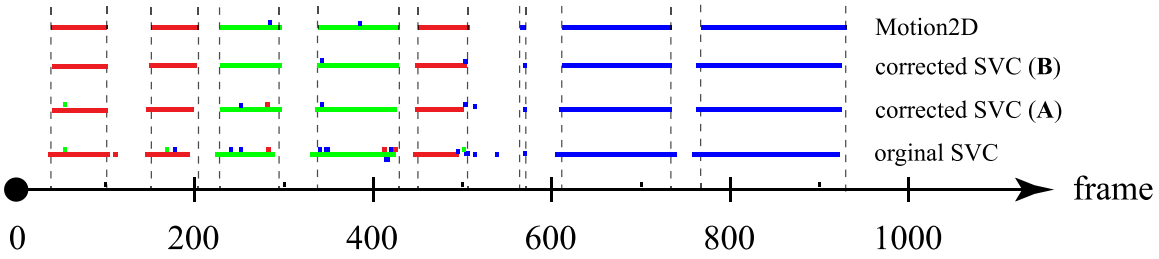


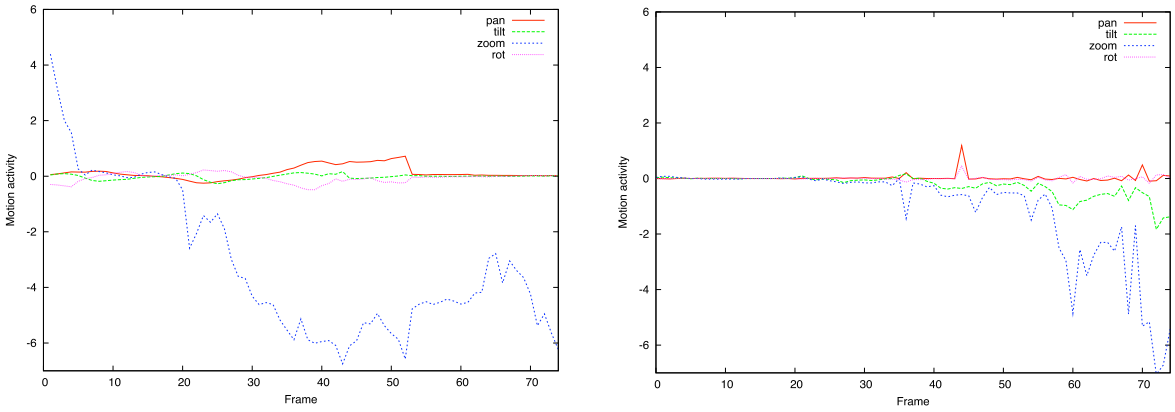
Figure 7.8: Detection results for fixed $\lambda_{det} = \pm 0.7$ and $\lambda_{MV} = 15$. (A) $\lambda_{GME} = 2$ (B) $\lambda_{GME} = 1$. The dotted lines depict the limits of the manually determined ground truth. Sequence *camMotion*

The obtained results after correction nearly reach the performance of Motion2D. In particular, the indexing results of frames with large, low-textured areas could be improved, which are considered as the most problematic ones [26], [160].

7.3.3 Example 3 - *personZoom*

The third example demonstrates the motion vector correction on a test sequence which was shot in a very low-contrast, low-texture environment with a moving object and camera operation. A short zoom-in is performed on a person who is walking towards the camera, followed by a long zoom-out while the person is turning slightly to the right. The simplicity of the background allows for efficient coding with mostly zero-motion or SKIP blocks on the flat white walls. However, camera motion estimation and object detection in the compressed domain becomes difficult due to the very sparse motion density despite the presence of strong scene motion.

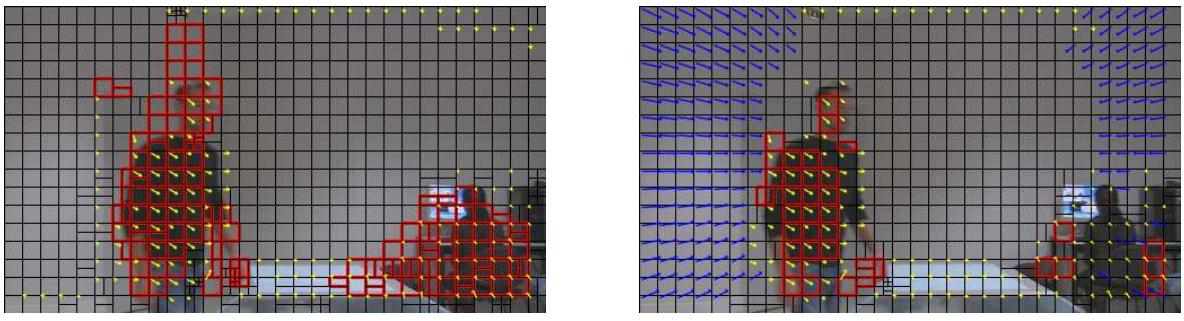
Figure 7.9 shows the global motion estimation results with Motion2D in the image domain together with the compressed domain results. In contrast to the image domain results, the short zoom-in is completely missed in the compressed domain, and the long zoom out is detected only during the last third of its duration. This leads to the correction of 53 percent of all motion fields, which in turn causes a large bit rate increase of about 16 percent.



(a) Motion2D (b) Compressed domain GME

Figure 7.9: Global motion estimation results on raw image sequence with Motion2D and in compressed domain. Sequence *personZoom*

The effect of the motion correction on automatic object detection is illustrated in Fig. 7.10, which shows original and corrected MVs at a moment of strong zoom-out. In the original H.264 stream, motion vectors are mostly present on well-textured areas. The white background regions show zero motion, which leads to the classification of nearly all non-zero vectors as outliers. After motion correction, the refined outlier masks mostly cover only real objects in motion. As it can be noticed in Fig. 7.10b, at the given value of $\lambda_{MV} = 15$, only macroblocks on both horizontal border regions have been corrected because zooming is most pronounced in these areas.



(a) Original H.264 MVs (b) Corrected MVs

Figure 7.10: Example of motion vector correction. Original H.264 MVs are drawn in yellow, corrected vectors in blue. Outlier blocks are framed in red. Sequence *personZoom*

7.4 Summary and Conclusions

Concerning traditional image domain processing, the presence of large low-textured areas in the scene usually facilitates vision tasks. In the compressed domain however, this results in corrupted motion vector fields, most notably in moments containing camera operations like panning or zooming. Hence, global motion estimates which are for instance necessary to perform motion compensation become erroneous and the whole processing chain is affected. The main reason for noisy motion vector fields is the block matching algorithm which is employed in standards like H.264.

In order to enable efficient compressed domain processing in such scenarios, we proposed an extension of the H.264 motion estimation module. Parallel to traditional block matching, we perform robust global motion estimation on the raw input sequence [36, 4] during encoding and decide for each frame independently if the original motion vector field is regarded as precise enough to allow for successful compressed domain analysis. If the results from both processing domains differ significantly, a subset of the original motion vectors is corrected according to the reference motion. We demonstrated by means of examples how the quality of global motion estimates as well as object detection results in the compressed domain can be improved through motion correction. The proposed approach lead to the publication in [161].

The advantages of the proposed approach are that the resulting streams are still standard compliant and that existing compressed domain algorithms can be applied without modification at better results. The major drawbacks are the increased computational complexity on encoder side and the decreased coding efficiency, since the new motion vectors usually cause higher prediction residuals. Possible application scenarios for the proposed method are closed systems such as surveillance networks, where the provided analysis and indexing gains outweigh the decreased coding efficiency.

