From Sprites to Global Motion Temporal Filtering

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Verleihung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

vorgelegt von
Dipl.-Ing. Andreas Krutz

Tag der wissenschaftlichen Aussprache: 25.02.2010

Berlin 2010

D 83

To Jana and my parents

Acknowledgment

I would like to express my honest gratitude to my supervisor Prof. Dr. Thomas Sikora, who supported and motivated me regarding my research and beyond. The same gratitude goes to Prof. Dr. Michael Frater for our collaborative research over the last years.

Very special thanks go to my colleague and friend Alexander Glantz for a lot of delightful and successful collaborative work, which resulted not only in quite a number of publications. Special thanks go also to my long-time friend and colleague Martin Haller for enlightning discussions about and beyond research topics.

I would like to thank my colleagues Dr. Matthias Kunter, Michael Droese and Dr. Sebastian Knorr for collaborations and sharing new and forward-looking ideas.

Further, I would like to acknowledge Dr. Heiko Schwarz, Prof. Dr. Thomas Wiegand, and Dr. Karsten Müller for discussions and technical support.

I also would like to thank Prof. Dr. Mark Pickering. We had (and hopefully will have) a great time at various places around the world.

Finally, my greatest thank goes to my parents and girlfriend. To my mother and father for all their support and love. And, above all, to Jana for staying by my side at any moment in time, as strange as it was.

Abstract

Video coding techniques have evolved over recent decades. Since digital video representation and transmission have replaced the analogue counterpart, efficient compression of digitized video is a very important topic in the whole processing chain. As well as common TV-broadcast and storage media like DVD or Bluray-Disk, other devices and platforms showing video content have been developed such as handheld devices and, especially, the Internet. Popular internet platforms, e.g. YouTube, Myvideo, Sevenload, etc., have led to the transmission of large amounts of video data. Further, the latest development of High-Definition (HD) displays demands high-definition video content, which means higher resolution video than the common TV-broadcast format. It has been shown that the latest video coding standard H.264/AVC, which has outstanding coding performance for Standard-Definition (SD) resolution, can be significantly improved applying enhanced and new techniques for HD-resolution video content.

All these aspects point to a great increase of video data material for all media, requiring ongoing research, development and enhancement of existing techniques and finding new approaches for efficient encoding of this huge amount of data. For that, a number of algorithms has been developed. The most successful technique is DCT-based motion-compensated prediction. This so-called hybrid video coding approach has been the subject matter in various standardization processes and has been used until today in almost all applications described above.

Alternative approaches for efficiently encoding video data have also been pursued. One method, which can be described as “model-based”, analyzes the video content first, separates the content into objects and codes these separately. After transmission, the separated objects are decoded and merged to the original form. This technique brought very high coding gain in comparison to the hybrid video coding approach and therefore, it was standardized almost ten years ago. However, this “object-based” coding approach has several limitations, e.g. the object segmentation in the pre-analysis step and the content-dependent coding performance. Since the standardization, people have tried to develop techniques to improve this type of coding. Some improvements have been developed considering one object representation called Sprite, where all the background information of an entire video sequence is mapped into one image. New and more efficient algorithms have been developed to build such a Sprite. Furthermore, these Sprite representations have been included in

encoding environments to show some improvements comparing to hybrid video encoding. However, a lot of open issues remain for bringing this type of encoding, which was called Sprite coding during the standardization process, to the market.

Therefore, the motivation of this thesis is to build a bridge between Sprite coding and the hybrid video coding approach to both combine advantages and minimize disadvantages. It starts with the classical Sprite coding technique. Then, the Sprite-based representation is integrated in a coding environment using the latest standardized video codec, H.264/AVC. Although H.264/AVC is not designed for model- or object-based representations, a significant improvement of coding efficiency is shown using the Sprite-based approach. Further, pre-analysis steps, such as automatic object segmentation and analysis of the content of the video for checking whether the video is appropriate for Sprite coding or not are also examined. Different kinds of visual quality metrics are also used to even emphasize the subjective improvement of videos coded with Sprites. Finally, a filter design will be introduced using techniques inside the Sprite generation, which has a great potential to be used not only in coding environments but also as post-processing for video enhancement or as pre-processing for further video analysis techniques.

Zusammenfassung

Videocodierungstechniken haben sich über die letzten Jahrzehnte sehr stark entwickelt. Seit die digitale Videoverarbeitung und -übertragung die analoge Technik abgelöst hat, ist die Komprimierung von Videodaten vor der Übertragung ein sehr wichtiger Bestandteil der gesamten Prozesskette der Videoübertragung. Dabei kamen zusätzlich zu den schon vorhandenen Systemen, wie normale TV-Übertragung und Speichermedien wie DVD oder Bluray-Disk, nun neue Platformen und Geräte, in denen Video angezeigt werden kann dazu. Zwei wichtige Beispiele stellen hier mobile Geräte, wie Handys und mobile Spielkonsolen, und natürlich das Internet dar. Betrachtet man populäre Internet-Platformen, wie z.B. YouTube, Myvideo, Sevenload, etc., ist zu erkennen, wie drastisch die Anzahl der Videodaten heutzutage steigt. Weiterhin erfordert die jüngste Entwicklung von High-Definition TV-Endgeräte natürlich auch High-Definition Inhalt, d.h. Videodaten mit einer höheren Auflösung als der bisher bekannte TV-Standard. Es wurde schon gezeigt, dass der letzte Videocodierungsstandard H.264/AVC, welcher eine überragende Codierungseffizienz bei Videodaten bis zu einer Auflösung des bisherigen TV-Standards hat, durch erweiterte und neue Techniken bei Anwendung auf höher aufgelöstes Videomaterial signifikant verbessert werden kann.

All diese Aspekte zeigen, dass das allgemeine fast unvorstellbare Wachstum an digitalem Videodatenmaterial für jegliche Medien eine fortlaufende Forschung und Entwicklung zur Erweiterung bestehender Codierungstechniken und neuen Ansätzen zur effizienten Codierung dieser riesigen Datenmengen erfordert. Dafür wurden einige Ansätze zu Codierung von Video bereits vorgestellt. Die erfolgreichste Technik beinhaltet eine DCT-basierte bewegungskompensierte Prädiktion. Die sogenannte hybride Videocodierung wurde bereits mehrfach in verschiedenen Standardisierungen verarbeitet und befindet sich heutzutage in fast allen Anwendungen, die oben erwähnt wurden.

Neben der hybriden Videocodierung wurden alternative Verfahren ebenfalls verfolgt. Eine “Modell-basierte” Methode analysiert zuerst den Videoinhalt, um dann diesen Inhalt in unterschiedliche Objekte zu unterteilen und diese dann separat zu codieren und zu übertragen. Am Empfänger werden die Objekte dann decodiert und wieder zum ursprünglichen Inhalt zusammengesetzt. Diese Technik brachte einen sehr hohen Codiergewinn im Vergleich zum hybriden Ansatz und wurde deshalb auch vor ungefähr zehn Jahren standardisiert. Allerdings hat dieser objektbasierte Ansatz auch große Nachteile, wie z.B. die Objektsegmentierung im Voranalyseschritt und die allgemeine inhaltsabhängige Codiereffizienz. Seit der Standardisierung wurde versucht, diese Technologie fortlaufend zu verbessern. Verbesserungen wurden gezeigt in Bezug auf eine Objektrepräsentation, welche Sprite genannt wird. In einem sogenannten Sprite wird der gesamte Hintergrundinhalt über alle Bilder einer Eingangsvideosequenz zu einem Bild zusammengefasst. Neue und effizientere Algorithmen wurden entwickelt, um solch ein Sprite aufzubauen. Weiterhin wurden die Sprites in Codierungsumgebungen eingebunden, um dadurch Codierverbesserungen im Vergleich zum hybriden Ansatz zu erreichen. Allerdings verbleibt eine signifikante Anzahl an offenen Fragen, um diese Art der Codierung, welche auch "Sprite coding" genannt wird, zur Marktanwendung zu bringen.

Deshalb ist die Motivation dieser Dissertation, eine Brücke zwischen dem "Sprite coding" und der hybriden Videocodierung zu bauen, um Vorteile beider Verfahren zu kombinieren und mögliche Nachteile zu minimieren. Es wird damit begonnen, die klassische Spritecodierungstechnik in allen Teilen der gesamten Prozesskette zu verbessern, um maximale Kompressionseffizienz für den klassischen Bereich zu erziehlen. Danach wird die Sprite-basierte Repräsentation in eine Codierumgebung eingebracht, wobei der hybride Standard H.264/AVC zur Codierung verwendet wird. Obwohl H.264/AVC nicht für die Verarbeitung von modell- oder objektbasierter Repräsentation der Eingangsdaten entwickelt wurde, kann eine signifikante Verbesserung der Codiereffizienz bei dieser Codierumgebung gezeigt werden. Weiterhin werden Voranalyseschritte betrachtet, wie z.B. ein Ansatz zu automatischen Objektsegmentierung und Inhaltsanalyse zur Definition, ob der Sprite-basierte Ansatz verwendet werden kann oder nicht. Ergebnisse werden mit bekannten objektiven Metriken zur Bildqualitätsevaluierung erstellt. Dabei werden auch Metriken verwendet, die an die subjektive menschliche Wahrnehmung angepasst sind. Schließlich wird ein Filterdesign vorgestellt, wobei Techniken aus der klassischen Spritegenerierung verwendet werden. Es wird gezeigt, dass dieses Filter großes Potential bei der Anwendung in Codierungsumgebungen, als Nachverarbeitung zur Videoinhaltsverbesserung sowie als Vorverarbeitung für weitere Videoanalysetechniken aufweist.

Contents

Chapter 1

Introduction

The transmission of digital video data is a major aspect in the subject of handling multimedia data. Beside common video transmissions like TV-broadcast and DVD or Bluray-disk, the internet carries large amounts of video data. The so-called Web 2.0 connects social networks with a huge amount of content provided by users including multimedia content, e.g. pictures and video. For example, the video data available on well-known YouTube and the popularity of these new applications is underlined in the short video shown in Fig. 1.1.

More video was uploaded
TO YouTube IN THE LAST
2 months than if abc NBC had been
airing new content 24/7/365
SINCE 1948. (Which was when ABC started broadcasting.)

Figure 1.1: Video from "DID YOU KNOW 4.0" (Click on the image while using *Adobe Reader*, http://www.youtube.com/watch?v=6ILQrUrEWe8)

With an increasing amount of classical video data (high resolution, 3D content) and new platforms of the Web 2.0, the amount of video data to be stored, shared, analyzed, transmitted or screened in the near future is unimaginable. So the demand for enhanced and new techniques for efficient compression of video data is nowadays higher than ever. This motivates the research work of this thesis.

The compression of video data to reduce the amount of bits to transmit is a heavily investigated research field. The major goal in a design of a video codec including an encoder at the transmitter and a decoder at the receiver is to find a way to compress and decompress the video data with no perceptual loss achieving as few bits as possible for transmission with as little computational cost as possible for the encoding and decoding steps. This means that two problems have to be faced. The first is to encode the video data with the rate-distortion trade-off. The second is the complexity of the whole video codec. These requirements have led to a motion compensated DCT-based approach, also known as hybrid coding, which was developed about 25 years ago. Until today, a number of coding standards have been launched and brought into the market. The most widely used video coding standard is the MPEG-2 standard, which has enabled digital TV-broadcast. The latest standard H.264/AVC includes the most advanced video codec of the present. With an increasing amount of video data, especially due to increasing picture size, need for even better coding tools has come up.

Beside this motion compensated DCT-based design, alternative approaches have been found. While a number of researchers try to shift the complexity of the encoder to decoder, others have tried to work on so-called content-based coding tools. It has been found that when the content of the input video is analyzed first and coding tools are developed which are designed to encode the input data adapted to their content features, the hybrid coding method can be improved. Considering input video data having foreground objects and a background object, one specific method is to segment this data into a foreground object segment and a background object segment and encode these data separately. At the decoder, the separated video data was decoded and merged. One significant analysis step is to transform the background information of a certain number of frames into one frame called "Sprite" in the video coding community. A lot of work has been done in this field since it was developed in the early 90's until its standardization in MPEG-4 Part 2 [66], [14]. This work was summarized in [68] including a new development of a Sprite generation approach and an approach for Sprite coding and other analysis applications using these techniques. Despite its high potential already shown in first experiments, the Sprite coding technique has not yet come into the market.

There are a number of reasons for this. The main problem is the pre-analysis step where the input video content is segmented into foreground and background objects. The first approach here is to assume that the input video data is already seperated when coding is started. However, it turned out that this is not widely applicable. The second problem of the Sprite coding approach was the high computational cost at the encoder. The pre-analysis steps of segmenting the foreground and background objects and especially the generation of the Sprite image could not be handled at this time. Additionally, because Sprite coding only works for certain kinds of video sequences, a third analysis step would be necessary to segment the input video in time to classify valid sequences of the input video which can be coded with the Sprite technology. Meanwhile, the performance of the common hybrid video coding approach has continued to improve. Due to these reasons, i.e. of the non-applicability of the Sprite coding approach and the wide success of the hybrid method, people have put aside the Sprite technology.

However, as already mentioned, the Sprite technology with all its partial algorithms has a great potential as proposed in [67] and the general idea in terms of finding a better long-term prediction is very promising. So some researchers have kept working on this technology. In [92], advanced Sprite generation algorithms were developed and its application on Sprite coding was determined. However, the general Sprite coding approach was still the same as during the standardization work years ago. It was assumed that foreground/background segmented video data is available beforehand. In [10], the aspect of object segmentation using the already generated background Sprites was examined. The Sprite image is not only a very good long-term prediction signal but can be used for automatic object segmentation because it can also be used as a background model. During the Sprite generation, the foreground objects of the sequence are removed. So together with an improved Sprite generation method called "multiple Sprites", Farin et. al showed automatic object segmentation results using Sprites as a background model. Having these works available, Kunter [39] considered all aspects before and developed enhanced algorithms including global motion estimation, multiple Sprite generation, and superresolution Sprite generation. The main step here was the development of a codec which included an automatic object segmentation step and code the segments with H.264/AVC. This was the main novelty in comparison to the general Sprite coding approach as standardized in MPEG-4 Part 2. The new automatic Sprite codec, called "Object-based Video Codec (OBVC)", was first published in Krutz et. al. [26] and using improved multiple Sprites in Kunter et al. [41], which will also be a subject matter of this thesis.

With a video codec including the Sprite technology and an in-built automatic object segmentation, it is possible to think about a next step for a general usability as foreseen in [39]. But this is not the only aspect which can be further developed. A large number of algorithms have been published as cited above including their references concerning the aspects of the whole processing chain of a Sprite coder, i.e. global motion estimation, Sprite generation, automatic object segmentation using Sprites as a background model. Further analysis for finding the validity of a sequence for Sprite coding and coder design in general. So the goal of this thesis is to find improved and new algorithms in these aspects just mentioned starting at points of the work cited above and others. A more detailed overview of what will be developed in this thesis is the following.

Main contributions including related publications:

- The problem of estimating the global motion of an input sequence is the most important pre-processing step for the whole approach. The better the frame-by-frame estimation the better is the background Sprite with all following methods including automatic object segmentation and coding. Therefore, known algorithms are evaluated and a new approach for improving the estimation of the global motion is shown in this thesis.

- A. Krutz, M. Frater, T. Sikora
  Improved Image Registration using the Up-sampled Domain
  International Workshop on Multimedia Signal Processing (MMSP) 2006,
  Victoria, BC, Canada, 03.10.2006 - 06.10.2006
- A. Krutz, M. Frater, M. Kunter, T. Sikora
  Windowed Image Registration for Robust Mosaicing of Scenes with Large Background Occlusions
  IEEE Int. Conf. on Image Processing (ICIP'06),
  Atlanta, GA, USA, 08.10.2006 - 11.10.2006
- A. Krutz, M. Frater, T. Sikora
  Window-Based Image Registration Using Variable Window Sizes
  IEEE Int. Conf. on Image Processing (ICIP 2007),
  San Antonio, Texas, USA, 16.09.2007 - 19.09.2007

• The generation of a Sprite image has been widely researched and significant improvements have been proposed. It has been shown that the use of multiple Sprites outperforms a classic Sprite image, which includes all the background information of the entire video sequence considered. The two further applications of a Sprite, the automatic object segmentation and the coding, show significantly better results with the use of a multiple Sprite. In this thesis, a Sprite generation approach will be introduced which further improves the Sprites. These are called "local background Sprites" and are firstly applied to automatic object segmentation.

- A. Krutz, A. Glantz, M. Haller, M. Droese, T. Sikora
  Multiple Background Sprite Generation using Camera Motion Characterization for Object-based Video Coding
  3DTV Conference 2008, The True Vision Capture, Transmission and Display of 3D Video,
  Istanbul, Turkey 28.05.2008 - 30.05.2008

- A. Krutz, A. Glantz, M. Frater, T. Sikora
  Local Background Sprite Generation
  International Workshop on Local and Non-Local Approximation in Image Processing, LNLA 2008, Lausanne, Switzerland, 23.08.2008 - 24.08.2008

- As mentioned above, the Sprite image can be used for an automatic object segmentation step, which can be included in an automatic Sprite codec. Reasonable results have been shown here, but, as always in the automatic segmentation issue, there is need for improving this step. Therefore, the Sprite generation approach is applied to a background subtraction-based object segmentation algorithm. So a new object segmentation algorithm based on "local background Sprites" including a number of small improvements is shown and evaluated comprehensively.

  - A. Krutz, M. Kunter, M. Mandal, M. Frater, T. Sikora
    Motion-based Object Segmentation using Sprites and Anisotropic Diffusion
    8th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), Santorini, Greece, 06.06.2007 - 08.06.2007
  - A. Krutz, A. Glantz, T. Borgmann, M. Frater, T. Sikora
    Motion-Based Object Segmentation using Local Background Sprites
    Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009), Taipei, Taiwan, 19.04.2009 - 24.04.2009

- Features of the automatic Sprite coding system (OBVC (already proposed in [39])), which has been developed in collaboration with Dr. Matthias Kunter and Dipl.-Ing. Michael Droese, are determined. Regarding the in-built automatic object segmentation, an optimal automatic object segmentation algorithm is developed which performs best within a coding environment. Further, the coding system (OBVC) is designed to use different kinds of codecs. This means that the analysis part is completely separate from the coding part. This affects that each possible hybrid video coder can be used. At the moment, the coding system is developed with MPEG-4 Part 2 and H.264/AVC. If there will come a new codec in the future it can be easily included in the OBVC. Furthermore, the extension to multi-view video coding is also discussed.