A number of extensions and modifications of the proposed method can be considered for future work. Ideas include to incorporate image domain segmentation and texture analysis algorithms to identify problematic regions and to replace the reference motion estimator Motion2D with true optical flow techniques. Besides the modification of the motion fields, another possibility would be to take advantage of the fact that H.264 uses different block sizes. A pre-defined block coding mode may for instance be reserved for object borders, and the compressed domain extraction of object silhouettes could be performed in a straight-forward way by stream parsing. The negative effect on the coding efficiency would be negligible, but the position of object borders had to be determined beforehand in the image domain.

Chapter 8

Summary and Conclusions

In this thesis, we presented efficient methods to index and analyze single- or multi-layer H.264/AVC or SVC streams without a full decoding to the pixel level. The most valuable information found in the stream is motion in the form of block-based displacement vectors, which become accessible after reversing the entropy coding.

The extracted motion vectors can be used to robustly estimate global scene motion, which often originates from camera operations like panning or tilting. This step is necessary if the target application shall not be limited to fixed view sequences. Furthermore, the robust estimation scheme with iterative rejection delivers outlier masks as a by-product, providing a rough segmentation of the scene background and moving foreground objects. Due to its proven robustness, we ported an existing global motion estimation algorithm to the H.264/SVC domain, which can also handle standard AVC streams without further modification.

Based on the obtained outlier masks and the motion vectors of foreground regions, we developed novel methods for detecting and tracking moving objects in the image plane seen by the camera. The most difficult tracking situations arise from multiple occluding objects, resulting in merged silhouettes in the binary mask image. In a general approach not trained to a specific type of object and where non-rigid object may appear, merged object masks cannot be separated by looking at a single instant in time. We proposed a temporal analysis and the construction of object energy images as dynamic object models to resolve merged mask situations. The image plane tracking results are illustrated by 2D+t representations, which show the object trajectories together with the camera motion over time for a given video sequence.

From the temporal evolution of the object silhouette dimensions, we presented an algorithm to estimate the relative distance of moving objects to the camera. The computed depth information allows us to construct pseudo 3D representations of the object trajectories or, via projection, the construction of 2D ground plane trajectories. Based on single-view ob-

ject tracking results, we proposed a new approach to estimate the camera orientation angle, which is often important information towards a better understanding of the scene.

We demonstrated how real-world applications benefit from the presented techniques using examples from the domains of traffic surveillance and video copy detection. We also presented the limitations of pure compressed domain processing in both scenarios, leading to processing schemes that combine image domain techniques with compressed domain motion information. All of the presented methods require no manual training, can handle camera motion and multiple objects, are fully unsupervised and allow computationally efficient analysis.

With the recent development of video coding standards, we believe that the presented techniques will still be relevant in many years to come. The large-scale deployment of H.264/AVC and SVC continues, and the potential future recommendations, like High-Performance Video Coding (HVC) discussed by the MPEG[1], or H.265/Next-generation Video Coding (NGVC) discussed by the ITU-T VCEG[2], will most likely be based on the existing H.264 standard.

8.1 Future Work

Unsupervised video indexing and analysis is a challenging area of research with many unsolved problems. Large-scale applications like video retrieval or surveillance require robust and efficient processing schemes, and analysis in the compressed domain offers great possibilities to achieve this goal. Considering the scope of this thesis, we consider the following problems as the most promising future directions:

- The improvement and combination of existing compressed domain algorithms. Up to now, most approaches are optimized for a specific application environment for which they deliver good results but fail in others. The combination of multiple techniques and dynamically switching between them as a function of the current conditions could enable more general solutions. As an example, object detection in static scenes can be efficiently performed on the number of coded bits per macroblock, while object detection based on global motion estimation could take over in case of camera motion.

- The further study of hybrid image/compressed domain approaches. Robust pixel-level techniques can be used on sporadically decoded frames to initialize, assist or correct compressed domain algorithms.

[1]Motion Picture Experts Group: `http://www.mpeg.org/`
[2]ITU-T Video Coding Experts Group: `http://www.itu.int/ITU-T/`

- We presented methods to estimate elements of the basic scene geometry like the orientation angle of the camera or the position of the horizon line. We have not yet exploited these results. Tasks like object detection, tracking and classification can greatly benefit from this information. Examples include the correction of estimated object velocities and dimensions in a classification framework.

Appendix A

Additional Images

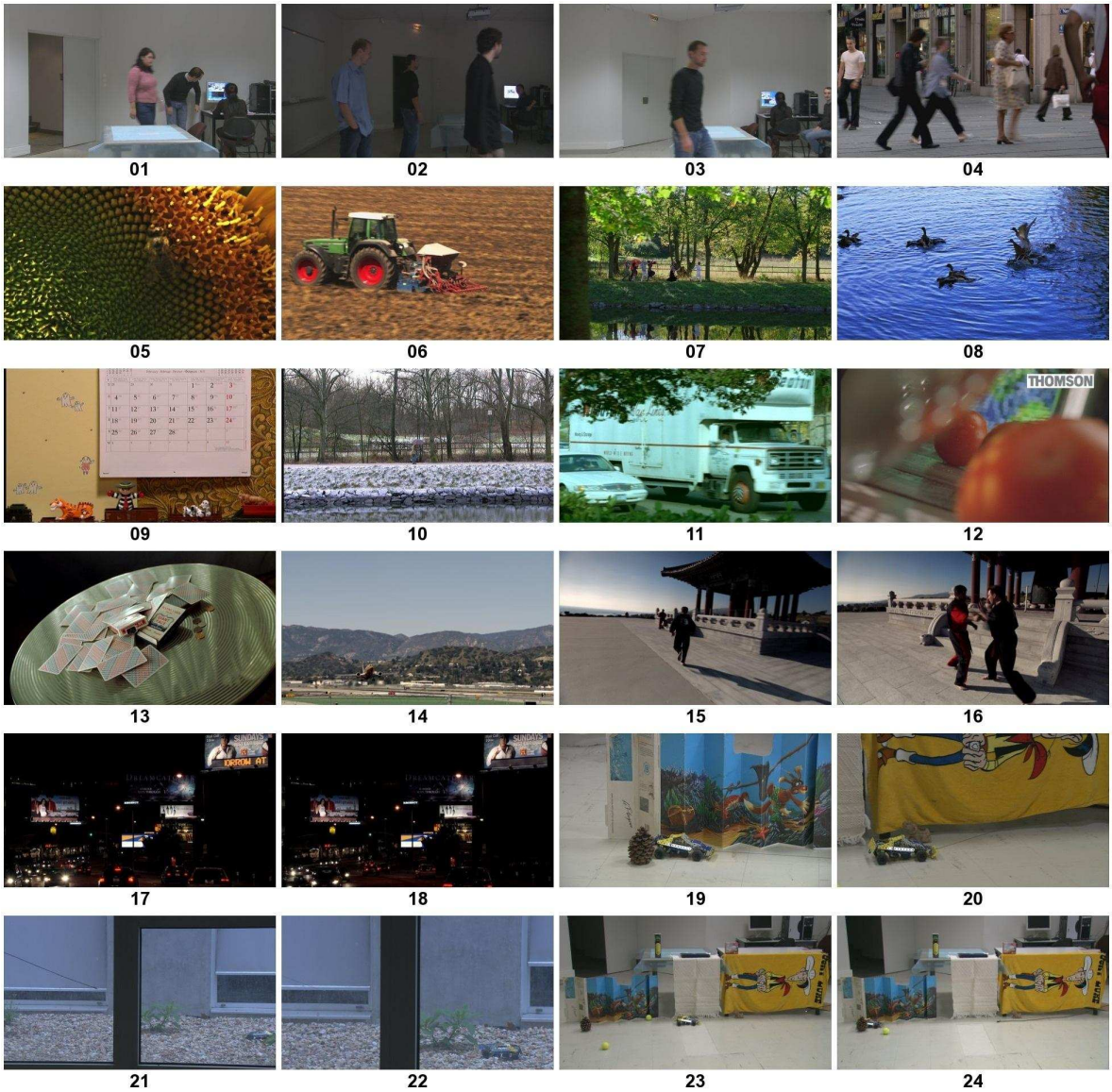


Figure A.1: Screenshots of base clips 1-24 (of 47) in the test data base for video copy detection. Copyrights: 1-3,19-41,44-45 ©*LaBRI*; 4-6 ©*TUM / Taurus Media Technik*; 7-10 ©*SVT*; 11,13 ©*Technicolor*; 12, 14-18 ©*Thomson*; 42-43, 46-47 ©*Warner Bros. Adv. Media Services Inc.*

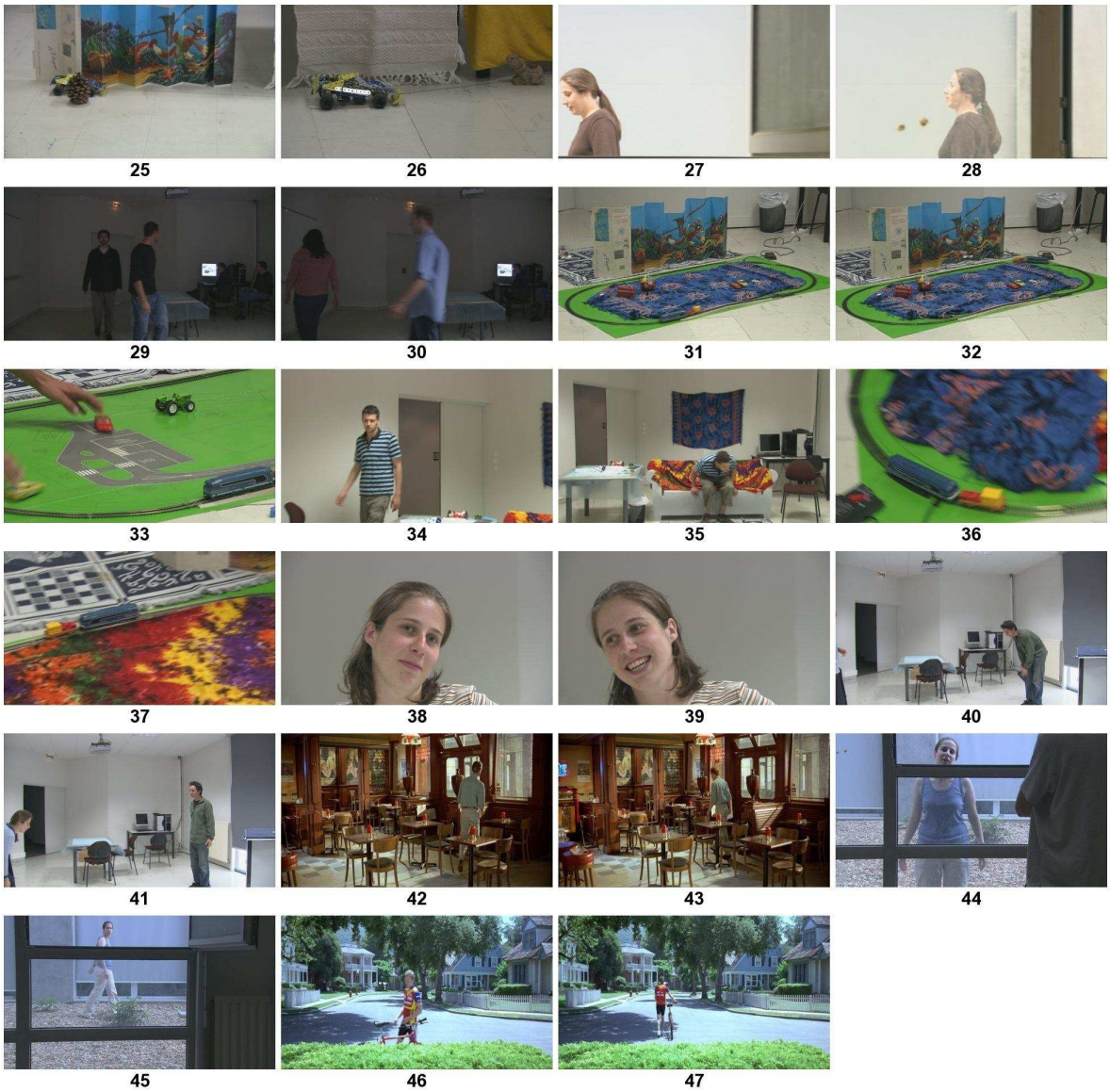


Figure A.2: Screenshots of base clips 25-47 (of 47) base clips in the test data base for video copy detection. Copyrights: 1-3,19-41,44-45 ©*LaBRI*; 4-6 ©*TUM / Taurus Media Technik*; 7-10 ©*SVT*; 11,13 ©*Technicolor*; 12, 14-18 ©*Thomson*; 42-43, 46-47 ©*Warner Bros. Adv. Media Services Inc.*