  - A. Krutz, M. Dröse, M. Kunter, M. Mandal, M. Frater, T. Sikora
    Low Bit-Rate Object-Based Multi-View Video Coding using MVC
    3DTV-Conference,
    Kos Island, Greece, 07.05.2007 - 09.05.2007

  - M. Kunter, A. Krutz, M. Dröse, M. Frater, T. Sikora
    Object-based Multiple Sprite Coding of unsegmented Videos using H.264/AVC
    IEEE Int. Conf. on Image Processing (ICIP 2007),
    San Antonio, Texas, USA, 16.09.2007 - 19.09.2007
  - A. Krutz, A. Glantz, T. Sikora, P. Nunes, F. Pereira
    Automatic Object Segmentation Algorithms for Sprite Coding using MPEG-4
    50th International Symposium ELMAR-2008,
    Zadar, Croatia, 10.09.2008 - 12.09.2008
  - A. Glantz, A. Krutz, T. Sikora, P. Nunes, F. Pereira
    Automatic MPEG-4 Sprite Coding - Comparison of Integrated Object Segmentation Algorithms
    Multimedia Tools and Applications, Special Issue on "Advances in Image and Video Processing Techniques",
    Springer, 2010

- A very important issue in video coding in general is to find the best encoder parameter settings for optimal encoding of the input video in a rate-distortion sense. There is a large number of work in this field available. Most of it, of course, refers to the hybrid video coding approach. However, there is also work done for object-based video coding. Because the newly developed automatic Sprite coding system (OBVC) has features, which have not been determined before. There is a need for a development of a rate-distortion algorithm for this codec. An approach is outlined in this thesis based on the work done for hybrid video encoders.

  - A. Krutz, A. Glantz, M. Frater, T. Sikora
    Rate-Distortion Optimization for Automatic Sprite Video Coding using H.264/AVC
    16th IEEE International Conference on Image Processing (ICIP 2009),
    Cairo, Egypt, 07.11.2009 - 11.11.2009

- Having the OBVC, it is possible to encode the input video data fully automatic. As already discussed, the Sprite coding approach can only be used for a certain kind of video content. To bring the Sprite coding to a general use, pre-analysis algorithms are developed and coding systems including these analysis steps are designed using the OBVC and a hybrid encoder, e.g. H.264/AVC. The main goal of these analysis steps is to extract features for classifying whether a video sequence is valid for Sprite coding or not. The system can then easily decide, which coder is the best for each sequence. It will be seen that using this "coder selection" a significant improvement of encoding performance is possible, when video data is processed where valid sequences for Sprite coding are included (e.g. sports broadcast, home videos, documentary).

- A. Krutz, M. Kunter, M. Dröse, M. Frater, T. Sikora
  Content-adaptive Video Coding Combining Object-based Coding and H.264/AVC
  Picture Coding Symposium (PCS),
  Lisbon, Portugal, 07.11.2007 - 09.11.2007
- A. Krutz, S. Knorr, M. Kunter, T. Sikora
  Camera Motion-Constraint Video Codec Selection
  IEEE 10th International Workshop on Multimedia Signal Processing (MMSP),
  Cairns, Queensland, Australia, 08.10.2008 - 10.10.2008

- Until now, the most widely used metric for evaluating the quality of a decoded video is the PSNR (peak signal-to-noise ratio). However, a lot of work has been done over the past decades to find a quality metric which fits more to the human visible system (HVS) and thus is more useful to evaluate video processing algorithms, such as video coding systems. The PSNR-metric is widely used because of the easy equation and its convenience regarding a design of rate-distortion methods. However, it is also known that it does not fit well to the HVS. A good example is the decoded video data coming from a Sprite codec. Due to the Sprite processing, the PSNR-values are quite low, but subjectively it is much better than with hybrid codecs at the same bit rate or even lower. A PSNR-rate-curve does not show this. Therefore, alternative quality metrics are considered and an evaluation comparing the metrics with the use of the OBVC will be shown.

- The "local background Sprites" are not only used for automatic object segmentation. It will be shown in this work that it is possible to use this method to develop a temporal deblocking method. Most of the existing deblocking methods, such as included in the latest video coding standards, work more or less spatially. The effect of a temporal filtering and its very good performance will be shown in this thesis.

  - A. Glantz, A. Krutz, M. Haller, T. Sikora
    Video Coding using Global Motion Temporal Filtering
    16th IEEE International Conference on Image Processing (ICIP 2009),
    Cairo, Egypt, 07.11.2009 - 11.11.2009
  - A. Krutz, A. Glantz, T. Sikora
    Background Modeling for Video Coding: From Sprites to Global Motion Temporal Filtering
    IEEE International Symposium on Circuits and Systems (ISCAS 2010),
    Paris, France, 30.05.2010 - 02.06.2010

- Using this temporal filtering and the alternative visual quality metrics, a new video coding system, called visual quality assessed video coding (VQVC), is developed. First, problems of the new approach are analyzed and solved. It will be shown that with this approach it is possible to perform an optimization regarding the visual human perception. First experiments evaluate the new coding scheme and show its promising performance.

- Finally, the question has come up why this approach using the local background Sprites can be successful in a video coding environment. For that, a theoretical modeling of the new method is developed and proves this method. Two aspects, the use of local background Sprites for deblocking in general, and the motion estimation error are taken into account. The theoretically achieved rate-distortion function matches the performance of the temporal deblocking and the VQVC in a real usage.

- Techniques, escpecially global motion estimation and object segmentation, developed in this thesis are also part of collaborative work apart from video coding. Applications range from compressed domain global motion estimation, video analysis (video summarization, compressed domain object segmentation), and 2D to 3D conversion.
  - M. Haller, A. Krutz, T. Sikora
    A Generic Approach for Motion-based Video Parsing
    15th European Signal Processing Conference (EUSIPCO 2007),
    Pozna?, Poland, 03.09.2007 - 07.09.2007
  - E. Dumont, B. Merialdo, S. Essid, W. Bailer, H. Rehatschek, D. Byrne, H. Bredin, N. E. O'Connor, G. J.F. Jones, A. F. Smeaton, M. Haller, A. Krutz, T. Sikora, T. Piatrik
    Rushes Video Summarization using a Collaborative Approach
    RECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia 2008,
    Vancouver, BC, Canada, 27.10.2008 - 01.11.2008
  - M. Kunter, S. Knorr, A. Krutz, T. Sikora
    Unsupervised object segmentation for 2D to 3D conversion
    IS&T/SPIE's Electronic Imaging,
    San Jose, California, USA, 18.01.2009 - 22.01.2009
  - M. G. Arvanitidou, A. Glantz, A. Krutz, T. Sikora, M. Mrak, A. Kondoz
    Global motion estimation using variable block sizes and its application to object segmentation
    International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2009),
    London, UK, 06.05.2009 - 08.05.2009

This thesis starts with fundamental algorithms, such as global motion estimation and the development of the local background Sprite generation. The object segmentation using the local background Sprites is considered next. Afterwards, the OBVC is determined and the development of an approach for a rate-distortion algorithm for OBVC is described. Having this, pre-analysis methods for a general usability are shown. Finally, alternative metrics are introduced and the temporal deblocking issue including a design for a visual quality assessed video codec is discussed. All reference work, where all these algorithms start will be cited in the introduction of each chapter.

Chapter 2

Enhanced Global Motion Estimation and Background Sprite Generation

This chapter provides recent developments in global motion estimation and Sprite generation techniques. For global motion estimation (GME), an algorithm is developed, which uses a windowing approach to initialize the gradient descent, i.e. the core technique within the GME-algorithm, to converge in the background motion even when sequences with large background occlusions are used. Up-sampling in the final estimation step is used to prevent aliasing affected from resampling due to the image warping process. Furthermore, an approach for automatic selection of window sizes is presented. The second part of the chapter describes a new Sprite generation technique. A first application for this new method is automatic object segmentation. However, the potential and further applications are examined throughout the thesis, especially in the last chapter.

2.1 Introduction

Global motion estimation is a key technique for many applications such as video mosaicing, video segmentation and coding. Input images are aligned by estimating the motion of the background object, i.e. the camera motion, along the video sequence. We consider well-known concepts in video coding and object segmentation as well as new approaches. For that, enhanced global motion estimation and background Sprite generation techniques are developed and described in this chapter.

A number of techniques for global motion estimation and video mosaicing/Sprite generation have been proposed including [50], [21], [60], [61], [8], [70], [93]. Furthermore, there is a large amount of publications building panoramic mosaics from a given image sequence [78], [79], [65], [52] as well as other applications like video indexing [19]. In the case of building a panoramic mosaic, no foreground objects are assumed in the image sequence. It is important here to register the images very accurately pixel-by-pixel. To accelerate this costly process, Steedly et. al. [74] proposed a technique where they reduced the amount of registration points to create a very fast panoramic mosaicing algorithm.

We consider the use of the Sprite generation technique for separating the background from foreground objects in a corresponding video sequence as well as coding applications. The most important step is the accurate estimation of the background motion. A pure global motion estimation algorithm suffers from errors caused by the influence of foreground objects on the calculation of the motion parameters. As a result, the motion parameters achieved do not describe the camera motion accurately relative to the background, but rather a mix of all the motions that occur in the sequence. Methods have been presented to solve this problem in [8] and [69]. Both ideas dealt with extracting large error values from the error function that has to be minimized. It has been shown in [69] that this technique works very well if one small foreground object occurs in the sequence. However, there is still the problem of trapping in not-desired minima, and large foreground objects can still lead to poor motion estimates, as in situations where there is more than one moving foreground object.

For applications like segmentation and object-based coding, the accuracy of the estimate of the motion of the background object is critical. A windowing approach is developed to achieve good accuracy, even if a large and/or more than one foreground objects occur. This enhanced window-based global motion estimation algorithm is based on the work in [8], enhanced by windowing and a number of additional techniques. It was firstly introduced in [27]. For the core gradient-based energy minimization method, an improved algorithm is used for reducing the computational complexity [1]. The Gauss-Newton algorithm is chosen for the energy minimization because it has the best performance in comparison to other approaches [2]. Both, performance and computational efficiency are enhanced by the use of an image pyramid, in which initial motion estimates are obtained using low-resolution images. The Gauss-Newton algorithm relies on a good inital estimate of the motion, especially in the translational components. This is provided using phase correlation [38] enhanced by windowing in one case and feature tracking [80], [63] in a second case. The impact of errors that occur due to spatial aliasing during the warping process is minimized by the use of an additional level in the image pyramid, in which images are up-sampled to twice their original resolution, based on the results of [93] and [28]. Due to these improvements connected with that windowing approach, it is possible to esimate the background motion for a number of test sequences. Experimental results show the very good performance in comparison to recent work even with difficult test material.

Having an improveed GME-algorithm, we come to the next step. Background modeling – meaning the description of the background of a video sequence – is an important research field. Application scenarios in which background modeling is used range from video surveillance systems to video coding.

A so-called background Sprite or single Sprite generated over a certain number of frames can be such a background model. A background Sprite is an image that typically is of large dimensions and depicts only the background pixels of a video sequence. This approach can be extended to so-called multiple Sprites or super-resolution Sprites. In case of multiple Sprites, the video sequence in divided into partitions and for each a background Sprite is generated independently. In case of super-resolution, a higher quality background Sprite is generated.

The potential for using background Sprites for object-based video coding has been summarized in [67]. Furthermore, background modeling is an efficient means for video object segmentation. Various approaches using single or multiple background Sprites in a background subtraction method have demonstrated that this kind of background model is very promising [12], [36]. However, the mapping of pixel content from various frames in a scene into a single Sprite or a collection of multiple Sprites may cause severe geometrical distortion of the background. For reconstruction of the background of a single frame a second mapping needs to be performed, which causes additional distortion, overall resulting in erroneous segmentation masks.

In our new local background Sprite algorithm, a mapping of content from many frames in a scene is performed for each individual frame for background construction. In other words, global motion estimation is performed from many adjacent frames into the frame where the background needs to be reconstructed. No backward mapping is required. Thus, our background Sprites are local and there are as many individual Sprites generated as frames exist in a sequence. This results in a more precise background reconstruction compared to conventional global Sprites. It will be seen later in this work that this technique can also serve for further applications.

2.2 Windowed Global Motion Estimation

2.2.1 The Core and the Tools used

The core global motion estimation algorithm, inspired by [8], is illustrated in Fig. 2.1. The input images are subdivided using an image pyramid to reduce the computational complexity and improve the initialization of the gradient-descent algorithm. The algorithm starts at the lowest resolution of the input images using the phase correlation method to calculate the translational motion parameters [38]. Afterwards, the Gauss-Newton (GN) algorithm using the affine motion model is applied. The achieved parameters then initialize the motion parameters of the perspective model in the next upper stage of the pyramid and the GN-algorithm is applied again. This procedure is repeated through the pyramid until the highest resolution (the original input frames) and the final motion parameters are calculated. A simplified robust M-estimator [69] is applied on the error function, which has to

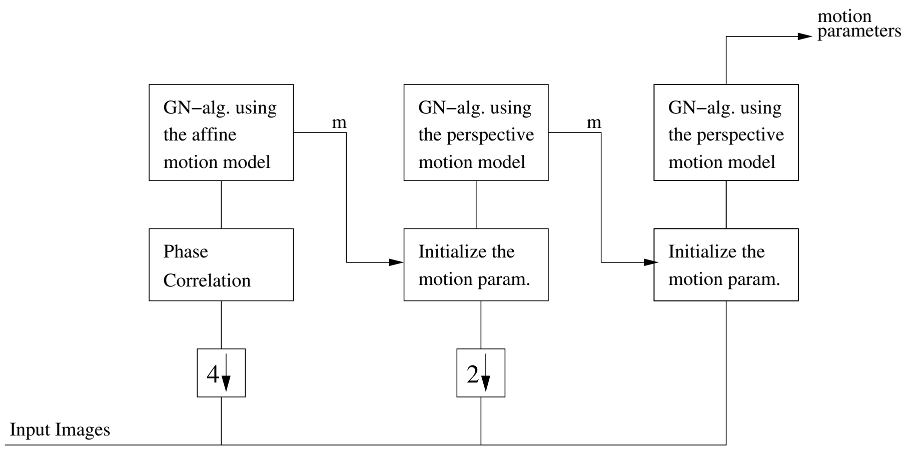

motion parameters
GN–alg. using the affine motion model
m
GN–alg. using the perspective motion model
m
GN–alg. using the perspective motion model
Phase Correlation
Initialize the motion param.
Initialize the motion param.
4
2
Input Images

Figure 2.1: Core Algorithm

be minimized to prevent the influence of outliers. It has been shown that the Gauss-Newton gradient descent algorithm has a very good performance if the start point of the gradient descent is close to the desired minimum [2]. Therefore, the initialization of the translational parameters is very important and the estimation of the phase correlation method is robust and fast with good results [25].

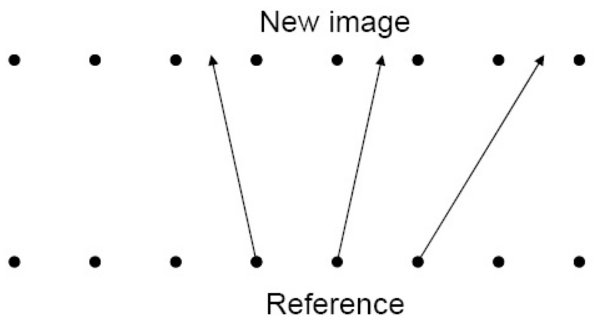


Figure 2.2: Resampling due to warping

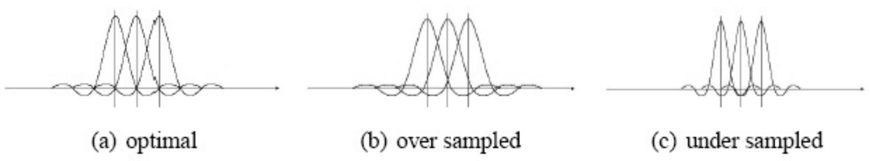


Figure 2.3: Sampling issue (pixel domain)

2.2.2 The Up-Sampling Issue

Due to the warping process, under-sampling can occur, which is illustrated in Fig. 2.2. If the warping process is applied at original input images for the final estimation and we assume that the original input images are optimally sampled, aliasing could appear in the warped image due to possible under-sampling. This aliasing produces an increased error and the performance of the gradient descent algorithm decreases. The robust M-estimator may remove some large errors, but cannot compensate for under-sampling. The design of the binary M-estimator is for prevention of errors due to outliers. Errors in consequence of aliasing in the warped image range through the whole image and affect the estimation process in general. To prevent under-sampling the input images can be up-sampled. Fig. 2.3 shows how up-sampling can avoid aliasing in the pixel domain. In Fig. 2.3, point spread functions are shown for each sampling case. It can be seen that in the over-sampled case (Fig. 2.3 (b)), the point spread functions are more overlapped, which means that they are robust against possible under-sampling. Fig. 2.3 (a) shows the optimally sampled case (assuming the original images). If under-sampling arises here, the point spreads of each pixels are spread like illustrated in Fig. 2.3 (c). As a result, aliasing appears in the warped image. The same issue can be explained in the frequency domain. The over-sampling can prevent the under-sampling during the image warping, which could cause aliasing, so long as the magnitude of the over-sampling is larger than that of the under-sampling.

To prevent that problem, the input images are up-sampled before the last step of the motion parameter calculation. This means that over-sampled versions are used for the last estimation step. The 7-tap wavelet filter is taken for the interpolation process because it has a very good approximation to the optimal *sinc*-function. Figure 2.4 shows a block diagram where the up-sampling is used in the final step.