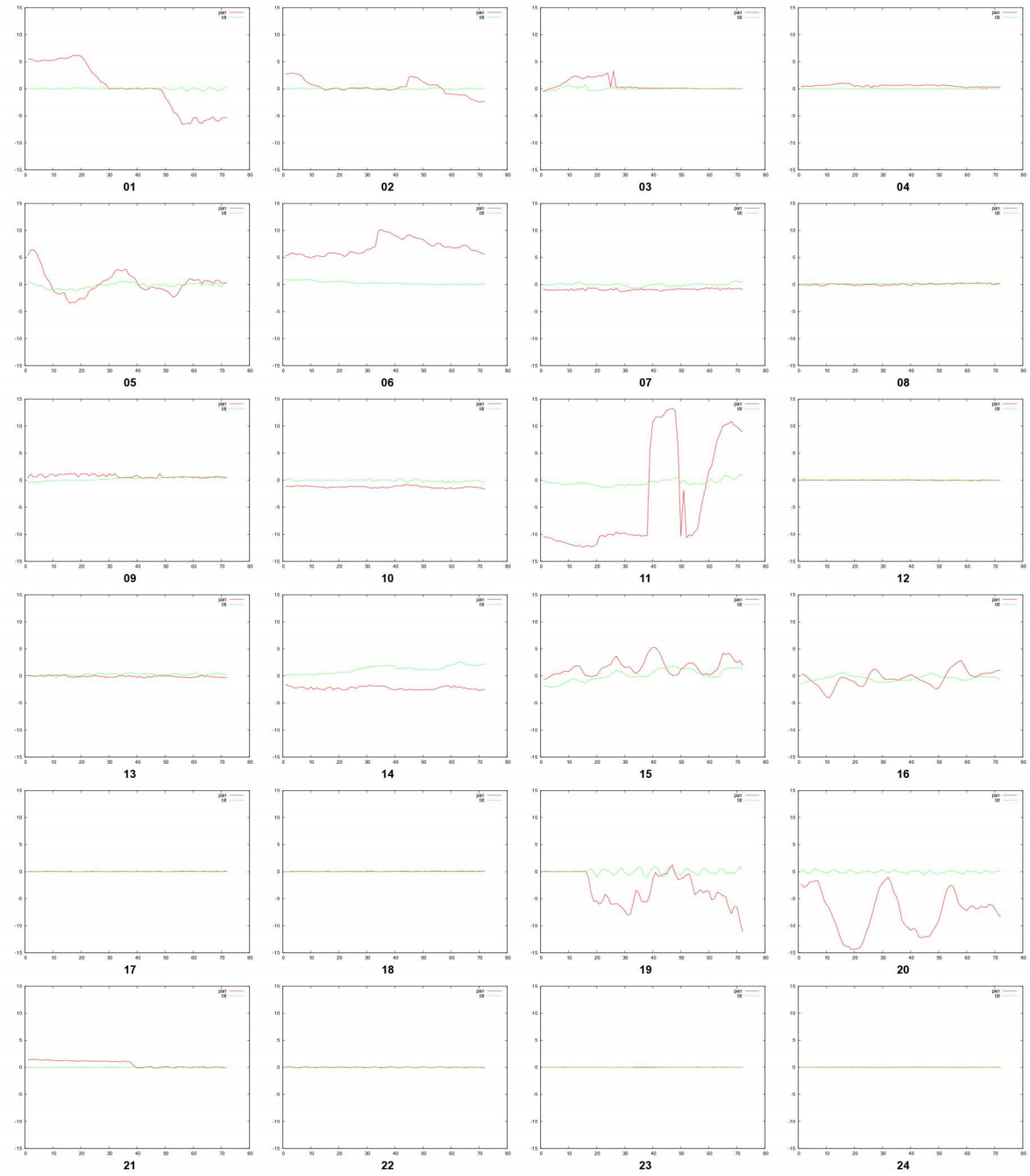


Figure A.3: Global motion estimation of base clips 1-24 (of 47) in the test data base for video copy detection.

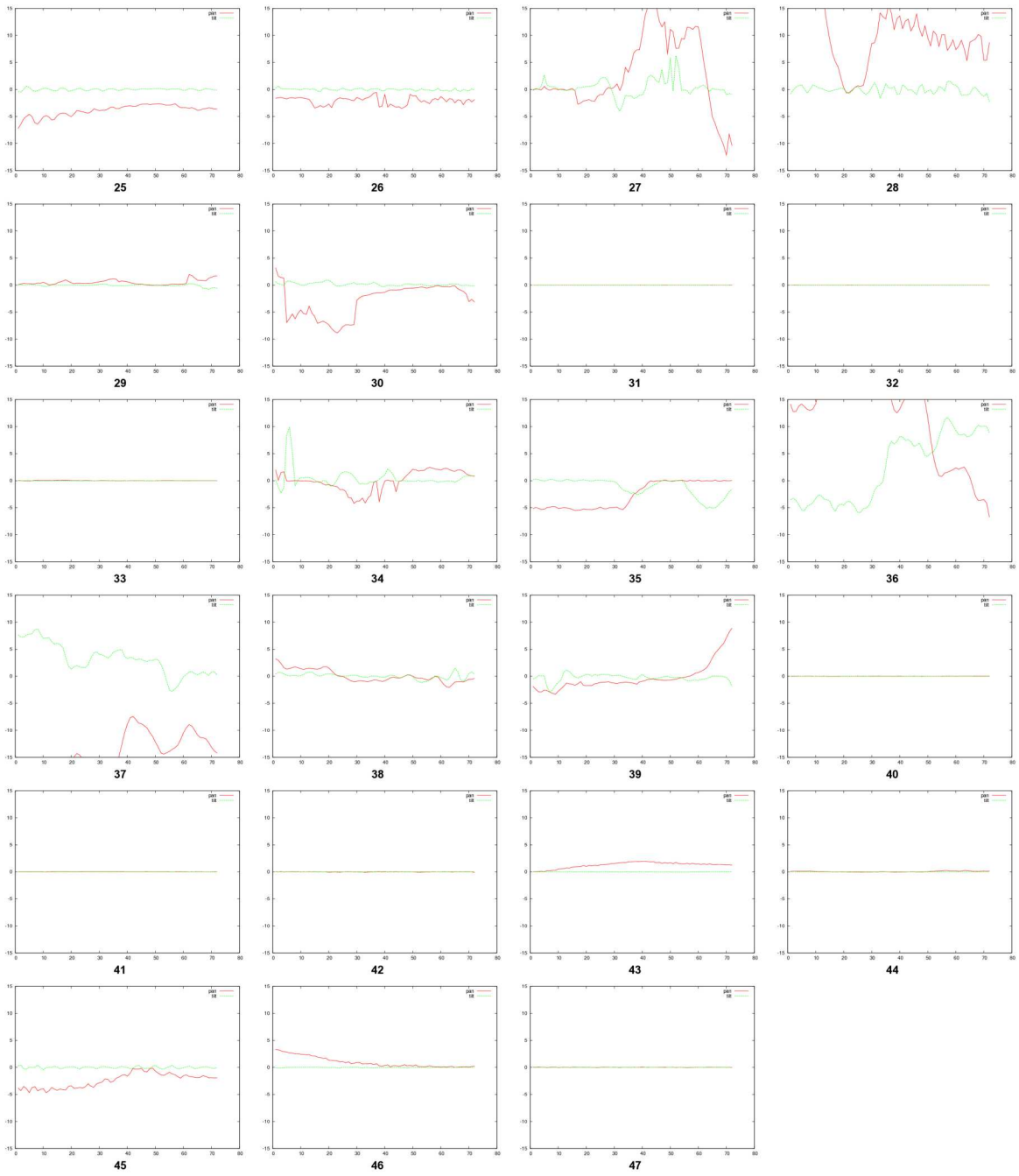


Figure A.4: Global motion estimation base clips 25-47 (of 47) in the test data base for video copy detection.

Bibliography

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, Circuits and Systems for Video Technology, IEEE Transactions on 13 (7) (2003) 560–576. xiii, 18, 22

[2] H. Schwarz, D. Marpe, T. Wiegand, Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Transactions on Circuits and Systems for Video Technology 17 (9) (2007) 1103–1119. xiii, xiv, 22, 23, 24, 25, 26, 28, 73, 77

[3] Fraunhofer Henrich-Hertz-Institut (HHI), The scalable video coding amendment of the H.264/AVC standard (November 2009) [cited 26/10/2009].
URL `http://ip.hhi.de/imagecom_G1/savce/index.htm` xiii, 24

[4] I. R. T. VISTA, Motion2D - A software to estimate 2D parametric motion models, http://www.irisa.fr/vista/Motion2D/. xiv, xvii, 42, 43, 117, 140, 142, 144, 147

[5] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proceedings of the Institute of Radio Engineers (IRE) 40 (9) (1952) 1098–1101. 9

[6] J. Rissanen, G. G. L. Jr., Arithmetic coding, IBM Journal of Research and Development 23 (2) (1979) 149–162. 9

[7] I. E. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia, 1st Edition, Wiley, 2003. 9, 22

[8] D. Vatolin, A. Moskvin, O. Petrov, A. Titarenko, JPEG 2000 Image Codecs Comparison, Tech. rep., CS MSU Graphics and Media Lab - Video Group (September 2005). 10

[9] I. Daubechies, Ten Lectures on Wavelets. CBMS - NSF Regional Conference Series in Applied Mathematics, Society for Industrial & Applied Math, 1992. 10

[10] D. S. Taubman, M. W. Marcellin, JPEG 2000: Image Compression Fundamentals, Standards and Practice, Kluwer Academic Publishers, Norwell, MA, USA, 2001. 12

[11] C. Parisot, M. Antonini, M. Barlaud, 3D Scan-Based Wavelet Transform And Quality Control for Video Coding, in: EURASIP Special Issue Multimedia Signal Processing, 2003, pp. 56–65. 12

[12] M. Tagliasacchi, D. Maestroni, S. Tubaro, A. Sarti, Motion estimation and signaling techniques for 2D+t scalable video coding, EURASIP Journal on Applied Signal Processing 2006 (2009) 266–266. 12

[13] K. Rao, P. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications, Academic Press, 1990. 13

[14] P. Tudor, MPEG-2 video compression, IEE Electronics & Communications Engineering Journal (1995) 257–264. 14, 16

[15] T. Sikora, Digital Video Coding Standards, McGraw-Hill Inc., Hightstown, NJ, USA, 1997. 16

[16] G. J. Sullivan, P. N. Topiwala, A. Luthra, The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions, Applications of Digital Image Processing XXVII 5558 (1) (2004) 454–474. 17

[17] S. Moiron, S. Faria, A. Navarro, V. Silva, P. Assunç ao, Video transcoding from h.264/avc to mpeg-2 with reduced computational complexity, Signal Processing: Image Communication 24 (8) (2009) 637–650. 21

[18] C. Cedras, M. Shah, Motion-based recognition: A survey, IVC 13 (2) (1995) 129–155. 32

[19] JSVM reference software for H.264/SVC.
URL `http://ftp3.itu.ch/av-arch/jvt-site/` 33, 38, 75, 118, 139

[20] OpenSVC decoder: A real-time coding framework for the SVC extension of H.264/AVC.
URL `http://sourceforge.net/projects/opensvcdecoder/` 33

[21] Y. Liu, W. Wang, W. Gao, W. Zeng, A novel compressed domain shot segmentation algorithm on H.264/AVC, in: IEEE International Conference on Image Processing (ICIP'04), Vol. 4, 2004, pp. 2235–2238. 36

[22] S. D. Bruyne, W. D. Neve, D. D. Schrijver, P. Lambert, P. Verhoeve, R. V. de Walle, Shot boundary detection for H.264/AVC bitstreams with frames containing multiple types of slices, in: PCM, 2007, pp. 177–186. 36

[23] S. De Bruyne, D. Van Deursen, J. De Cock, W. De Neve, P. Lambert, R. Van de Walle, A compressed-domain approach for shot boundary detection on H.264/AVC bit streams, Signal Processing: Image Communication 23 (7) (2008) 473–489. 36

[24] V. Kobla, D. Doermann, K. Lin, C. Faloutsos, Compressed domain video indexing techniques using DCT and motion vector information in MPEG video, in: SPIE Conference on Multimedia Storage and Archiving Systems, Vol. 3022, 1997, pp. 200–211. 37

[25] W.-Y. Yoo, J. Lee, Analysis of camera operations in MPEG compressed domain based on generalized hough transform, in: PCM '01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia, Springer-Verlag, London, UK, 2001, pp. 1102–1107. 37

[26] W. Hesseler, S. Eickeler, MPEG-2 compressed-domain algorithms for video analysis, EURASIP Journal on Applied Signal Processing 2 (2006) 1–11. 37, 46, 53, 145

[27] R. Wang, T. Huang, Fast camera motion analysis in MPEG domain, in: Proceedings of IEEE International Conference on Image Processing (ICIP'99), Kobe, Japan, Vol. 3, 2001, pp. 691–694. 37

[28] J. Odobez, P. Bouthemy, Robust multiresolution estimation of parametric motion models, Journal of Visual Communication and Image Representation (JVCIR) 6 (4) (1995) 348–365. 37

[29] P. Bouthemy, M. Gelgon, F. Ganansia, A unified approach to shot change detection and camera motion characterization, Vol. 9, 1999, p. 1030. 37, 42

[30] M. Durik, J. Benois-Pineau, Robust motion characterisation for video indexing based on MPEG2 optical flow, in: Proceedings of International Workshop on Content-Based Multimedia Indexing (CBMI'01), Brescia, Italy, 2001, pp. 57–64. 37, 38, 39, 40

[31] J.-G. Kim, H. Sung, J. Kim, H.-M. Kim, Threshold-based camera motion characterization of MPEG video, ETRI Journal 26 (3) (2004) 269–272. 38

[32] H. C. Longuet-Higgins, The Visual Ambiguity of a Moving Plane, Royal Society of London Proceedings Series B 223 (1984) 165–175. 39

[33] G. Adiv, Determining three-dimensional motion and structure from optical flow generated by several moving objects, IEEE Transactions on Pattern Analysis and Machine Intelligence 7 (4) (1985) 384–401. 39

[34] M. Irani, P. Anandan, Video indexing based on mosaic representations, IEEE Transactions on Pattern Analysis and Machine Intelligence 86 (5) (1998) 905–921. 39

[35] Y.-P. Tan, S. Kulkarni, P. Ramadge, A new method for camera motion parameter estimation, International Conference on Image Processing (ICIP'95) 1 (1995) 406. 39

[36] J. M. Odobez, P. Bouthemy, Robust multiresolution estimation of parametric motion models, Journal of Visual Communication and Image Representation 6 (4) (1995) 348–365. 42, 43, 140, 147

[37] P. Kraemer, O. Hadar, J. Benois-Pineau, J. Domenger, Super-resolution mosaicing from MPEG compressed video, in: Proc. IEEE International Conference on Image Processing (ICIP'05), Vol. 1, Genova, Italy, 2005, pp. 893–896. 45, 137

[38] F. Manerba, J. Benois-Pineau, R. Leonardi, B. Mansencal, Multiple moving object detection for fast video content description in compressed domain, EURASIP J. Adv. Signal Process 2008 (1) (2008) 1–13. 45, 53, 138

[39] C. Stauffer, W. E. L. Grimson, Adaptive background mixture models for real-time tracking, in: Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., Vol. 2, 1999, p. 252. 46, 77

[40] B. Han, D. Comaniciu, L. Davis, Sequential kernel density approximation through mode propagation: Applications to background modeling, in: In Proceedings of Asian Conf. on Computer Vision (ACCV'04), 2004. 46

[41] A. M. Elgammal, D. Harwood, L. S. Davis, Non-parametric model for background subtraction, in: Proceedings of the 6th European Conference on Computer Vision-Part II (ECCV'00), Springer-Verlag, London, UK, 2000, pp. 751–767. 46

[42] N. Oliver, B. Rosario, A. Pentland, A bayesian computer vision system for modeling human interactions, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (1999) 831–843. 46

[43] C. Käs, H. Nicolas, An approach to trajectory estimation of moving objects in the H.264 compressed domain, in: Pacific Rim Symposium on Image and Video Technology (PSIVT'09), Tokyo, Japan, 2009. 46, 86

[44] O. Sukmarg, K. Rao, Fast object detection and segmentation in MPEG compressed domain, in: 10th IEEE Region Annual International Conference, Vol. 3, Kuala Lumpur, Malaysia, 2000, pp. 364–368. 46

[45] V. Mezaris, I. Kompatsiaris, N. V. Boulgouris, M. G. Strintzis, Real-time compressed-domain spatiotemporal segmentation and ontologies for video indexing and retrieval, in: IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, 2004, pp. 606–621. 46, 74

[46] X.-D. Yu, L.-Y. Duan, Q. Tian, Robust moving video object segmentation in the MPEG compressed domain, in: International Conference on Image Processing (ICIP'03), Vol. 3, Barcelone, Spain, 2003, pp. 933–936. 46, 74

[47] M. Coimbra, , M. Davies, Segmentation of moving pedestrians within the compressed domain, in: IEEE International Conference on Acoustics, Signal- and Speech Processing (ICASSP'04), 2004. 46