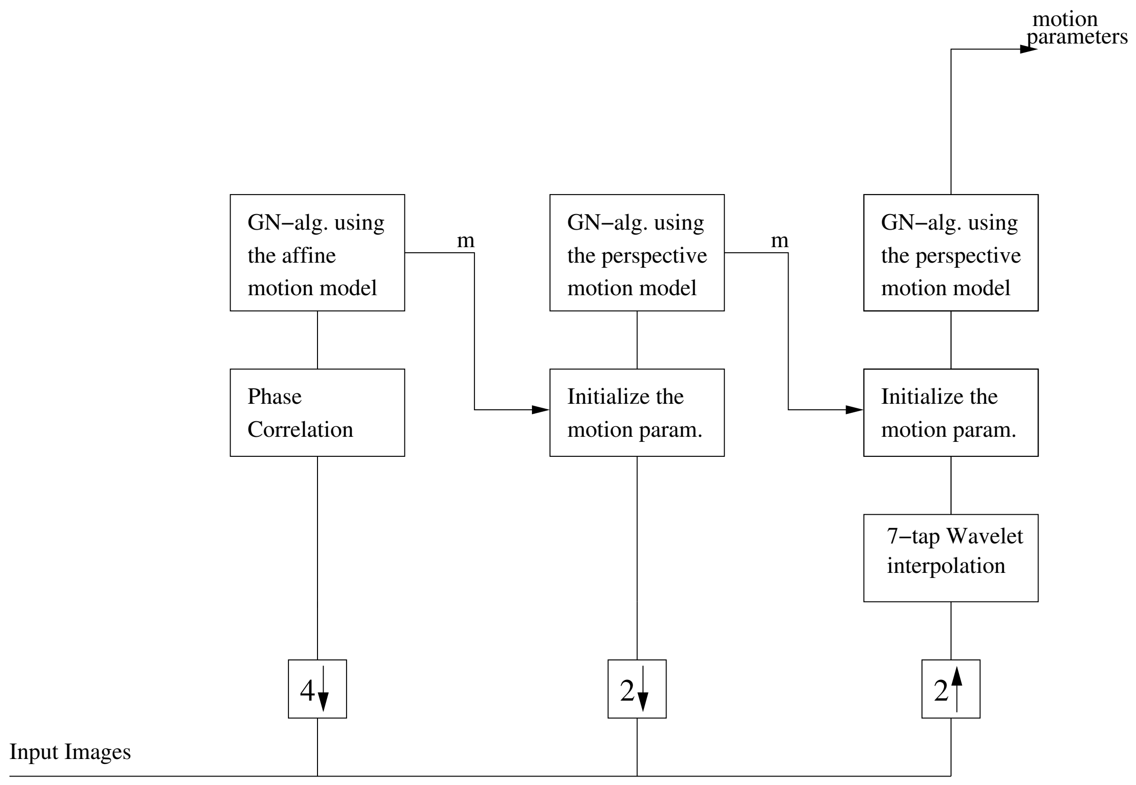
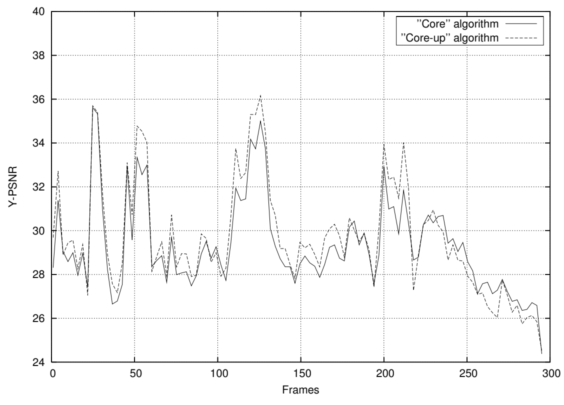
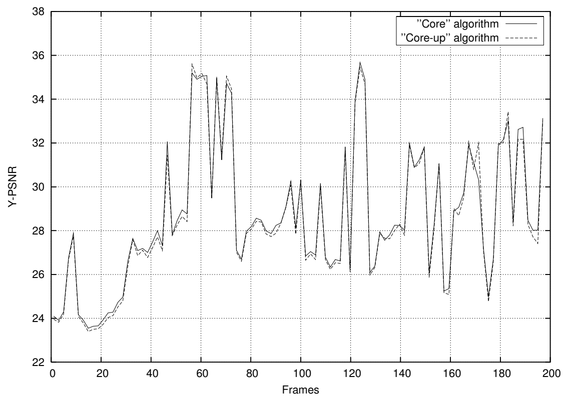

motion parameters
GN−alg. using the affine motion model
m
GN−alg. using the perspective motion model
m
GN−alg. using the perspective motion model
Phase Correlation
Initialize the motion param.
Initialize the motion param.
7−tap Wavelet interpolation
4↓
2↓
2↑
Input Images

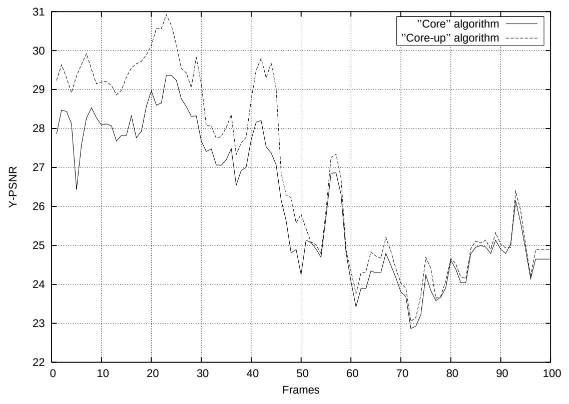
Figure 2.4: Core algorithm with up-sampling



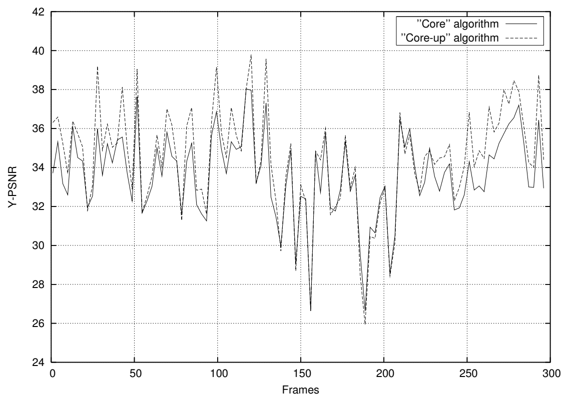
(a) “Stefan”



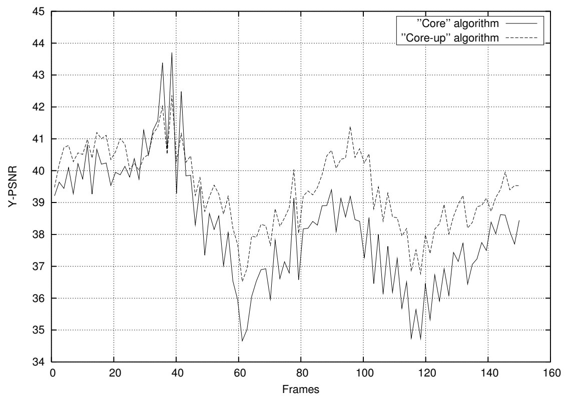
(b) “Biathlon”



(c) “Mobile & Calendar”



(d) “Foreman”



(e) “Monaco”

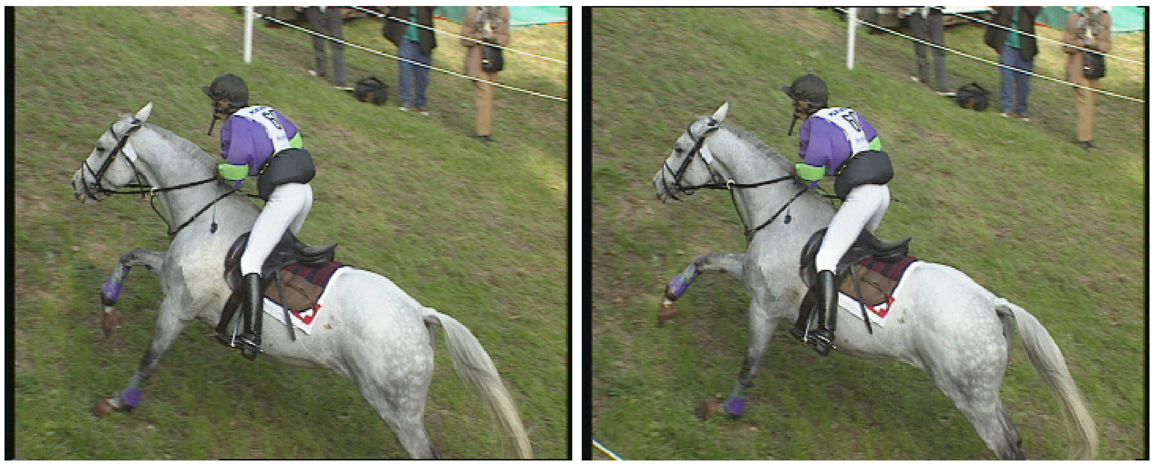
Figure 2.5: Background PSNR-curves comparing core alg. and core alg. with up-sampling

The global motion estimation algorithm with up-sampling is compared to the state-of-the-art core algorithm. A set of test video sequences is used to show the advantage of the up-sampling step discussed above. The first is the well-known “Stefan” sequence (352x240 pixel, 300 frames). Here, large camera motion occurs with one foreground object and a dominant background object. Using this kind of sequence, the accuracy of the frame-by-frame global motion estimation can be examined without any interference of the foreground object. The

Table 2.1: Mean Background-PSNR values of the compared algorithms

Avg. PSNR in dB	Core Algorithm	Core-up Algorithm	Smolic Algorithm
“Stefan”	29.37	29.75	28.49
“Biathlon”	28.60	28.49	–
“Mobile&Calendar”	26.17	26.92	–
“Foreman”	33.74	34.55	–
“Monaco”	38.29	39.44	–

second sequence is similar to the “Stefan”-sequence. It is also a sport sequence called “Biathlon” (352x288 pixel, 200 frames) captured from the first German TV station (“Das Erste”). The third test video is also well-known, “Mobile&Calendar” (352x288 pixel, 100 frames). Several moving foreground objects occur here with high texture in the background. The fourth test sequence is the MPEG-sequence “Foreman” (352x288 pixel, 300 frames). Here, we also have a large background occlusion due to the large foreground object in the first part of the sequence. In the second part, there is a camera pan without any forground objects. The last sequence called “Monaco” (352x288 pixel, 150 frames) shows a camera pan over the harbor of Monaco without any foreground objects. Short-term motion-compensated PSNR-values are measured and compared for each algorithm. For an accurate comparison of the background pixels, ground truth is available for each video sequence separating the foreground object pixels from the background. We would like to emphasize that the calculation of the motion parameters is accomplished without any foreground/background separation. The foreground/background masks are only used for an accurate calculation of the PSNR-value using the error image achieved. Figure 2.5 shows the frame-to-frame PSNR-curves over the five considered test sequences considering the up-sampling in the final registration step. It can be seen that except for “Biathlon” we achieve an overall higher estimation performance. The highest improvement can be calculated with test sequence “Monaco” (up to 1.15 dB) (see Tab. 2.1). This means that we are able to improve the estimation process by applying the up-sampling step in sequences with small, large, and no foreground objects. The reason for the “Biathlon”-case is the content of this sequence. Here, we have wide regions with no texture (snow). Applying the phase correlation here for initialization leads to misestimations in the early stage of the algorithm, which makes it hard for the gradient descent algorithm to converge into the global minimum. This issue is discussed in more detail later in this section.



(a) Frame 100 (b) Frame 101

Figure 2.6: Frame 100 and 101 of sequence “Horse”

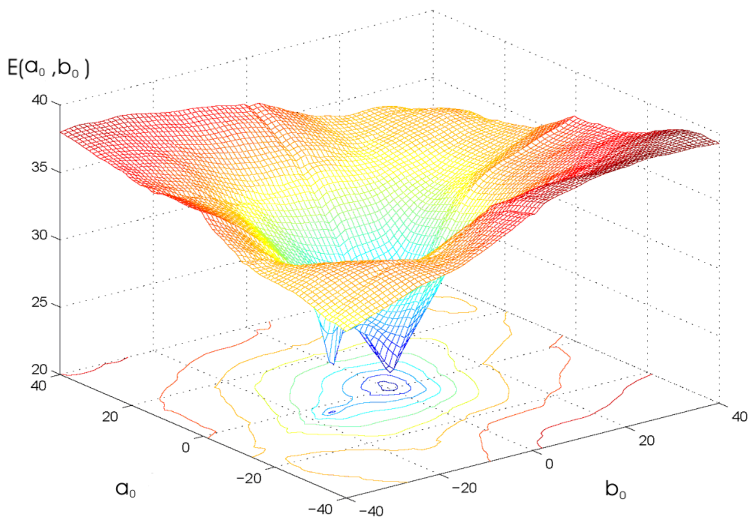


Figure 2.7: Error function using the 2-parameter motion model

2.2.3 The Windowing Approach

The use of statistical robust estimation methods as shown in [69] and [8] fails if, for example, large foreground objects occur. In this case, it is possible that the background is no longer the largest object, requiring the GN-algorithm to find a minimum which is not global. The problem is illustrated in Fig. 2.6 and Fig. 2.7. The example shows two consecutive frames of test sequence “Horse”. A two-dimensional error surface is built using the 2-parameter translational motion model.

In this scene the camera follows the foreground object and therefore the global minimum lies in the center of the translational coordinates. The background object moves relative to the camera and produces a local minimum beside the global one. To obtain the global camera

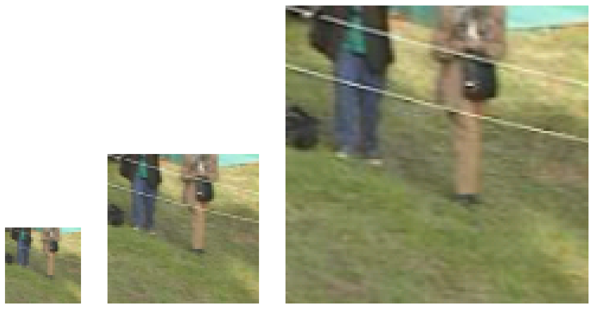


Figure 2.8: Best block match for Frame 101 ("Horse")

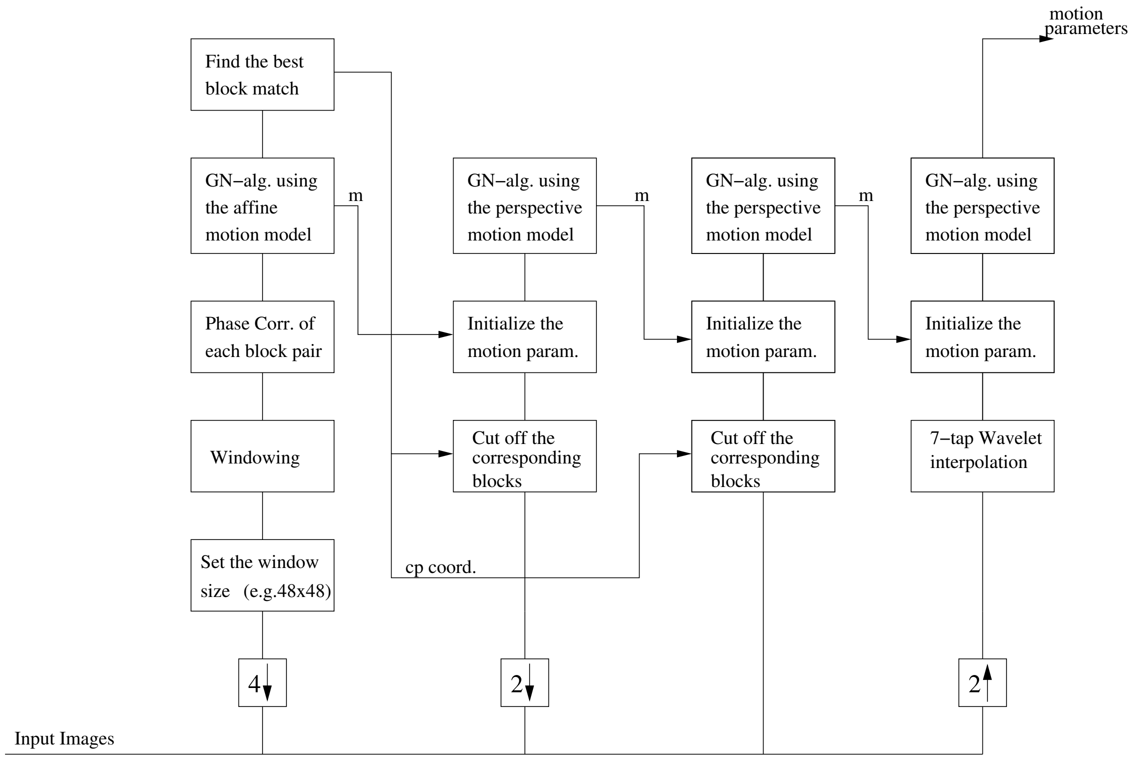
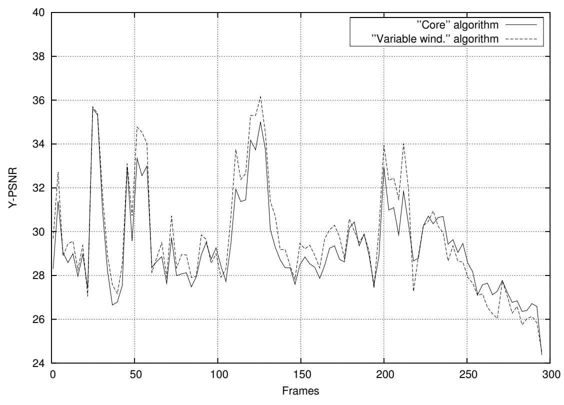
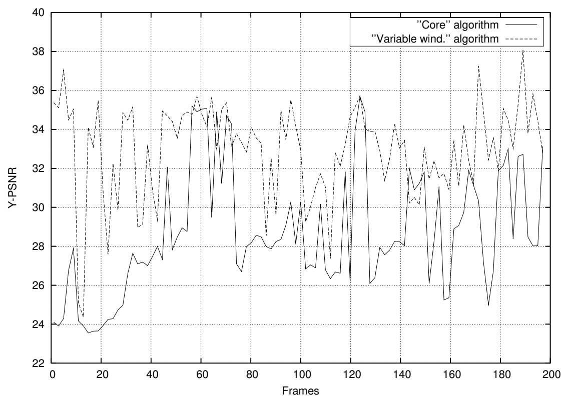

motion parameters
Find the best block match
GN−alg. using the affine motion model
m
GN−alg. using the perspective motion model
m
GN−alg. using the perspective motion model
m
GN−alg. using the perspective motion model
Phase Corr. of each block pair
Initialize the motion param.
Initialize the motion param.
Initialize the motion param.
Windowing
Cut off the corresponding blocks
Cut off the corresponding blocks
7−tap Wavelet interpolation
Set the window size (e.g.48x48)
cp coord.
4↓
2↓
2↑
Input Images

Figure 2.9: Windowed global motion estimation algorithm

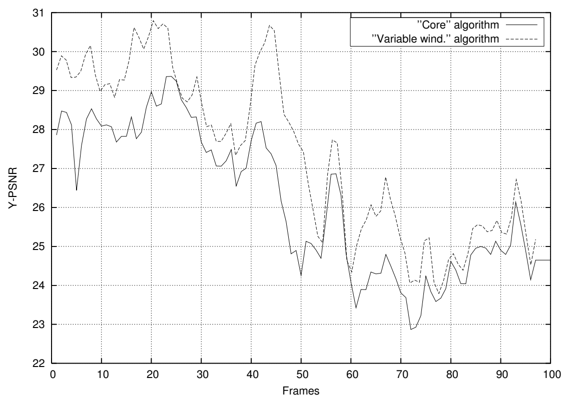
motion, the gradient descent algorithm has to be initialized close to that local minimum. This is achieved applying a windowing technique at the coarser levels of the image pyramid. The input images on the coarsest level are divided into blocks. The blocks are arranged with overlapping of 3/4 of the block size. To find the best match, phase correlation [38] and gradient descent using the affine motion model are applied on each block. Then the compensation error block is computed. Only the block with the lowest error is taken for further processing. The matching can also be achieved using only phase correlation to accelerate the algorithm. However, the use of phase correlation combined with the gradient descent algorithm produces more accurate results and is more stable. In the next level the gradient descent is only executed for the found block. Fig. 2.8 shows the blocks used for the calculation of the motion parameters throughout the image pyramid for the example given in Fig. 2.6.



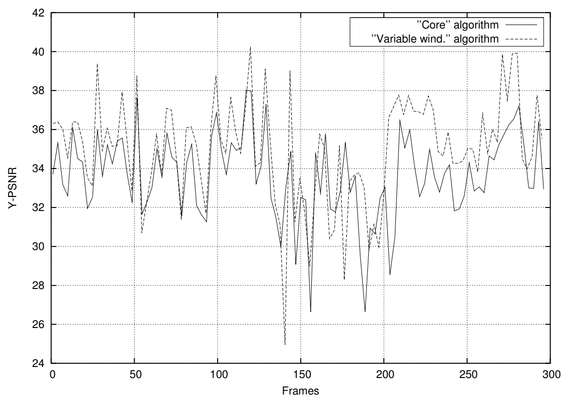
(a) "Stefan"



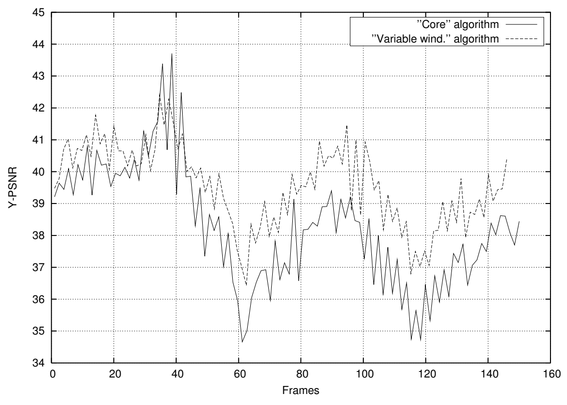
(b) "Biathlon"



(c) "Mobile & Calendar"



(d) "Foreman"



(e) "Monaco"

Figure 2.10: Background PSNR-curves comparing core alg. and window-based alg. with optimal window size

It can be seen that the best block match belongs to the background objects. Thus, the final gradient descent algorithm at the finest level, i.e. the up-sampled image level, can be initialized by the motion parameters obtained with the blocks to find the local minimum beside the global one.

The techniques described above lead to the windowed global motion estimaton algorithm.

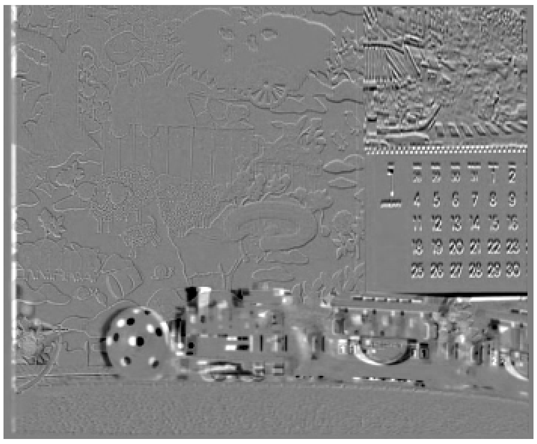
The algorithm using windowing (fixed size) is depicted in Fig. 2.9. Two versions of this algorithm are considered for the experimental evaluation, one with fixed window size and the other with automatic window size alignment. For the considered test sequences, two available window sizes lead to successful results. The updating step with a new window size has to be accomplished only a few times when an outlier is detected. This means that the computational complexity does not increase too much in comparison to the use of a fixed window size or to the core algorithm. The most costly calculation is the parameter estimation on up-sampled input images at the last step of the algorithm. However, it has been found that this step is necessary for a more precise calculation. Furthermore, the motion parameters are well-initialized due to the computation at the latter stages of the pyramid and the calculation at the finest stage takes only a few steps more.

For the evaluation, the test sequences “Stefan” and “Biathlon” for the small foreground objects case and “Mobile&Calendar” and “Foreman” with large foreground objects as well as sequence “Monaco” with no foreground objects are taken into account. Figure 2.10 depicts the comparison of the core and the proposed algorithm including all features. For the proposed algorithm, three window sizes are considered, that are 32x32, 40x40, 48x48. We calculated the motion parameters using these three different window sizes for each test sequence. Table 2.2 lists the mean PSNR-values for each window size. The last column contains the mean values where the optimal window size was taken depending on the PSNR for each frame pair. Additionally, samples of error images are shown to emphazise the improvement especially due to the use of windowing in Fig. 2.11.

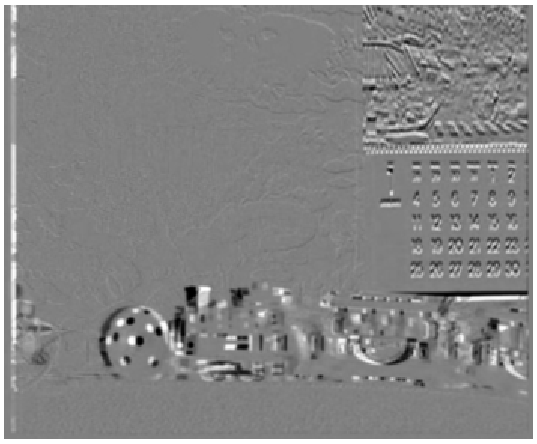
Table 2.2: Mean Background-PSNR values of different window sizes for the proposed algorithm

Avg. PSNR in dB	Wind.-size 32x32	Wind.-size 40x40	Wind.-size 48x48	Wind.-size (optimal)
“Stefan”	29.78	29.86	29.83	29.96
“Biathlon”	32.04	29.55	28.74	33.01
“Calendar”	27.28	27.24	27.27	27.36
“Foreman”	34.08	34.07	34.08	34.90
“Monaco”	39.21	39.24	39.19	39.54

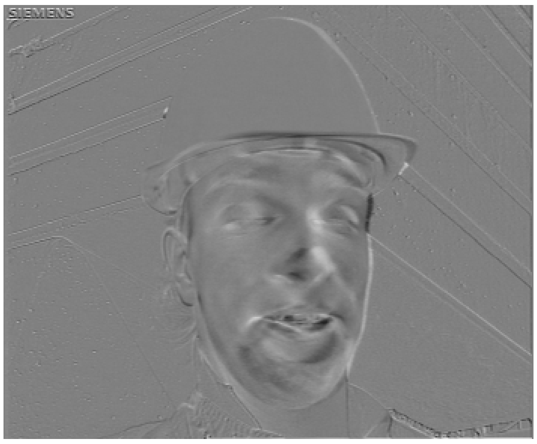
These results show that the proposed algorithm brings the best and robust performance in comparison to recent approaches. The problem of the fixed-size windowing approach is also obvious. By using a fixed window size it can occur that the gradient descent algorithm is sub-optimally initialized, and converges to a non-desired minimum. This leads to lower mean PSNR-values. Therefore, we show the improvement of using the optimal window size for each image pair. Optimal means that the window size taken leads to the best PSNR-value for the current image pair. The PSNR-values are calculated using the ground truth masks. Having these optimal window sizes, it can be seen that the global motion estimation using the proposed algorithm leads to very good results. These experiments show the necessity of using windowing especially for sequences with large foreground objects. The biggest challenge is to include in the proposed algorithm automatic window size selection during the estimation process. The first approach for an online automatic window size selection algorithm is given next.



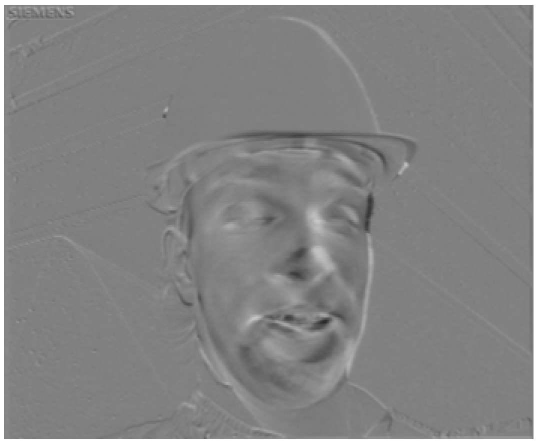
(a) Error frame (43) core alg. “Calendar”



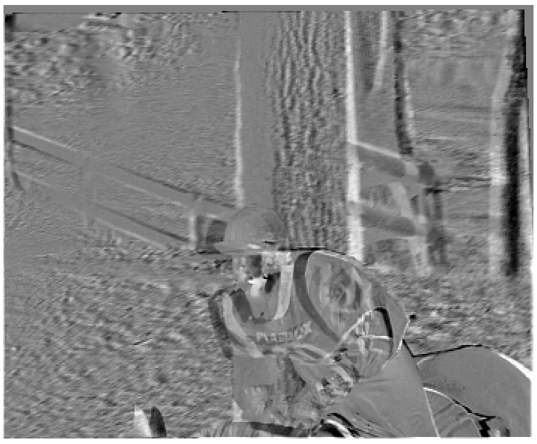
(b) Error frame (43) wind. alg. (32x32) “Calendar”



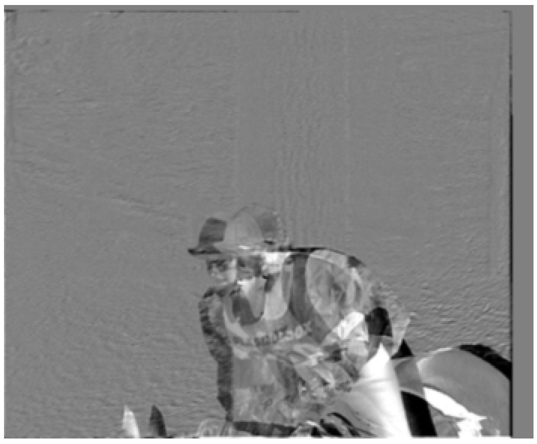
(c) Error frame (1) core alg. “Foreman”



(d) Error frame (1) wind. alg. (32x32) “Foreman”

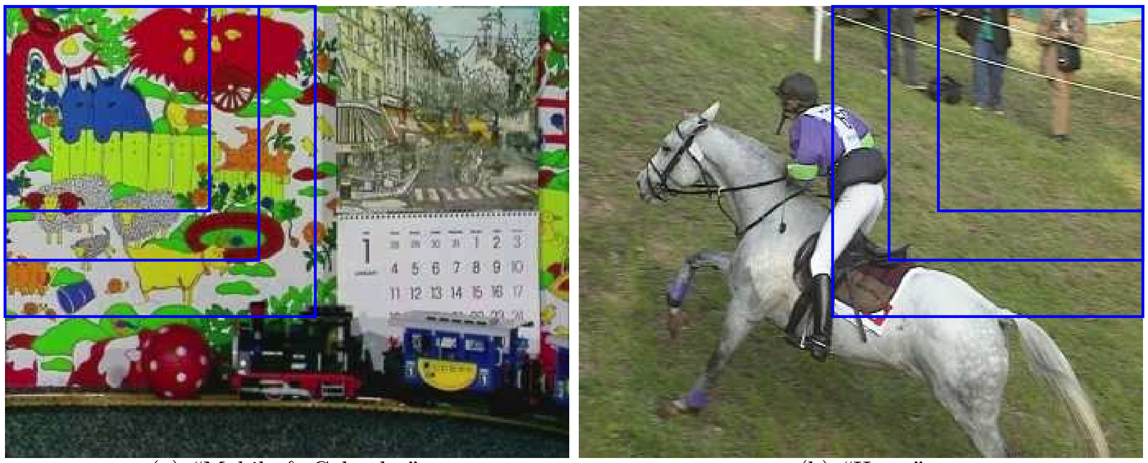


(e) Error frame (29) core alg. “Horse”



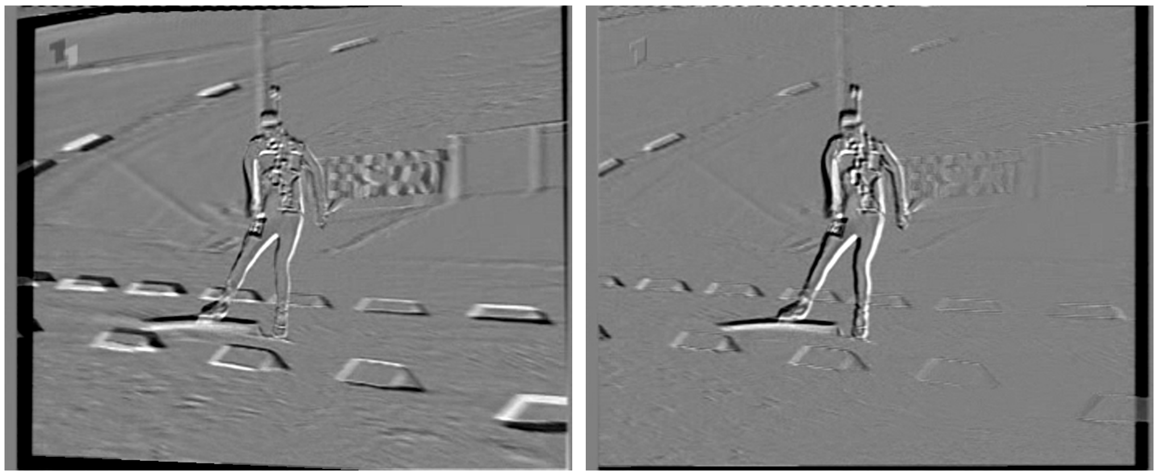
(f) Error frame (29) wind. alg. (32x32) “Horse”

Figure 2.11: Visualization of the best window match (32x32, 40x40, or 48x48)



(a) "Mobile & Calendar" (b) "Horse"

Figure 2.12: Visualization of the best window match (32x32, 40x40, or 48x48)



(a) Error frame (outlier with fixed window size) (b) Error frame (window size changed)

Figure 2.13: Comparison of a fixed window size and variable window size at the outlier case (frame 94, "Biathlon" sequence)

2.2.4 Online-automatic window sizes

The failure of the initialization can be prevented if variable window sizes are used. Two sizes (32x32 and 40x40 pixel) are used for the first approach. The algorithm starts with one window size and the task is to detect the appearance of an outlier online during the calculation of a sequence. The RMSE-value (Root Mean Square Error) of the current image pair is calculated at the third stage of the algorithm, at the original size of the input images. If an outlier occurs, this RMSE-value increases significantly in comparison to the previous value. This can be emphasized if the first numerical derivative of the RMSE-values is considered. The calculation is not that difficult because only the RMSE-value of the previous image pair has to be stored. The differential RMSE is then calculated as :

$$RMSE_{diff} = RMSE_{curr} - RMSE_{prev},$$

A threshold defines if a peak in the differential RMSE appears and the algorithm goes back and starts with the second window size for the current image pair. After the calculation of the motion parameters, the window size is changed to the previous size for the next input image pair. The threshold can be calculated using the variance of the differential RMSE:

$$T = \frac{1}{N}\sum_{k=1}^{N}(e_k - \mu)^2,$$

where e_k holds the differential RMSE-value at the point k, μ is the average, and N is the current number of the differential RMSE-values. For the first pair of frames in the sequence, both sizes are applied. The size that produces the lower RMSE is taken for further calculation. This can be extended when more than one example of the test sequence is considered. Figure 2.12 illustrates the case of the best window match. Here, three options of window sizes can be chosen. The algorithm begins with the largest window size if the foreground object is very small. The examples show frames of sequences with larger background occlussion due to one big or several foreground objects. It can be seen that in this case smaller window sizes have a better match to the background.