[48] S. Treetasanatavorn, U. Rauschenbach, J. Heuer, A. Kaup, Bayesian method for motion segmentation and tracking in compressed videos, in: LNCS Pattern Recognition, Vol. 3663, Springer Berlin / Heidelberg, 2005, pp. 277–284. 46

[49] R. V. Babu, K. Ramakrishnan, Content-based video retrieval using motion descriptors extracted from compressed domain, in: IEEE International Symposium on Circuits and Systems (ISCAS2002), Vol. 4, Phoenix, USA, 2002, pp. 141–144. 46, 74

[50] R. V. Babu, K. Ramakrishnan, S. Srinivasan, Video object segmentation: a compressed domain approach, IEEE Transactions on Circuits Systems for Video Technology 14 (4) (2004) 462–474. 46, 74

[51] W. Zeng, J. Du, W. Gao, Q. Huang, Robust moving object segmentation on H.264/AVC compressed video using the block-based mrf model, Real-Time Imaging 11 (4) (2005) 290–299. 47, 74, 75

[52] Z. Liu, Z. Zhang, L. Shen, Moving object segmentation in the H.264 compressed domain, Optical Engineering 46 (1). 47

[53] C. Poppe, S. De Bruyne, T. Paridaens, P. Lambert, R. Van de Walle, Moving object detection in the H.264/AVC compressed domain for video surveillance applications, J. Vis. Comun. Image Represent. 20 (6) (2009) 428–437. 47, 137

[54] S. Verstockt, S. D. Bruyne, C. Poppe, P. Lambert, R. V. de Walle, Multi-view object localization in h.264/avc compressed domain, Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS'09) (2009) 370–374. 47

[55] G. R. Bradski, J. W. Davis, Motion segmentation and pose recognition with motion history gradients, Mach. Vision Appl. 13 (3) (2002) 174–184. 48

[56] A. Prati, I. Mikic, C. Grana, M. M. Trivedi, Shadow detection algorithms for traffic flow analysis: A comparative study, in: In Proceedings of Intelligent Transportation Systems, 2001., 2001, pp. 340–345. 51

[57] N. R. Pal, S. K. Pal, A review on image segmentation techniques, Pattern Recognition 26 (9) (1993) 1277 – 1294. 51

[58] A. Briassouli, N. Ahuja, Integration of frequency and space for multiple motion estimation and shape-independent object segmentation, IEEE Transactions on Circuits, Systems and Video Technology 18 (5) (2008) 657–669. 51

[59] A. Gelb, Applied optimal estimation, MIT Press, 1974. 52

[60] D. Reid, An algorithm for tracking multiple targets, IEEE Transactions on Automatic Control 24 (6) (1979) 843–854. 52

[61] M. Isard, A. Blake, CONDENSATION - conditional density propagation for visual tracking, International Journal of Computer Vision 29 (1) (1998) 5–28. 52

[62] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 25 (5) (2003) 564–575. 52

[63] M. Han, W. Xu, H. Tao, Y. Gong, An algorithm for multiple object trajectory tracking, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04) 1 (2004) 864–871. 52

[64] B. Leibe, K. Schindler, L. v. Gool, Coupled detection and trajectory estimation for multi-object tracking, in: International Conference on Computer Vision (ICCV'07), Rio de Janerio, Brasil, 2007, pp. 1–8. 52

[65] W.-N. Lie, W.-C. Hsiao, Content-based video retrieval based on object motion trajectory, in: IEEE Workshop on Multimedia Signal Processing, 2002, pp. 237–240. 53

[66] P. M. Fonseca, J. Nesvadba, Face tracking in the compressed domain, EURASIP J. Appl. Signal Process. 2006 (2006) 187–187. 53

[67] L. Gu, Scene analysis of video sequences in the MPEG domain, in: In Proceedings of International Conference of Signal and Image Processing (SIP'98), Las Vegas, USA, 1998, pp. 384–398. 53

[68] H. Chen, Y. Zhan, F. Qi, Rapid object tracking on compressed video, in: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia (PCM'01), Springer-Verlag, London, UK, 2001, pp. 1066–1071. 53

[69] A. Radhakrishna, M. Kankanhalli, P. Mulhem, Compressed domain object tracking for automatic indexing of objects in MPEG home video, in: IEEE International Conference in Multimedia and Expo (ICME 2002), Lausanne, Switzerland, 2002. 53

[70] W. Zeng, W. Gao, D. Zhao, Automatic moving object extraction in MPEG video, in: In Proceeding of the IEEE International Symposium on Circuits and Systems (ISCAS'03), Vol. 2, 2003, pp. 524–527. 53

[71] F. Porikli, Real-time video object segmentation for MPEG encoded video sequences, in: SPIE Conference on Real-Time Imaging VIII, Vol. 5297, 2004, pp. 195–203. 53

[72] A. Aggarwal, S. Biswas, S. Singh, S. Sural, A. Majumdar, Object tracking using background subtraction and motion estimation in MPEG videos, in: ACCV06, 2006, pp. II:121–130. 53, 74, 75

[73] L. Favalli, A. Mecocci, F. Moschetti, Object tracking for retrieval applications in MPEG-2, in: IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, 2000, pp. 427–432. 53, 74

[74] W.-N. Lie, R.-L. Chen, Tracking moving objects in MPEG-compressed videos, in: IEEE International Conference on Multimedia and Expo (ICME'01), Vol. 2001, 2001, p. 245. 53, 74

[75] S.-M. Park, J. Lee, Object tracking in MPEG compressed video using mean-shift algorithm, in: 4th Pacific Rim Conference on Multimedia, Vol. 2, Singapore, 2003, pp. 748–752. 53, 75

[76] R. D. Sutter, K. D. Wolf, S. Lerouge, R. V. de Walle, Lightweight object tracking in compressed video streams demonstrated in region-of-interest coding, EURASIP Journal on Applied Signal Processing 2007 (1) (2007) 59–59. 53, 74

[77] W. You, M. Sabirin, M. Kim, Moving object tracking in H.264/AVC bitstream, in: International Workshop on Multimedia Content Analysis and Mining (MCAM'07), 2007, pp. 483–492. 53, 74, 75

[78] K. Mehmood, M. Mrak, J. Calic, A. Kondoz, Object tracking in surveillance videos using compressed domain features from scalable bit-streams, Signal Processing: Image Communication 24 (10) (2009) 814–824. 53, 74, 75

[79] M. Ritch, N. Canagarajah, Motion-based video object tracking in the compressed domain, in: IEEE International Conference on Image Processing (ICIP 2007), Vol. 6, San Antonio, Texas, USA, 2007, pp. 301–304. 53, 54

[80] G. B. Rath, A. Makur, Iterative least squares and compression based estimations for a four-parameter linear global motion model and global motion compensation, IEEE Transactions on Circuits and Systems for Video Technology 9 (7) (1999) 1075–1099. 53

[81] A. F. Bobick, J. W. Davis, An appearance-based representation of action, in: Proceedings of the 13th International Conference on Pattern Recognition (ICPR'96), Vol. 1, 1996, pp. 307–312. 57

[82] A. Turolla, L. Marchesotti, C. Regazzoni, Multicamera object tracking in video surveillance applications, in: Target Tracking 2004. Algo. and App., IEE, 2004, pp. 85–90. 60

[83] K.-S. Park, J. Lee, M. Stanacevic, S. Hong, W.-D. Cho, Iterative object localization algorithm using visual images with a reference coordinate, EURASIP Journal on Image and Video Processing 2008 (2008) 16. 60

[84] R. Rosales, S. Sclaroff, 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Fort Collins, CO, USA, 1999, pp. 117—123. 60