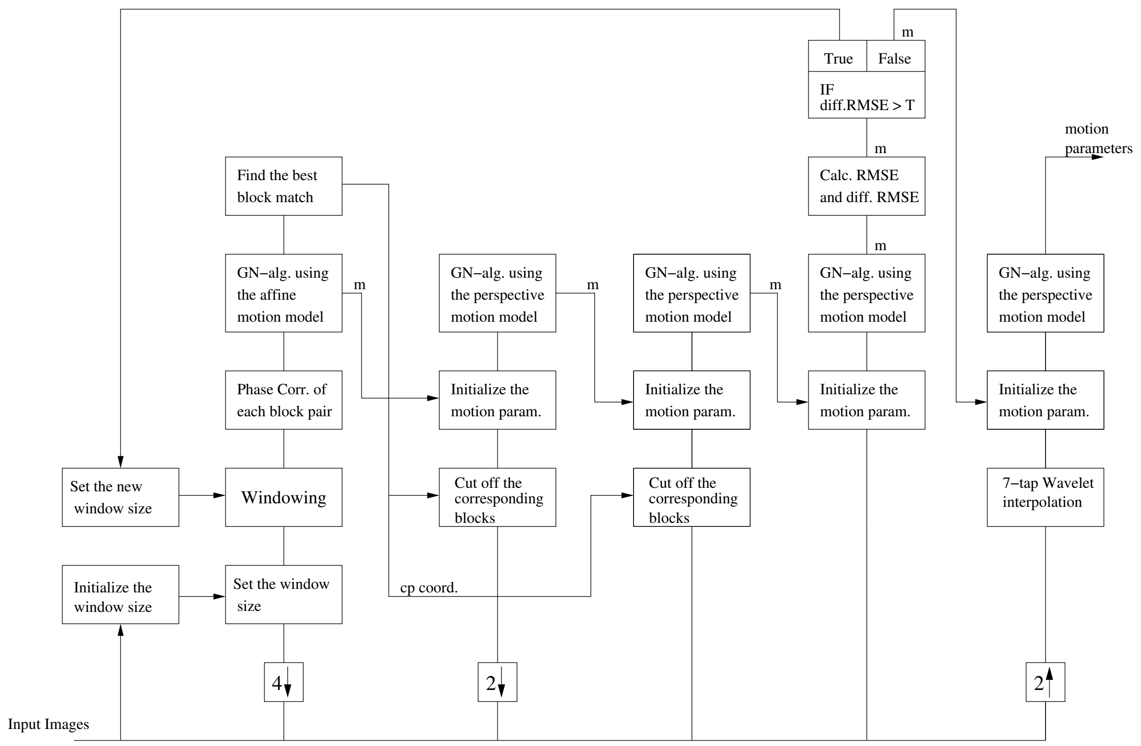

m
True
False
IF diff.RMSE > T
m
motion parameters
Find the best block match
Calc. RMSE and diff. RMSE
m
GN−alg. using the affine motion model
m
GN−alg. using the perspective motion model
m
GN−alg. using the perspective motion model
m
GN−alg. using the perspective motion model
GN−alg. using the perspective motion model
Phase Corr. of each block pair
Initialize the motion param.
Initialize the motion param.
Initialize the motion param.
Initialize the motion param.
Set the new window size
Windowing
Cut off the corresponding blocks
Cut off the corresponding blocks
7−tap Wavelet interpolation
Initialize the window size
Set the window size
cp coord.
4
2
2
Input Images

Figure 2.14: Windowed global motion estimation algorithm with arbitrary window sizes

Figure 2.13 shows an example for estimation improvement at an outlier case. The automatic window size technique described is now integrated in the present windowed global motion estimation algorithm. The modification at the beginning and at the third level of the pyramid brings the new updated algorithm, which can be seen in Fig. 2.14. It has been found experimentally that the aligned windowing technique works best at the third stage of

Table 2.3: Mean Background-PSNR values at the outlier cases

Avg. PSNR in dB	fix window size (40x40)	changed window size
"Stefan"	25.32	29.36
"Mobile"	19.80	31.57
"Biathlon"	24.56	29.94

Table 2.4: Mean Background-PSNR values of the compared algorithms

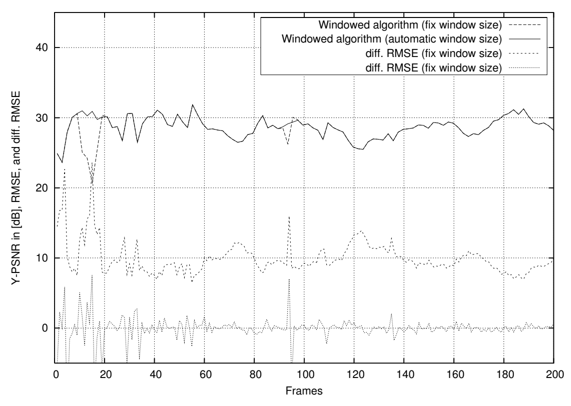
Avg. PSNR in dB	Core Alg.	Smolic Smolic	Proposed (40x40)	Proposed (arbitrary)
"Stefan"	28.08	28.49	29.24	29.52
"Mobile"	28.60	-	31.18	31.37
"Biathlon"	28.44	-	28.89	29.08

the algorithm. The computational complexity does not increase so much because calculating the RMSE-value using initialized motion parameters takes only a few more steps. It can be seen that only a few outliers occur and the re-computation during the algorithm at the outlier-case using a new window size has to be accomplished rarely. This technique using only two different window sizes works very well with the considered test sequences as shown in the experiments.

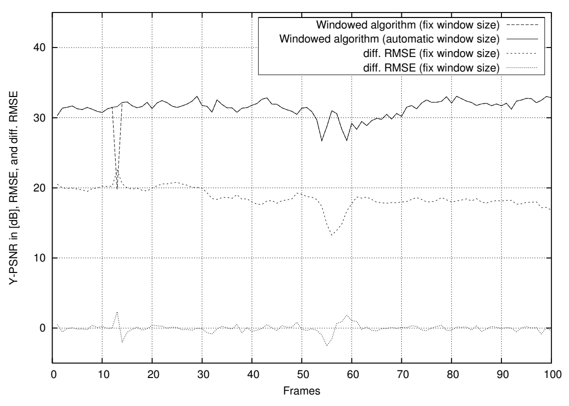
Three test videos are selected to show the performance of the automatic window size approach ("Stefan", "Mobile&Calendar", "Biathlon"). The PSNR-, RMSE-, and differential RMSE-value are computed for each input image pair. The curves are illustrated in Fig. 2.15 for each of the three test sequences. It can be seen that at each sequence several outliers occur when a fixed window size is used. The new algorithm using adaptive window sizes intercepts the outliers. Table 2.3 shows mean PSNR-values for the outlier cases. The updated algorithm significantly improves the performance of the algorithm at this stage. Table 2.4 shows average PSNR-values comparing the proposed and recent algorithms. The average PSNR-value of the proposed algorithm increases up to about 2.7 dB in comparison to recently proposed algorithms.

2.2.5 Considering Initialization Techniques regarding the Video Content

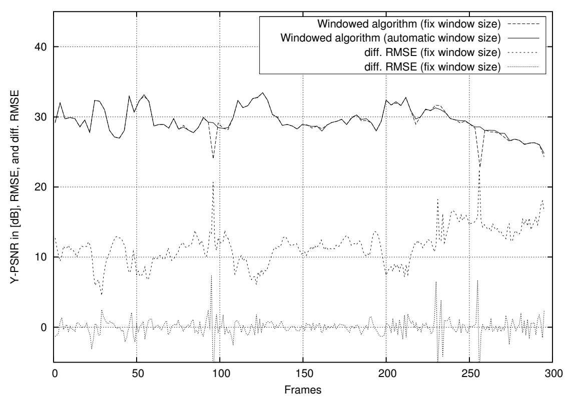
It was already mentioned above that a good initialization is very important for a successful gradient descent algorithm. Until now, the phase correlation technique was used because of its robustness and the low computational cost. However, during the experiments it turned out that the performance of the phase correlation technique depends on the content of



(a) “Biathlon”



(b) “Mobile & Calendar”



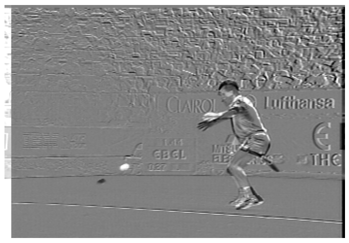
(c) “Stefan”

Figure 2.15: PSNR-curves with and without RMSE-analysis

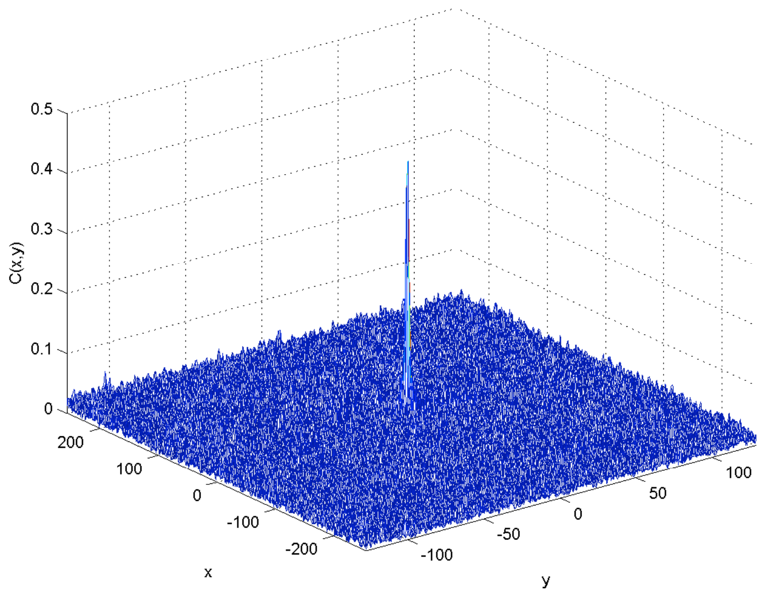
the input frames. For a good estimation of the displacement, the input frames have to contain large regions with high texture resulting in a wide spectrum. If this is not the case the global translational motion cannot be determined robustly. An alternative approach is feature tracking, which is also widely used as an initialization technique for global motion estimation and image registration in general. Now we compare both approaches regarding test videos with different kind of content in the background. For that, the phase correlation technique is briefly introduced next and the problem used with different kinds of background content is developed. Afterwards, a state-of-the-art feature tracking algorithm is described. An experiment is conducted comparing both algorithms estimating the translational motion and as an initialization step for a full global motion estimation algorithm.

(a) Frame 100 (I_1)

(b) Frame 101 (I_2)



(c) Error frame



(d) Correlation function $C(x, y)$

Figure 2.16: Example for calculating the translational motion parameters using the phase correlation

2.2.5.1 Phase Correlation

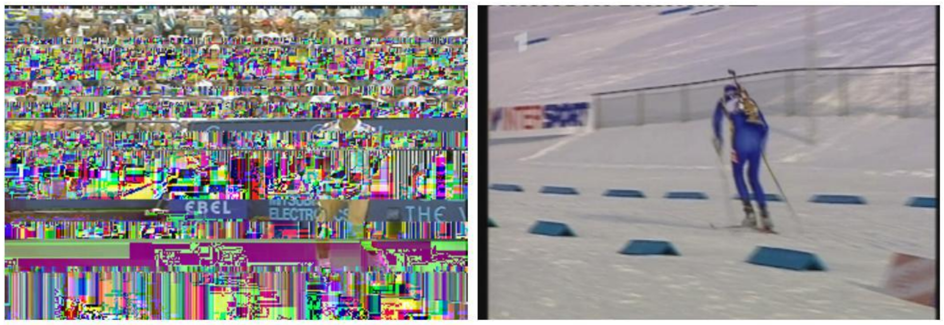
The phase correlation method is derived from the Fourier transform shift theorem [38]. The two images I_1 and I_2 are $2D$-functions. It is assumed that I_2 is a shifted version of I_1:

$$I_1(x + x_s, y + y_s) = I_2(x, y) \tag{2.1}$$

The $2D$-Fourier transforms are computed for both images and using (2.1) and the features of the function $C(x, y)$ can be calculated. The derivation is:

$$\begin{aligned}
\mathcal{F}\{I_1(x + x_s, y + y_s)\} &= \mathcal{F}\{I_2(x, y)\} \\
\mathcal{I}_1(\omega_1, \omega_2)e^{j(\omega_1 x_s + \omega_2 y_s)} &= \mathcal{I}_2(\omega_1, \omega_2) \\
\Rightarrow C(x, y) &= \mathcal{F}^{-1}\Big\{\frac{\mathcal{I}_2(w_1, w_2)}{\mathcal{I}_1(w_1, w_2)}\Big\} = \mathcal{F}^{-1}\{e^{j(w_1 x_s + w_2 y_s)}\} \\
\Rightarrow C(x, y) &= \delta(x - x_s, y - y_s)
\end{aligned} \tag{2.2}$$

Figure 2.16 illustrates an example. Two consecutive images taken from the test sequence “Stefan” are shown in Fig. 2.16 (a) and (b). In Fig. 2.16 (c) the error frame is depicted



(a) Frame 130, “Stefan” (b) Frame 100, “Biathlon”

Figure 2.17: Example images with high and low texture

after applying the calculated translational motion parameters using the phase correlation method. Figure 2.16 (d) shows the correlation function $C(x, y)$ for that example. The peak value represents the translational shift.

In this case, estimation with full-pixel accuracy is provided. An approach for subpixel estimation using the phase correlation method is proposed in [13]. In this work, phase correlation is used as initialization technique. Therefore, only full-pixel accuracy is required. It has two big advantages. Firstly, the computation is fast. Second, the estimation is robust against aliasing due to interlaced videos and against noise [13]. Furthermore, it is robust against two or more moving objects.

All the advantages of this technique are possible if the spectrum of one input image I_2 is nearly the shifted version of the reference image I_1. This is given when the spectra of both images include a wide range of fequencies, i.e. content with high texture such as in the “Stefan”-sequence Fig. 2.17 (a). However, frames including wide regions without high textures, such as in the sequence “Biathlon” (the snow), the dominant values of the spectra of both input images are in the lower bound, which are not shifted and only a few high edges are shifted according to the camera motion (see Fig. 2.17 (b)). In this case, the translational coordinates of the cross correlation function are zero because the dominant lower bound spectrums of the input images are the same. To tackle this problem, we have to consider a feature-based techniqe to find the right camera motion also if only small regions with high texture are available.

2.2.5.2 Feature Tracking

As described above, in some video sequences the use of phase correlation as initialization leads to poor results in the final gradient descent-based global motion estimation algorithm. Therefore, we tackle this problem applying a feature tracking approach. We use a state-of-the-art algorithm proposed by Kanade, Lucas, Shi, and Tomasi (KLT) [47], [80], [63].

The KLT-algorithm is based on feature windows which are tracked using a gradient descent algorithm. The evaluation of the sum of squared differences is taken for the error measurement during the tracking steps. For the gradient descent algorithm, the Newton-method is used instead of the well-known Gauss-Newton, which is the core algorithm for the pixel-based global motion estimation method used in this work. For the feature tracker, only a set of pre-selected features are used for the gradient descent. So it is possible to apply the full Newton-method including calculating the complete Hessian matrix. In general, the algorithm assumes that only slow motion occurs between two successive images in a natural video sequence. Therefore, the translational model is applied in the first place. Later, if more and more images are taken into account for the feature tracking, a higher-order motion model, e.g. the affine model, is used. In this work, the KLT-algorithm is applied as an initialization method to estimate the displacement of the two input images considered. For that, only the translational motion model is used.

The selection of the feature windows is the second very important task beside the tracking. Kanade et. al define :

"a good feature is one that can be tracked well".

That means, a feature window is good if the matrix (also structure tensor)

$$Z = \begin{pmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{pmatrix} \tag{2.3}$$

is above the noise level of the image and well-balanced, with

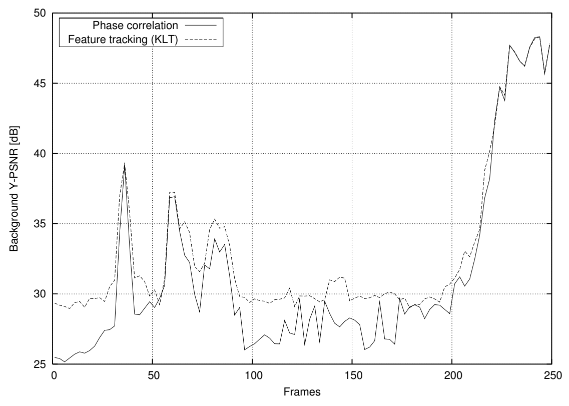
$$g_x^2 = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} g_x^2(i,j) \tag{2.4}$$

$$g_y^2 = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} g_y^2(i,j) \tag{2.5}$$

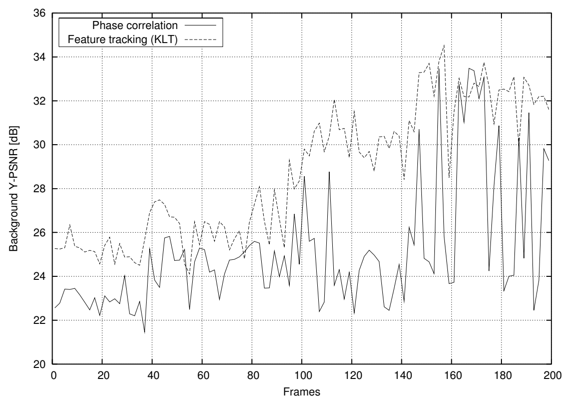
$$g_x g_y = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} g_x(i,j) \cdot g_y(i,j) \tag{2.6}$$

where $g_x(i,j)$ and $g_y(i,j)$ are the gradients in x- and y-direction and PxQ is the size of the feature window. The requirement of the noise level means that both eigenvalues of matrix Z are high. Well-balanced means that the eigenvalues are not far away from each other. For example, two low eigenvalues indicate a relatively constant intensity profile. A high and a low eigenvalue describe a uni-directional pattern. Only two high eigenvalues mean edges or more generally, a good feature to track. Thus, a feature window is used if its eigenvalues λ_1 and λ_2 of the gradient matrix Z lie above a pre-defined threshold λ_{Th}.

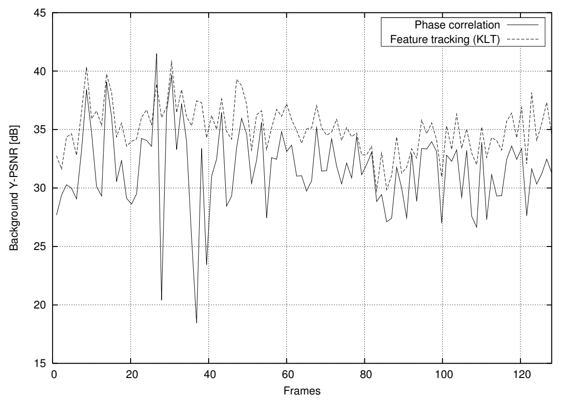
In this work, initializing is performed choosing the N best feature windows of a reference frame in the first step. The next step is the tracking of these feature windows using the KLT-feature tracking algorithm. Feature windows are lost if occlusion occurs or if the feature windows are out of the frame range. That means, after this tracking step there exist a number of translational motion vectors lower or equal to N. If only small foreground objects occur in the video and a high number of features have been tracked, there should only be a few outliers with respect to the main motion direction. These outliers are reduced using the well-known state-of-the-art RANSAC-algorithm. The mean value of the remaining motion vectors is used to initialize the global motion estimation algorithm using a translational motion model.



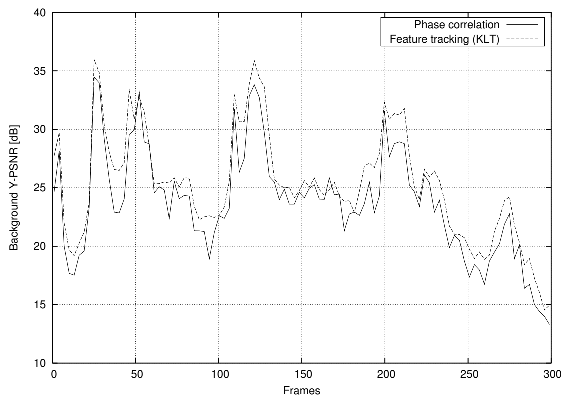
(a) "Allstars"



(b) "Biathlon"



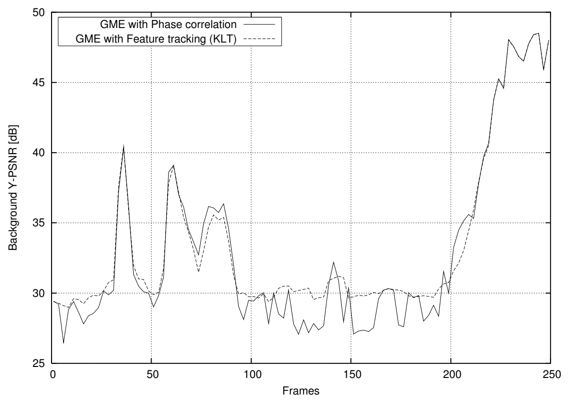
(c) "Mountain"



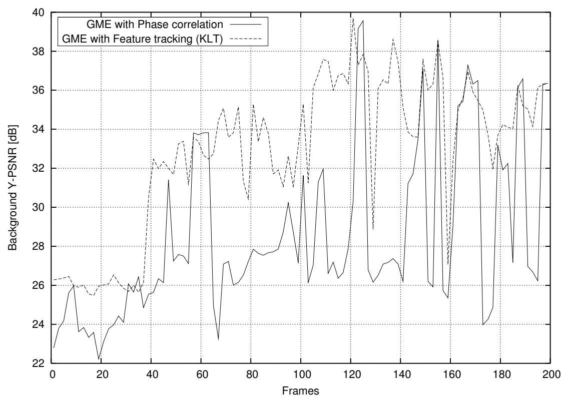
(d) "Stefan"

Figure 2.18: Comparison of phase correlation and feature tracking for translational global motion estimation

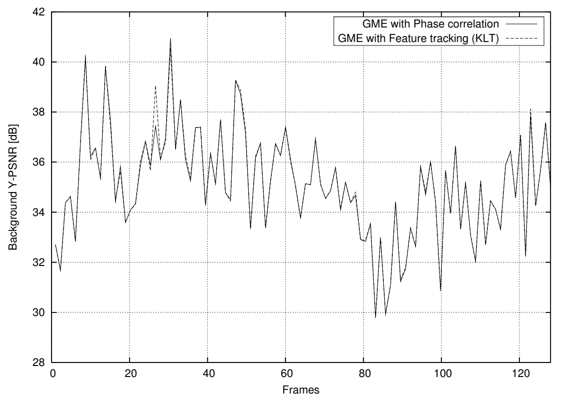
Now, the performance of phase correlation and KLT-feature tracking for initialization are compared. For that, a short-term motion estimation and compensation applied on various video sequences are determined. First, only phase correlation (PC) and feature tracking (KLT) are used for the estimation, respectively. Second, phase correlation (GME/PC) and feature tracking (GME/KLT) are used to initialize the global motion estimation algorithm described above. The performance of both algorithms is evaluated by use of two test video sequences with highly textured content ("Stefan" and "Mountain") and use of two test sequences with low textured content ("Allstars (cif)" and "Biathlon"). The results are shown in Fig. 2.18, Fig. 2.19, and Tab. 2.5, respectively. Visual examples of the results are also provided in Fig. 2.20. As expected, the performance of both techniques is nearly the same for highly textured content. However, it can be seen that the KLT-algorithm outperforms the PC-algorithm significantly applied to test sequences with low textured content. The results between both approaches are equal if the camera is still (e.g. from frame 220, sequence “Allstars (cif)”) and no translational initialization is needed (Fig. 2.18 (a) and Fig.2.19 (a)). It can also be observed that initialization techniques using the translational model have problems if high zoom motion takes place. Here, the feature tracker can be extended by use of the affine motion model which possibly improves the initialization. Furthermore, an interesting issue is the relatively high mean PSNR-value using the KLT-feature tracker without additional gradient descent in sequences “Allstars (cif)”, “Biathlon”, and “Mountain” (Tab. 2.5).



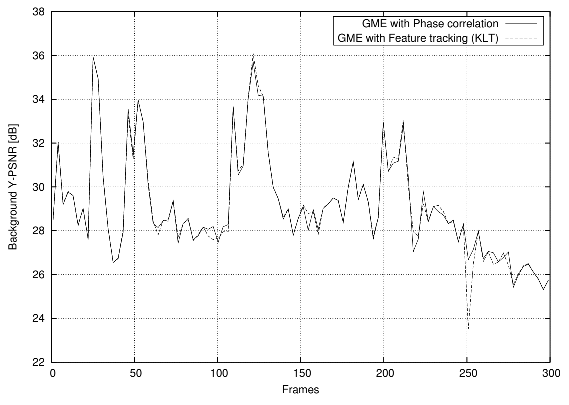
(a) “Allstars”



(b) “Biathlon”



(c) “Mountain”

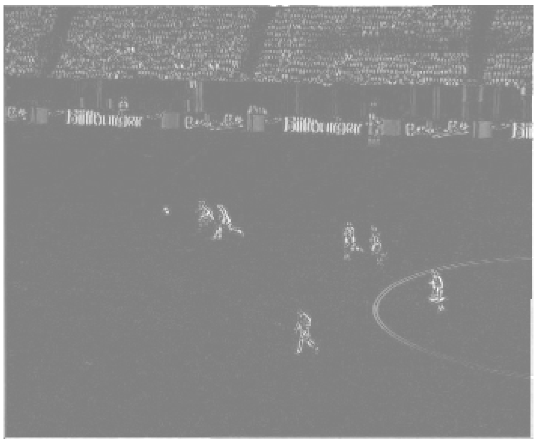


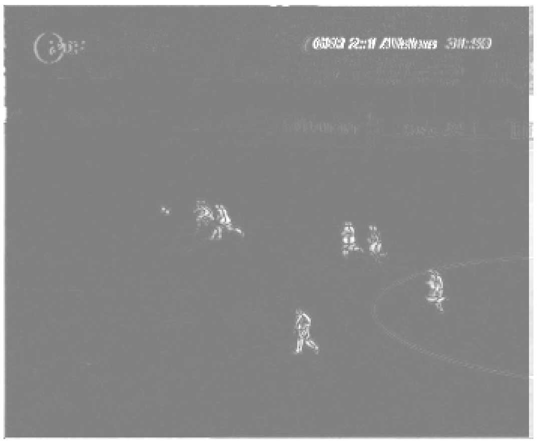
(d) “Stefan”

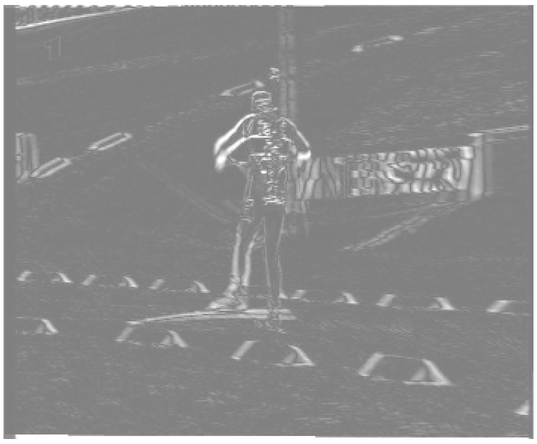
Figure 2.19: Comparison of phase correlation and KLT as initialization for a full global motion estimation algorithm

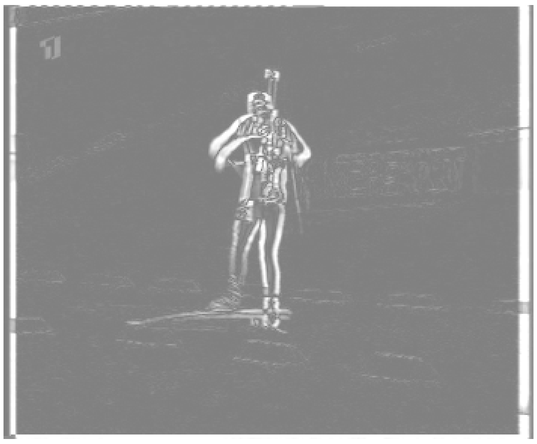
2.2.6 Discussion and final enhanced global motion estimation algorithm

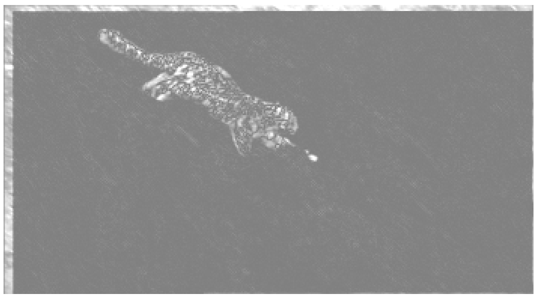
The examination of accurately estimating the global motion between two consecutive frames of a video sequence leads to the final algorithm. Improvements are developed in this work

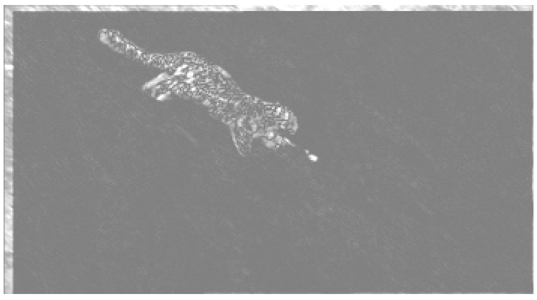

(a) GME using PC error frame 130, “Allstars”

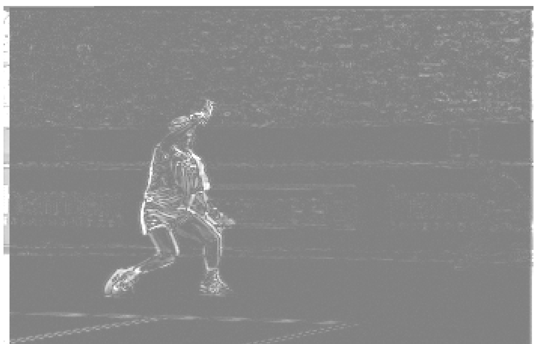

(b) GME using KLT error frame 130, “Allstars”


(c) GME using PC error frame 140, “Biathlon”


(d) GME using KLT error frame 140, “Biathlon”


(e) GME using PC error frame 80, “Mountain”


(f) GME using KLT error frame 80, “Mountain”


(g) GME using PC error frame 200, “Stefan”

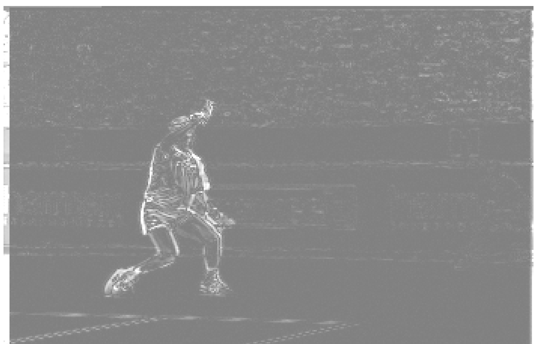

(h) GME using KLT error frame 200, “Stefan”

Figure 2.20: Example error frames of short-term frame-by-frame estimation

	PC	KLT	GME/PC	GME/KLT
"Allstars"	31.2063	32.9973	32.8629	33.3948
"Biathlon"	25.2105	28.7308	28.8015	32.7583
"Mountain"	31.7076	35.1628	35.1883	35.2114
"Stefan"	23.7608	25.3430	29.1280	29.1117

Table 2.5: Mean Background-PSNR values of short-term frame-by-frame estimation

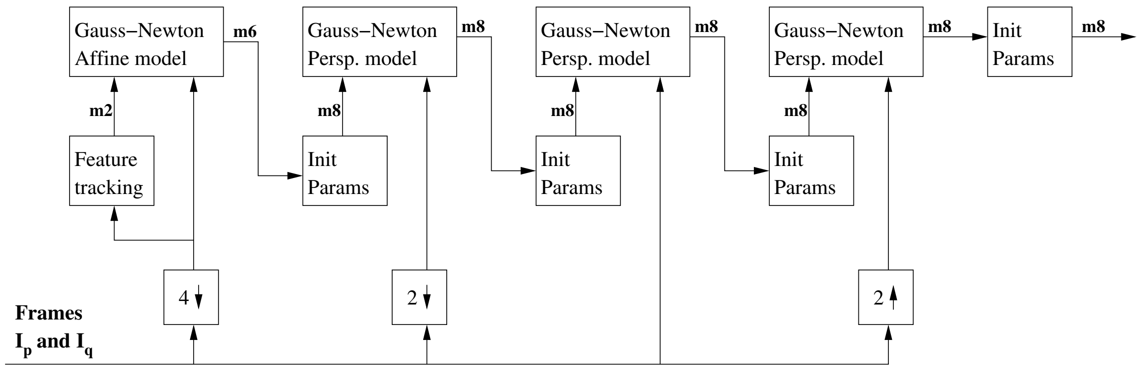


Figure 2.21: Global motion estimation algorithm

regarding the computational complexity, initialization, and tackling the problem of large background occlusions. From here, sequences are considered which contain large camera movements and the ratio between the foreground objects pixels and the background objects pixels is low. Those sequences often take place for example in sport broadcasts and documentary as can be seen in the test data sets later in this work. Therefore, the final global motion estimation algorithm includes all developed features except the windowing because this is only needed if large background occlusions occur. The main application of this work is to find alternative video coding approaches for increasing the compression performance while holding or improving the visual quality. The background Sprite-based approach mainly considered has the best performance in the conditions described. The complete algorithm is depicted in Fig. 2.21 and is summarized next.

The algorithm first generates a 4-step image pyramid for the two frames to register. The image pyramid contains the original frames, two downsampled versions and one in the upsampled domain. For downsampling a 5-tap Le-Gall wavelet filter is used and for upsampling a 7-tap Daubechies wavelet filter. The first gradient descent step is performed on the coarsest resolution and is initialized with a translational motion model using the KLT-feature tracker. The algorithm then performs a gradient descent step in every other layer of the image pyramid using the motion parameters from the step before as initialization. It has been shown that this configuration provides best results in comparison to prior art.