[85] S. Ono, S. Komatsu, Simultaneous image restoration and distance estimation of an object from a strongly defocused photon image, Optical engineering 42 (2003) 1024–1028. 62

[86] P. Gil, S. Lafuente, S. Maldonado, F. Acevedo, Distance estimation from image defocus for video surveillance systems, Electronics Letters 40 (2004) 1047–1049. 62

[87] R. Rao, L. Seungsin, A video processing approach for distance estimation, in: Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'06), Vol. 3, Toulouse, France, 2006. 62

[88] Q. Mei-bin, Z. Rui, J. Jian-guo, L. Xiang-tao, Moving object localization with single camera based on height model in video surveillance, in: 1st International Conference on Bioinformatics and Biomedical Engineering (ICBBE'07), Wuhan, China, 2007, pp. 490–493. 62

[89] C. Pang, A. Guesalaga, V. Roda, Robust estimation of 3D trajectories from a monocular image sequence, International Journal of Imaging Systems and Technology 12 (3) (2002) 128–137. 62

[90] D. M. Gavrila, The visual analysis of human movement: a survey, Comput. Vis. Image Underst. 73 (1) (1999) 82–98. 62

[91] V. Lepetit, P. Fua, Monocular model-based 3D tracking of rigid objects: A survey, Foundations and Trends in Computer Graphics and Vision 1 (1) (2005) 1–89. 62

[92] W. Hu, X. Xiao, Z. Fu, D. Xie, A system for learning statistical motion patterns, IEEE Trans. Pattern Anal. Mach. Intell. 28 (9) (2006) 1450–1464. 66

[93] A. Basharat, A. Gritai, M. Shah, Learning object motion patterns for anomaly detection and improved object detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'08) 0 (2008) 1–8. 66

[94] K. Sudo, T. Osawa, K. Wakabayashi, H. Koike, K. Arakawa, Estimating Anomality of the Video Sequences for Surveillance Using 1-Class SVM, IEICE - Transactions on Information and Systems E91-D (7) (2008) 1929–1936. 66

[95] C. Käs, H. Nicolas, H.264 Scene Motion Analysis, in: IEEE International Conference on Acoustics, Signal- and Speech Processing (ICASSP'09), Taipeh, Taiwan, 2009. 68

[96] D. Hoiem, A. Efros, M. Hebert, Putting objects in perspective, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006. 71, 91

[97] D. Hoiem, A. A. Efros, M. Hebert, Putting objects in perspective, International Journal of Computer Vision (IJCV) 80 (1) (2008) 3–15. 71, 91, 92, 99

[98] W. You, M. S. H. Sabirin, M. Kim, Real-time detection and tracking of multiple objects with partial decoding in H.264/AVC bitstream domain, in: 21st Annual IST/SPIE Symposium on Electronic Imaging, Vol. 7244 of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 2009. 75

[99] P. Kaewtrakulpong, R. Bowden, An improved adaptive background mixture model for realtime tracking with shadow detection, in: In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, (AVBS 2001), 2001. 77

[100] F. Chang, C.-J. Chen, C.-J. Lu, A linear-time component-labeling algorithm using contour tracing technique, Computer Vision Image Understanding 93 (2) (2004) 206–220. 82, 84

[101] H. Nicolas, J.-M. Pinel, Joint moving cast shadows segmentation and light source detection in video sequences, Signal processing. Image communication 21 (2006) 22–43. 83

[102] P. L. Rosin, T. Ellis, Image difference threshold strategies and shadow detection, in: Proceedings of the 1995 British conference on Machine vision, Vol. 1, BMVA Press, Surrey, UK, UK, 1995, pp. 347–356. 83

[103] T. Horprasert, D. Harwood, L. S. Davis, A statistical approach for real-time robust background subtraction and shadow detection, in: ICCV Frame-Rate WS, 1999, pp. 1–19. 83

[104] I. Mikic, P. C. Cosman, G. T. Kogut, M. M. Trivedi, Moving shadow and object detection in traffic scenes, International Conference on Pattern Recognition (ICPR'00) 1 (2000) 1321. 83

[105] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, Detecting moving objects, ghosts, and shadows in video streams, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (10) (2003) 1337–1342. 83

[106] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2004) 91–110. 85, 116, 121, 124

[107] K. Mikolajczyk, C.Schmid, A performance evaluation of local descriptors, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE 27 (10) (2005) 1615–1630. 85, 116

[108] C. Käs, M. Brulin, H. Nicolas, C. Maillet, Compressed domain aided analysis of traffic surveillance videos, in: Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC'09), Como, Italy, 2009. 88

[109] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, SURF: Speeded Up Robust Features, Computer Vision and Image Understanding (CVIU), Quebec City, Canada 110 (3) (2008) 346–359. 88, 124, 127

[110] C. Sas, G. O'Hare, R. Reilly, On-line trajectory classification, in: Proceedings of the International Conference on Computational Science, Vol. 2659 of LNCS, Springer-Verlag, Berlin, 2003, pp. 1035–1044. 90

[111] X. Wang, K. Tieu, E. Grimson, Learning semantic scene models by trajectory analysis, in: 9th European Conference on Computer Vision (ECCV'06), Graz, Austria, 2006, pp. 110–123. 90

[112] B. Morris, M. Trivedi, A survey of vision-based trajectory learning and analysis for surveillance, IEEE Transactions on Circuits and Systems for Video Technology 18 (8) (2008) 1114–1127. 90

[113] I. Ivanov, F. Dufaux, T. M. Ha, T. Ebrahimi, Towards Generic Detection of Unusual Events in Video Surveillance, in: Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009. 90

[114] O. Faugeras, Q.-T. Luong, T. Papadopoulou, The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications, MIT Press, Cambridge, MA, USA, 2001. 92

[115] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004. 92

[116] S. Soatto, P. Perona, Reducing "Structure From Motion": a general framework for dynamic vision part 1: Modeling, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (9) (1998) 933–942. 92

[117] Y. Ma, S. Soatto, J. Kosecka, S. Sastry, An Invitation to 3D Vision: From Images to Geometric Models, no. 26 in Interdisciplinary Applied Mathematics Series, Springer Verlag, New York, LLC, 2003. 92

[118] V. P. Namboodiri, S. Chaudhuri, Recovery of relative depth from a single observation using an uncalibrated (real-aperture) camera, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'08), Anchorage, Alaska, 2008, pp. 1–6. 92

[119] F. Dellaert, S. M. Seitz, C. E. Thorpe, S. Thrun, Structure from motion without correspondence, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00), Hilton Head, SC, USA, 2000, pp. 2557–2564. 92

[120] K. Mbonye, F. Ferrie, Attentive visual servoing in the mpeg compressed domain for un-calibrated motion parameter estimation of road traffic, in: ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition, IEEE Computer Society, Washington, DC, USA, 2006, pp. 908–911. 92

[121] R. H. Byrd, R. B. Schnabel, G. A. Shultz, A trust region algorithm for nonlinearly constrained optimization, SIAM Journal on Numerical Analysis 24 (5) (1987) 1152–1170. 96

[122] T. C. Branch, M.A., Y. Li, A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems, in: SIAM Journal on Scientific Computing, Vol. 21, 1999, pp. 1–23. 96

[123] P. Ward, Picture Composition, Focal Press, 2002. 102

[124] C. Käs, H. Nicolas, Rough compressed domain camera pose estimation through object motion, in: IEEE International Conference on Image Processing (ICIP 2009), Cairo, Egypt, 2009. 104

[125] J. M. A. Sanchez, X. Binefa, J. Vitria, P. Radeva, Local color analysis for scene break detection applied to tv commercials recognition, in: in Proceedings of Visual 99, 1999, pp. 237–244. 108