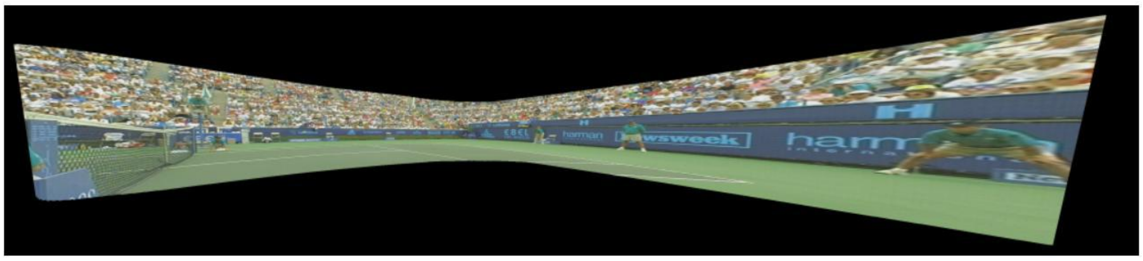
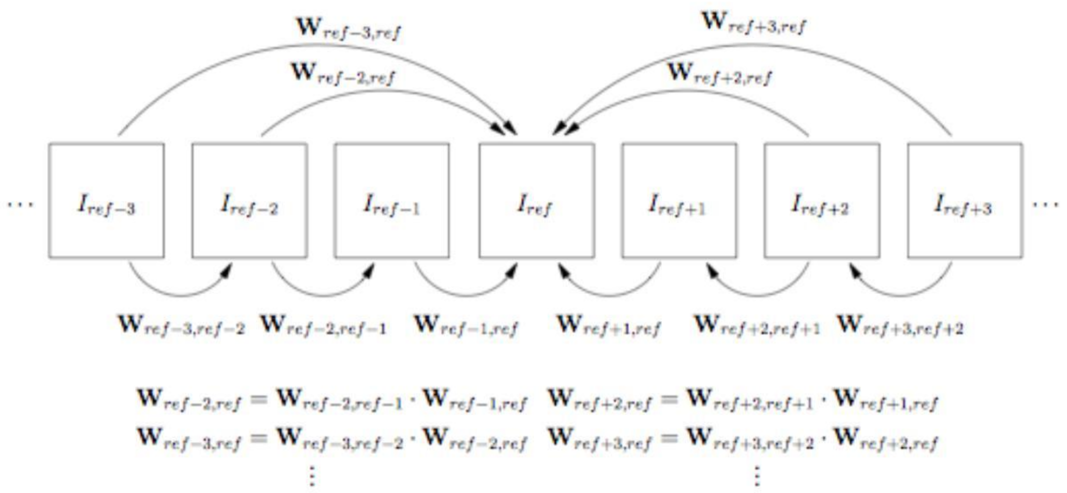


Figure 2.22: Single Sprite, sequence “Stefan”, reference frame 253


$\mathbf{W}_{ref-3,ref}$
$\mathbf{W}_{ref-2,ref}$
$\mathbf{W}_{ref+3,ref}$
$\mathbf{W}_{ref+2,ref}$
$\cdots$ I_{ref-3} I_{ref-2} I_{ref-1} I_{ref} I_{ref+1} I_{ref+2} I_{ref+3} $\cdots$
$\mathbf{W}_{ref-3,ref-2}$ $\mathbf{W}_{ref-2,ref-1}$ $\mathbf{W}_{ref-1,ref}$ $\mathbf{W}_{ref+1,ref}$ $\mathbf{W}_{ref+2,ref+1}$ $\mathbf{W}_{ref+3,ref+2}$
$\mathbf{W}_{ref-2,ref} = \mathbf{W}_{ref-2,ref-1} \cdot \mathbf{W}_{ref-1,ref}$ $\mathbf{W}_{ref+2,ref} = \mathbf{W}_{ref+2,ref+1} \cdot \mathbf{W}_{ref+1,ref}$
$\mathbf{W}_{ref-3,ref} = \mathbf{W}_{ref-3,ref-2} \cdot \mathbf{W}_{ref-2,ref}$ $\mathbf{W}_{ref+3,ref} = \mathbf{W}_{ref+3,ref+2} \cdot \mathbf{W}_{ref+2,ref}$
$\vdots$ $\vdots$
Figure 2.23: Creation of long-term motion parameters

2.3 Local Background Sprite Generation

2.3.1 General Background Sprites

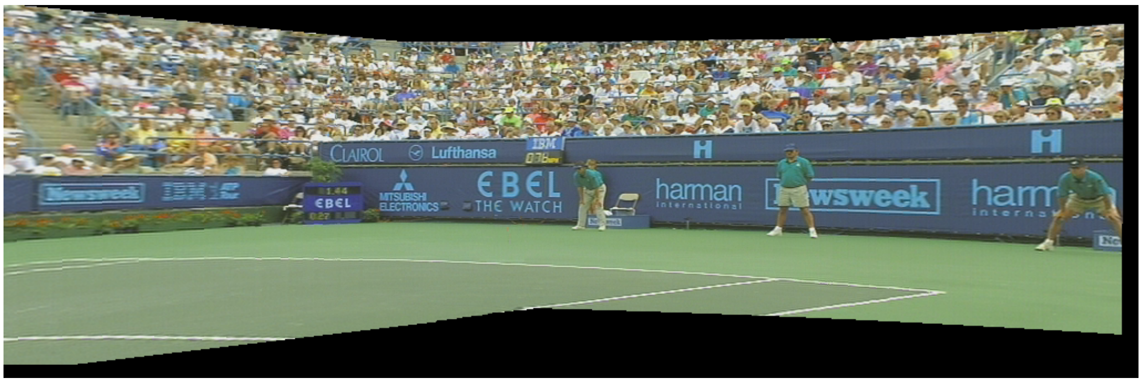
2.3.1.1 Single Sprites

A so-called single Sprite models the background of a given sequence in one single image. This image usually is of large dimensions and contains only the pixels from the background of the sequence. An example of a single Sprite is depicted in Fig. 2.22.

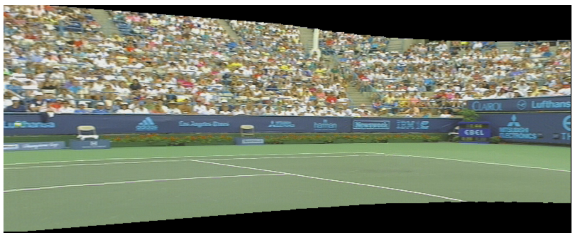
For the creation of a single Sprite, a reference frame is chosen and all other frames of the sequence are warped into the coordinate system of the reference. For that, so-called long-term higher-order motion parameters are computed that describe this transformation [70].

At first, short-term parameters are calculated using global motion estimation (GME) and a higher-order motion model. These short-term parameters are then accumulated by simple matrix multiplication as shown in Fig. 2.23.

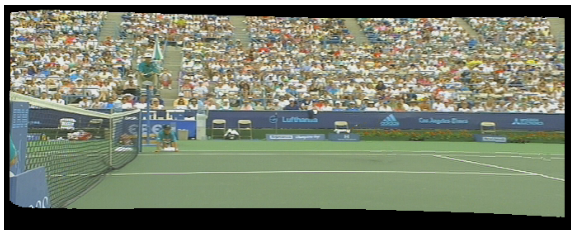
By transforming all frames using the long-term parameters into the coordinate system of the background Sprite, a stack of size $M \times N \times S$ is created where M and N are the



(a) Frames 0-244



(b) Frames 245-261



(c) Frames 262-297

Figure 2.24: Multiple Sprites, sequence “Stefan”

dimensions of the final background Sprite and S is the number of frames in the sequence. The images in this stack are then blended together to generate the background Sprite. The intention is to eliminate the foreground objects in the background Sprite. Blending filters normally used are the mean or the median of all pixel values lying in dimension S on top of each other.

2.3.1.2 Multiple Sprites

Multiple Sprites are used to improve the coding efficiency versus quality trade-off, especially for sequences with large camera pans. The development in this area can be simply described using one example. The well-known “Stefan” sequence has been used very often to evaluate Sprite techniques. For example, the multiple Sprite generation algorithm proposed in [41] leads to three partitions. The algorithms proposed in [11] and [94] generate four partitions by use of the same test sequence. The approach presented in [32] focuses more on the quality of the reconstructed frames from the Sprite and produces six partitions. The algorithms proposed in [41] and [32] are introduced next.

Concatenating the short-term perspective camera parameters (homographies $\mathbf{H}_{n-1,n}$) in a recursive way yields non-exact long-term parameters representing the transformation $\mathbf{H}_{0,n}$ between any frame and the reference frame. These homographies are the base for an robust but coarse camera calibration technique, published in [42]. Here we exploit the fact that for

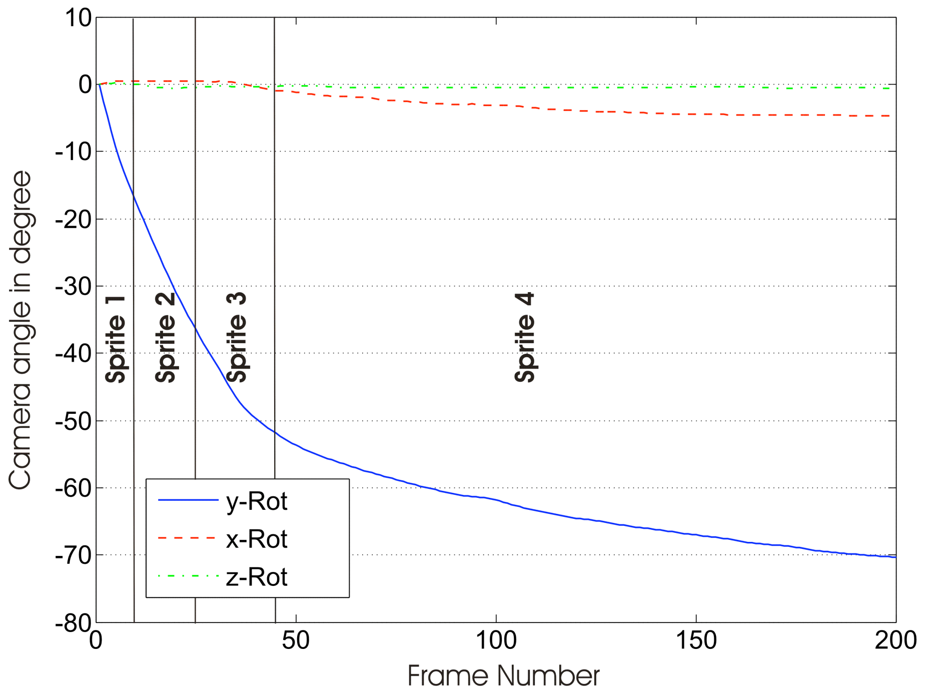


Figure 2.25: Rotation angles for test sequence "Biathlon" and sequence partitioning according to y-rotation for multiple Sprite construction

common camera setups the homographies can be decomposed in a product of intrinsic and extrinsic camera parameter matrices

$$\begin{aligned}\mathbf{H}_{0,n} &= \mathbf{F}_n \mathbf{R}_{0,n} \mathbf{F}_0^{-1} \\ &= \frac{1}{\alpha_{0,n}} \begin{pmatrix} r_{00} & r_{01} & f_0 r_{02} \\ r_{10} & r_{11} & f_0 r_{12} \\ r_{20}\alpha_{0,n}/f_0 & r_{21}\alpha_{0,n}/f_0 & r_{22}\alpha_{0,n} \end{pmatrix},\end{aligned} \tag{2.7}$$

where $\mathbf{R}_{0,n}$ is the rotation matrix between frame 0 and n and $\mathbf{F}_n$ and $\mathbf{F}_0$ contain focal length values of both views. After computing the focal length ratio $\alpha_{0,n} = f_0/f_n$ we calculate the focal length of the reference frame as median of all solutions resulting from Equ. 2.7. This is done by exploiting orthogonality and constant vector norm constraints for the matrices $\mathbf{H}_{0,n}$. Knowing all focal lengths, the rotation angles can finally be computed using trigonometrically properties of the center points of every image [42]. Figure 2.25 shows the rotation angles for sequence "Biathlon". The main motion is a left pan of the camera.

To split a video shot into several sequences, we first compute angle division for the rotation angle (φ_y or φ_x) with the maximum overall rotation $\Delta\varphi_{max}$. We minimize cost function C with respect to the number of angles M in order to find the number of Sprites to be generated for one rotation plane.

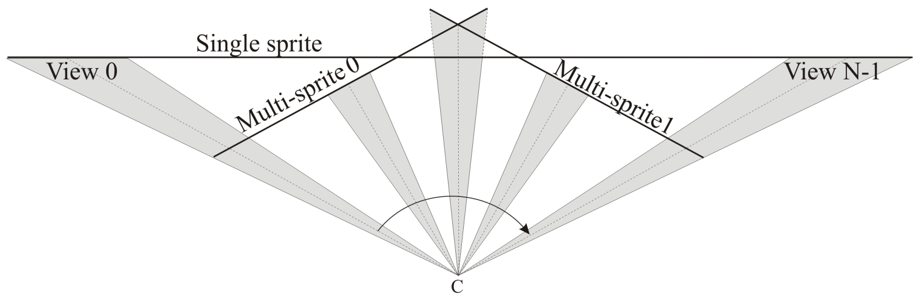


Figure 2.26: Partition of a sequence into multi-Sprites for panning camera with constant focal length

$$C\left(\Delta\varphi_{max}, M\right) = \sum_{i=0}^{M-1} f_{max,i} \cdot 2\tan\left(\frac{\Delta\varphi_{max}}{2} + \frac{FOV}{2}\right) \tag{2.8}$$

Thus, we optimize the Sprite memory cost with respect to the Sprite image size along one dimension. Figure 2.26 shows exemplarily the partition of a horizontal pan for the generation of two Sprites. Note that the horizontal latitudes of the multi-Sprites together is much smaller than the single Sprite latitude. The reference frame is chosen to be the middle frame of a sub-sequence with respect to the rotation angle. Sprites are finally constructed by applying direct frame-to-mosaic registration and advanced median blending to remove artifacts from the foreground objects.

The second multiple Sprite generation approach introduced here assumes that background Sprites are synthesized from image sequences with camera panning and tilting whereas sequences without these camera movements do not contribute to the generation of background Sprites. An object-based video encoder that uses multiple background Sprites can use metric-based measures such as the rotation angles for panning/tilting or alternatively the segmentation results of camera motion characterization. Since panning/tilting are the motion types of interest here, the characterization of camera work considers only panning left/right, tilting up/down, and no panning/tilting.

The camera motion characterization approach as shown in Fig. 2.27 has a feature extraction, classification, and a temporal segmentation stage. In the following, each of these stages is described briefly. A more detailed description of this approach can be found in [17].

The feature extraction uses the horizontal and vertical translational parameters $h_{x,l}$ and $h_{y,l}$ of the earlier estimated perspective global motion parameters contained in $\mathbf{H}$ as input to compute four features for pan and tilt, respectively. The index l addresses all motion parameters for frames l and $(l+1)$. The complex normalized value

$$\underline{t}_l = \frac{h_{x,l}}{w} + j\frac{h_{y,l}}{h} \tag{2.9}$$

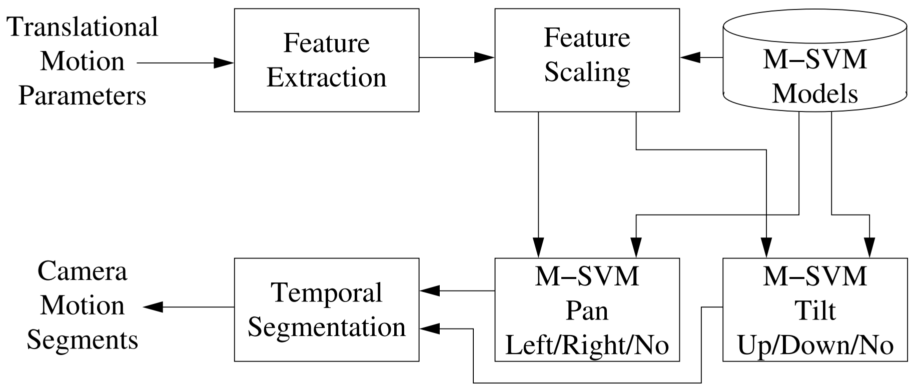


Figure 2.27: Camera motion characterization for pan left, pan right, no pan, tilt up, tilt down, and no tilt

is used to determine the median angle $\phi_{\mathrm{t,med},l}$ of translational motion and the median values $t_{\mathrm{x,med},l}$ and $t_{\mathrm{y,med},l}$ with

$$\phi_{\mathrm{t,med},l} = \underset{s_l \le k \le e_l}{\mathrm{median}} \left(\arg\left(\underline{t}_k\right)\right) \tag{2.10}$$

$$t_{\mathrm{x,med},l} = \underset{s_l \le k \le e_l}{\mathrm{median}} \left(h_{x,k}\right) \tag{2.11}$$

$$t_{\mathrm{y,med},l} = \underset{s_l \le k \le e_l}{\mathrm{median}} \left(h_{y,k}\right), \tag{2.12}$$

where w and h are the image width and height and s_l and e_l are given as

$$s_l = l - W_{\mathrm{med}} + 1 \tag{2.13}$$

$$e_l = l + W_{\mathrm{med}}. \tag{2.14}$$

The median filtered parameters are robust against possible GME outliers. The used windowed median filter has a length of $W_{\mathrm{med}} = \lfloor R_f/2 \rfloor$ with R_f as frame rate per second of the video sequence.

Short-time translational angle histograms based on $\phi_{\mathrm{t,med},l}$ are determined to obtain more robust features for the direction of translational motion. The used angle quantization scheme is shown in Fig. 2.28. The derived rates $R_{\mathrm{TAHPL},l}$, $R_{\mathrm{TAHPR},l}$, $R_{\mathrm{TAHTU},l}$, and $R_{\mathrm{TAHTD},l}$ represent the occurrence of angles for pan left/right and tilt up/down in the respective range of angles normalized to the window length W for the histogram computation. The used overlap of windows is extensive for a proper temporal resolution.

The zero-crossing rates for horizontal and vertical translational motion parameters are defined by

$$Z_{\mathrm{x},l} = \frac{1}{2W} \sum_{i=s_l}^{e_l} \left|\mathrm{sgn}\left(h_{x,i}\right) - \mathrm{sgn}\left(h_{x,i-1}\right)\right| \tag{2.15}$$

$$Z_{\mathrm{y},l} = \frac{1}{2W} \sum_{i=s_l}^{e_l} \left|\mathrm{sgn}\left(h_{y,i}\right) - \mathrm{sgn}\left(h_{y,i-1}\right)\right| \tag{2.16}$$

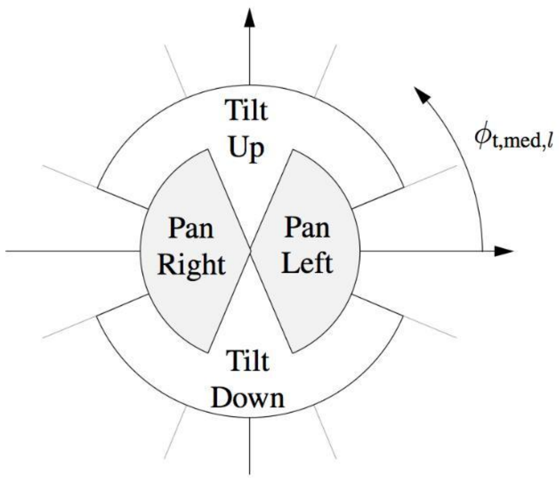


Figure 2.28: Quantization scheme for the translational motion angle histogram (TAH)

and capture the reliability of intended translational motion within the window of the length W.

The complete four-dimensional feature vectors for classification of pan and tilt are as follows

$$\mathbf{x}_{\mathrm{pan},l} = \begin{pmatrix} t_{\mathrm{x,med},l} & R_{\mathrm{TAHPL},l} & R_{\mathrm{TAHPR},l} & Z_{\mathrm{x},l} \end{pmatrix}^{\mathrm{T}} \tag{2.17}$$

$$\mathbf{x}_{\mathrm{tilt},l} = \begin{pmatrix} t_{\mathrm{y,med},l} & R_{\mathrm{TAHTU},l} & R_{\mathrm{TAHTD},l} & Z_{\mathrm{y},l} \end{pmatrix}^{\mathrm{T}}. \tag{2.18}$$

Multi-class support vector machines (M-SVMs) are used to classify the camera motion types. The M-SVMs provide for each image pair a result with the three possible states pan left/right, and no pan as well as tilt up/down, and no tilt. The models for the M-SVMs were trained on features extracted from selected videos of the TRECVid 2005 BBC rushes video corpus [17]. The temporal segmentation starts with a median filtering of results over 15 frames. This improves the temporal stability. Changes between camera motion types are then identified within an image sequence as boundaries of segments with the same type of camera motion. This leads to a camera motion-based temporal segmentation.