[126] N. Diakopoulos, S. Volmer, Temporally tolerant video matching, in: Proc. of the ACM SIGIR Workshop on Multimedia Information Retrieval, Toronto, Canada, 2003. 108

[127] A. Basharat, Y. Zhaia, M. Shah, Content based video matching using spatiotemporal volumes, Computer Vision and Image Understanding 110 (3) (2008) 360–377. 108

[128] A. Joly, O. Buisson, C. Frelicot, Content-based copy retrieval using distortion-based probabilistic similarity search, IEEE Transactions on Multimedia 9 (2) (2007) 293–306. 108

[129] J. Law-To, O. Buisson, V. Gouet-Brunet, N. Boujemaa, Robust voting algorithm based on labels of behavior for video copy detection, in: MULTIMEDIA '06: Proceedings of

the 14th annual ACM international conference on Multimedia, ACM, New York, NY, USA, 2006, pp. 835–844. 108

[130] X.-S. Hua, X. Chen, H.-J. Zhang, Robust video signature based on ordinal measure, in: International Conference on Image Processing (ICIP'04), Vol. 1, Singapore, 2004, pp. 685–688. 108

[131] S. fu Chang, W. Chen, H. J. Meng, H. Sundaram, D. Zhong, A fully automated content-based video search engine supporting spatiotemporal queries, IEEE Transactions on Circuits and Systems for Video Technology 8 (1998) 602–615. 108

[132] P. Indyk, G. Iyengar, N. Shivakumar, Finding pirated video sequences on the internet, in: Technical report, Stanford University, 1999. 108

[133] R. Mohan, Video sequence matching, in: Int. Conf. on Acoustics, Speech and Signal Processing, Vol. 6, Seatlle, WA, USA, 1998, pp. 3697–3700. 108

[134] L. Chen, F. W. M. Stentiford, Video sequence matching based on temporal ordinal measurement, Pattern Recogn. Lett. 29 (13) (2008) 1824–1831. 108

[135] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, F. Stentiford, Video copy detection: a comparative study, in: CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval, ACM, New York, NY, USA, 2007, pp. 371–378. 108

[136] L. Chen, F. W. M. Stentiford, Video sequence matching based on temporal ordinal measurement, in: Technical report No. 1, UCL Adastral, 2006. 108

[137] A. Hampapur, R. M. Bolle, Comparison of distance measures for video copy detection, IEEE International Conference on Multimedia and Expo (ICME'01) 0 (2001) 188. 108

[138] A.Hampapur, K. Hyun, R. Bolle, Comparison of sequence matching techniques for video copy detection, in: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 4676, 2001, pp. 194–201. 108

[139] E. Ardizzone, M. L. Cascia, A. Avanzato, A. Bruna, Video indexing using mpeg motion compensation vectors, in: ICMCS '99: Proceedings of the IEEE International Conference on Multimedia Computing and Systems Volume II-Volume 2, IEEE Computer Society, Washington, DC, USA, 1999, p. 725. 108

[140] V. Kobla, D. Doermann, K. Lin, Archiving, indexing, and retrieval of video in the compressed domain, in: in Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems, 1996, pp. 78–89. 109

[141] M. Naphade, M. Yeung, B. Yeo, A novel scheme for fast and efficient video sequence matching using compact signatures (January 2000). 109

[142] T. Jain, C. Jawahar, Compressed domain techniques to support information retrieval applications for broadcast videos, in: Proceedings of National Conference on Computer Vision Pattern Recognition Image Processing and Graphics (NCVPRIPG'08), 2008, pp. 154–159. 109

[143] G. Pass, R. Zabih, J. Miller, Comparing images using color coherence vectors, in: Proceedings of the Fourth ACM International Conference on Multimedia (MULTIMEDIA'96), ACM, New York, NY, USA, 1996, pp. 65–73. 109

[144] H. Wang, A. Divakaran, A. Vetro, S. Chang, H. Sun, Survey of compressed-domain features used in audio-visual indexing and analysis, Journal of Visual Communication and Image Representation 14 (2) (2003) 150–183. 109

[145] R. Babu, K. Ramakrishnan, Compressed domain video retrieval using object and global motion descriptors, Multimedia Tools and Applications 32 (2007) 93–113. 109

[146] S. Jeannin, A. Divakaran, MPEG-7 visual motion descriptors, Circuits and Systems for Video Technology, IEEE Transactions on 11 (6) (2001) 720–724. 112

[147] X. Sun, A. Divakaran, B. S. Manjunath, A motion activity descriptor and its extraction in compressed domain, in: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia (PCM'01), Springer-Verlag, London, UK, 2001, pp. 450–457. 112

[148] J. S. Beis, D. G. Lowe, Shape indexing using approximate nearest-neighbour search in high-dimensional spaces, in: In Proc. IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, 1997, pp. 1000–1006. 116

[149] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006, pp. 459–468. 116

[150] A. Auclair, L. D. Cohen, N. Vincent, How to use sift vectors to analyze an image with database templates, in: 5th International Workshop on Adaptive Multimedial Retrieval: Retrieval, User, and Semantics (AMR'07), Paris, France, 2007, pp. 224–236. 116

[151] ICOS-HD, French National Research Project ANR-06-MDCA-010-03, http://icos-hd.irisa.fr/. 118

[152] Source Forge, LibMPEG2 - A free MPEG-2 video stream decoder, http://libmpeg2.sourceforge.net/. 129

[153] C. Käs, H. Nicolas, Compressed domain copy detection of scalable SVC videos, in: 7th International Workshop on Content-Based Multimedia Indexing (CBMI'08), Chania, Greece, 2009. 132

[154] A. Hakeem, K. Shafique, M. Shah, An object-based video coding framework for video sequences obtained from static cameras, in: Proceedings of the 13th annual ACM International Conference on Multimedia (ACM MULTIMEDIA'05), ACM, New York, NY, USA, 2005, pp. 608–617. 138

[155] R. Babu, A. Makur, Object-based surveillance video compression using foreground motion compensation, in: 9th International Conference on Control, Automation, Robotics and Vision (ICARCV'06), 2006, pp. 1–6. 138

[156] T. Nishi, H. Fujiyoshi, Object-based video coding using pixel state analysis, in: Proceedings of the International Conference on Pattern Recognition (ICPR'04), Vol. 3, IEEE Computer Society, Washington, DC, USA, 2004, pp. 306–309. 138

[157] H. Perez-Iglesias, A. Dapena, L. Castedo, A novel video coding scheme based on principal component analysis, in: IEEE Workshop on Machine Learning for Signal Processing, 2005, pp. 361–366. 138

[158] W. R. Schwartz, H. Pedrini, L. S. Davis, Video compression and retrieval of moving object location applied to surveillance, in: Proceedings of the 6th International Conference on Image Analysis and Recognition (ICIAR'09), Springer-Verlag, Berlin, Heidelberg, 2009, pp. 906–916. 138

[159] A. Vetro, H. Sun, An overview of mpeg-4 object-based encoding algorithms, International Conference on Information Technology: Coding and Computing (ITCC'01) 0 (2001) 366. 139

[160] R. Jin, Y. Qi, A. Hauptmann, A probabilistic model for camera zoom detection, in: 16th Conference of the International Association for Pattern Recognition (ICPR'02), Quebec City, Canada, 2002. 145

[161] C. Käs, H. Nicolas, Joint global motion indexing/coding of scalable H.264/SVC HD video streams, in: Sixth International Workshop on Content-Based Multimedia Indexing (CBMI'08), London, UK, 2008. 147