The characterization of the camera motion separates the video sequence into segments depending on the camera motion. Afterwards, for each segment, the physical camera parameters are estimated and based on that it is decided whether a single or a multiple sprite is generated for the current segment. Figure 2.29 illustrates this method.

Exemplary, the new method is applied on the well-known "Stefan" - sequence. Six background Sprites are generated. Firstly, the camera motion characterization separates the video sequences in four parts. Then, for each part, physical camera parameters are calculated. These parameters are used to segment the fourth sub-sequence into three Sprites again. This leads to the six part Sprites as shown in Fig. 2.30.

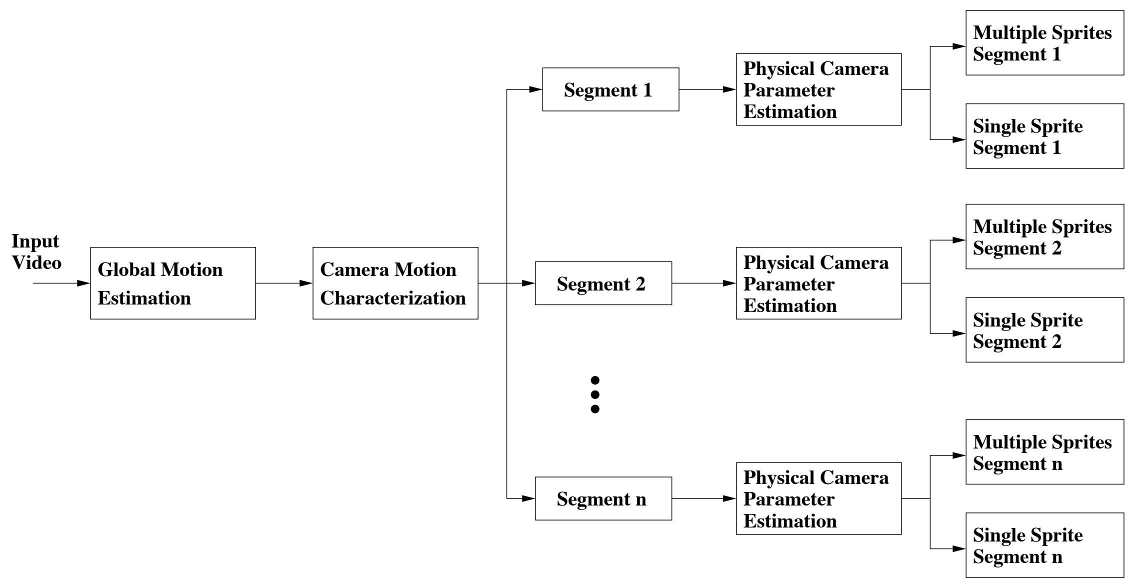

Input Video
Global Motion Estimation
Camera Motion Characterization
Segment 1
Segment 2
Segment n
Physical Camera Parameter Estimation
Physical Camera Parameter Estimation
Physical Camera Parameter Estimation
Multiple Sprites Segment 1
Single Sprite Segment 1
Multiple Sprites Segment 2
Single Sprite Segment 2
Multiple Sprites Segment n
Single Sprite Segment n

Figure 2.29: Multiple Sprite Generation using CMC

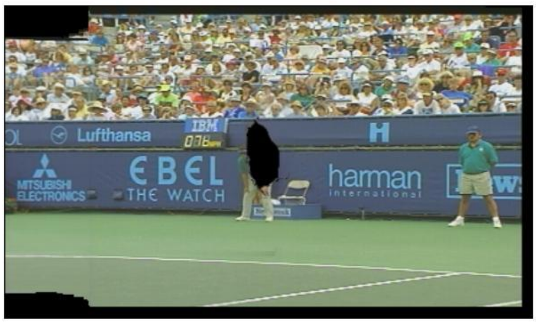
2.3.1.3 Super-resolution Sprites

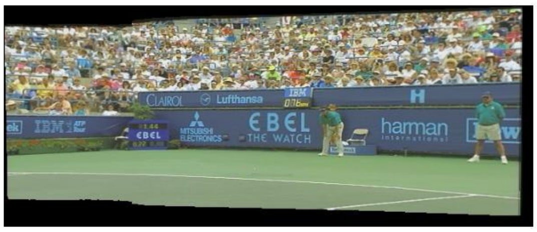
Super-resolution is a technique that aims on increasing the quality of an image. A high-resolution counterpart is built from several images with lower resolution. These images can originate from one camera taking multiple images of a scene in time, from multiple cameras each taking one image in time or from the frames of a moving video camera. The idea of this approach is that an arbitrary point is visible several times.

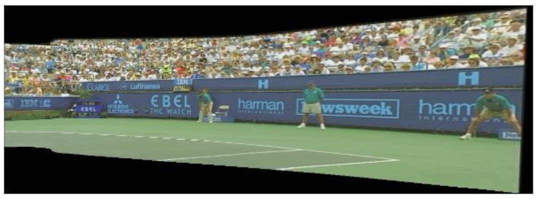
This method can easily be extended to background Sprite generation. When building a background Sprite a video sequence is used. After global motion estimation and transformation into the coordinate system of the reference frame the pixel locations are rarely integer values. This feature can be used to generate a single or multiple background Sprites of higher resolution [40], [43], [94]. It has been shown in the literature that these super-resolution techniques can improve the quality of the background Sprite. However, this type of Sprite generation is beyond the scope of this work.

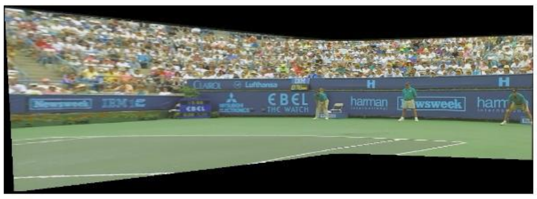
2.3.2 Local Background Sprites

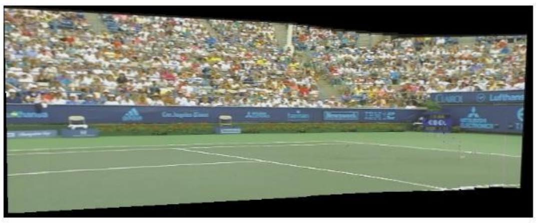
Now we come to our new approach. The tendency generating three, four and six partitions of the "Stefan"-example has led to the idea to build a local Sprite for each frame of the input sequence. The term local background Sprite specifies a model of the background. Other than general background Sprites one model is built for every frame and not one model for the whole video sequence. Only the local temporal neighborhood of each reference frame is taken into account for Sprite generation. The dimensions of a local background Sprite match those of the reference frame. The idea is to minimize distortion in background regions.


(a) Multi sprite 1, 1 – 42 frames


(b) Multi sprite 2, 43 – 106 frames


(c) Multi sprite 3, 107 – 201 frames


(d) Multi sprite 4, 202 – 244 frames


(e) Multi sprite 5, 245 – 261 frames

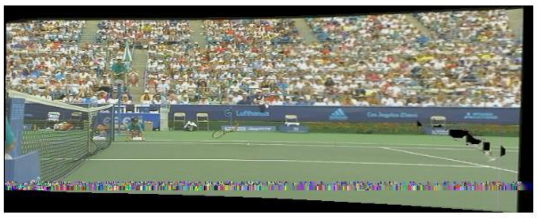

(f) Multi sprite 6, 262 – 297 frames

Figure 2.30: Multiple Background Sprites over 297 frames, test sequence "Stefan"

When a background frame is reconstructed from a general background Sprite, distortion can be severe. This is due to accumulated errors in the global motion estimation, non-ideal interpolation and the double mapping into the coordinate system of the background Sprite and back.

The algorithm for modeling local background Sprites for a given video sequence is depicted in Fig. 2.31. Its different parts are explained in this section.

2.3.3 Global Motion Estimation

For global motion estimation, a hierarchical gradient descent approach based on the Gauss-Newton method is used as presented in the previous section. A block chart of the approach can be seen in Fig. 2.21. The algorithm estimates the displacement between two temporally adjacent frames I_p and I_q of a sequence using the 8-parametric higher-order perspective motion model which is described by the following equations

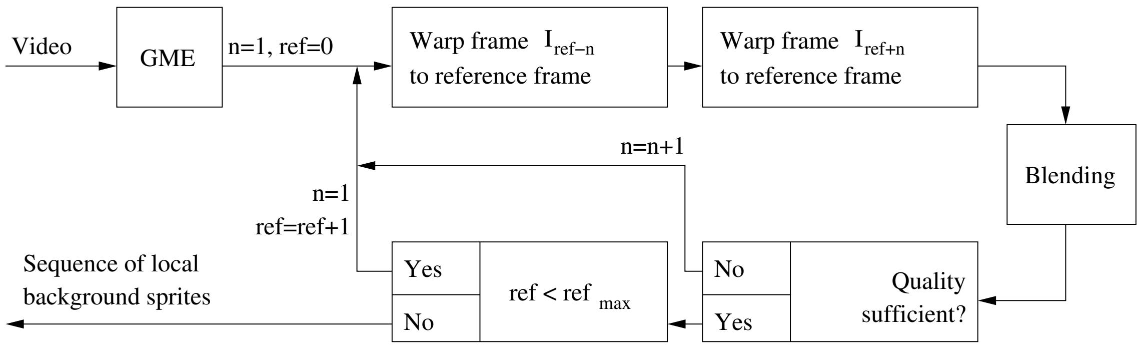

Video
GME
n=1, ref=0
Warp frame I_ref−n to reference frame
Warp frame I_ref+n to reference frame
Blending
n=n+1
n=1
ref=ref+1
Sequence of local background sprites
Yes
No
ref < ref max
No
Yes
Quality sufficient?

Figure 2.31: Modeling the background by means of local background Sprites

$$x_q = \frac{m_0 x_p + m_1 y_p + m_2}{m_6 x_p + m_7 y_p + 1} \tag{2.19}$$

$$y_q = \frac{m_3 x_p + m_4 y_p + m_5}{m_6 x_p + m_7 y_p + 1} \tag{2.20}$$

where $(x_p\ y_p)^{\mathrm{T}}$ is the location of a pixel in frame I_p and $(x_q\ y_q)^{\mathrm{T}}$ its corresponding position in frame I_q. The parameters m_0 to m_7 describe the motion by means of translation, scaling, rotation, sheering and perspective transformation.

Since the short-term displacement between two frames I_p and I_q is going to be used several times while creating all local background Sprites for a video sequence, the motion parameters are computed in a pre-processing step. This means for a sequence with n frames the set T of transformation matrices

$$T = \{\mathbf{W}_{0,1}, \mathbf{W}_{1,2}, \ldots, \mathbf{W}_{n-2,n-1}\} \tag{2.21}$$

and its inverted correspondences

$$T_{inv} = \{\mathbf{W}_{0,1}^{-1}, \mathbf{W}_{1,2}^{-1}, \ldots, \mathbf{W}_{n-2,n-1}^{-1}\} \tag{2.22}$$

are computed where $|T| = |T_{inv}| = n - 1$, $\mathbf{W}_{p,q}^{-1} = \mathbf{W}_{q,p}$ and

$$\mathbf{W}_{p,q} = \begin{bmatrix} m_{0,p,q} & m_{1,p,q} & m_{2,p,q} \\ m_{3,p,q} & m_{4,p,q} & m_{5,p,q} \\ m_{6,p,q} & m_{7,p,q} & 1 \end{bmatrix} \tag{2.23}$$

is the transformation matrix between frames I_p and I_q.

2.3.4 Warping and Blending

For every reference frame a local background Sprite is to be built. Therefore, the algorithm iteratively transforms temporally neighboring frames into the coordinate system of the reference. This produces a dynamically growing image stack of size $M \times N \times S_t$ where M and

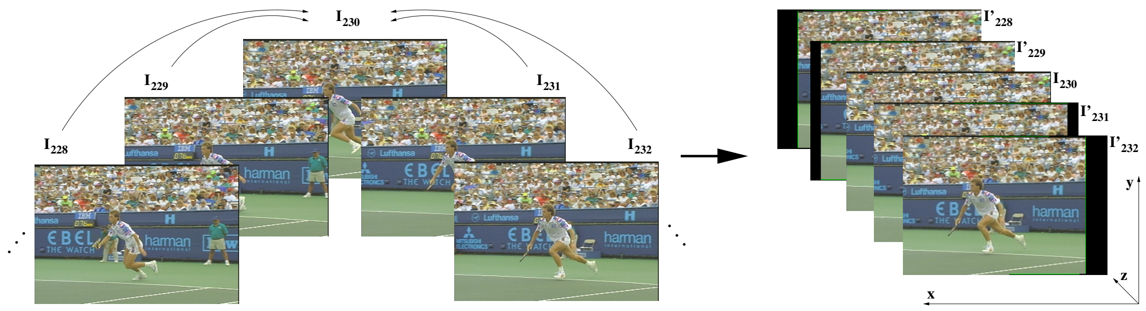


Figure 2.32: Creation of an image stack for the generation of a local background Sprite, sequence “Stefan”, reference frame 230

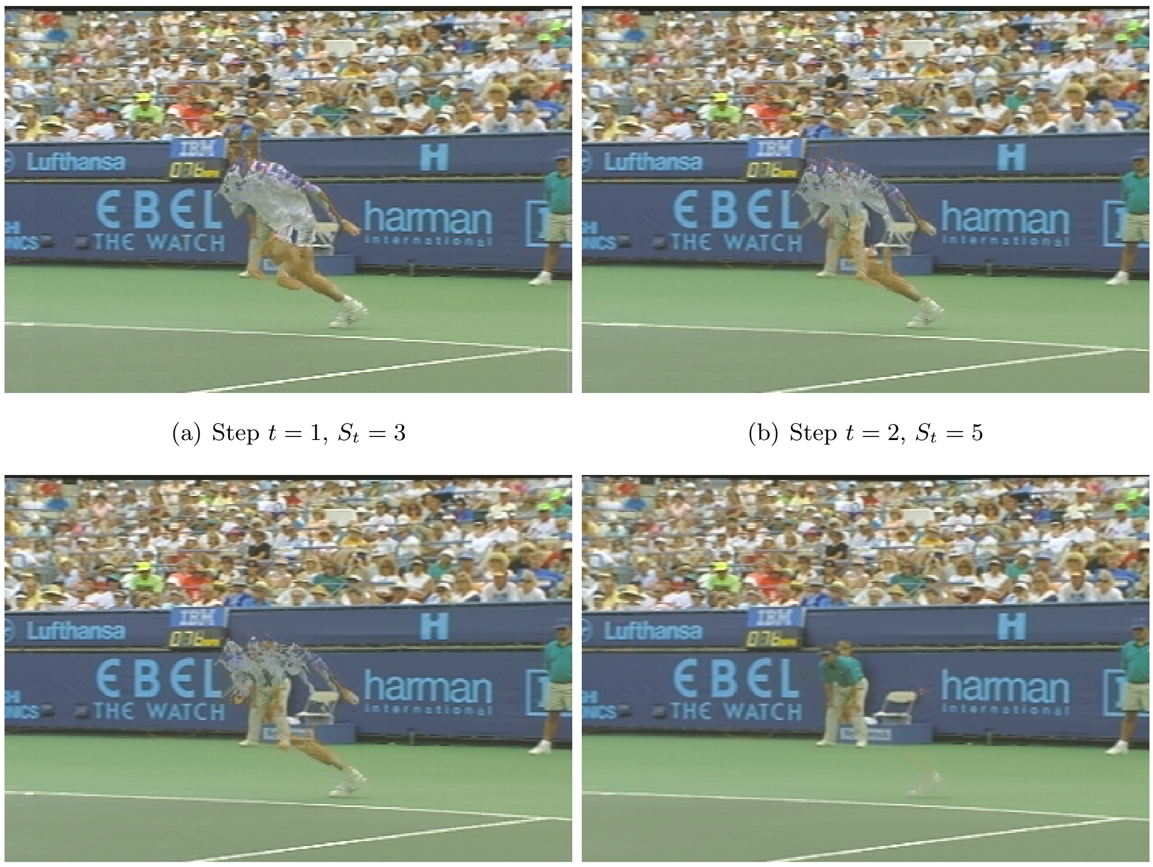
N are the dimensions of the reference frame and $S_t = 2t+1$ is the depth of the stack in step t. In step $t = 0$ the stack only contains the reference frame. This approach can be seen in Fig. 2.32.

For the transformation of an arbitrary frame into the reference’s coordinate system the short-term motion parameters from the preprocessing step are accumulated to generate long-term parameters which can be seen in Fig. 2.23. The global motion estimation can only compute the displacement between two frames by approximation. Due to existing small errors and their accumulation, the error in the long-term parameters grows larger with increasing temporal distance to the reference frame. Hence, the long-term parameters are used as initialization for another gradient descent step to reduce this error.

In every step t the images in the stack are merged together to build a preliminary local background Sprite of size $M \times N$. For this purpose a so-called blending filter is used, which here is a median filter. The median returns the middle value of an ordered dataset – in this case a set of luminance and chrominance values respectively. The advantage over using a mean filter is its robustness for outliers. Additionally, the median is always an element of the set itself and does not produce new values.

By successively adding temporally neighboring frames the foreground objects in the preliminary local background Sprites are removed step by step. This is due to the area behind the foreground objects that is disclosed because of their movements. This can be seen in Fig. 2.33. The preliminary local background Sprites are depicted for the “Stefan” sequence for various steps t. One can clearly see that the foreground object has nearly completely vanished after eight blending steps.

It is possible to evaluate the quality of the background model in every step subjectively. However, an automatic evaluation criterion is desirable that stops the generation of the local background Sprite when its quality is sufficient. An approach for automatic quality evaluation is presented next.


(a) Step $t = 1$, $S_t = 3$ (b) Step $t = 2$, $S_t = 5$

(c) Step $t = 3$, $S_t = 7$ (d) Step $t = 8$, $S_t = 17$

Figure 2.33: Preliminary local background Sprites, sequence “Stefan”, reference frame 230

2.3.5 Quality Evaluation of Local Background Sprites

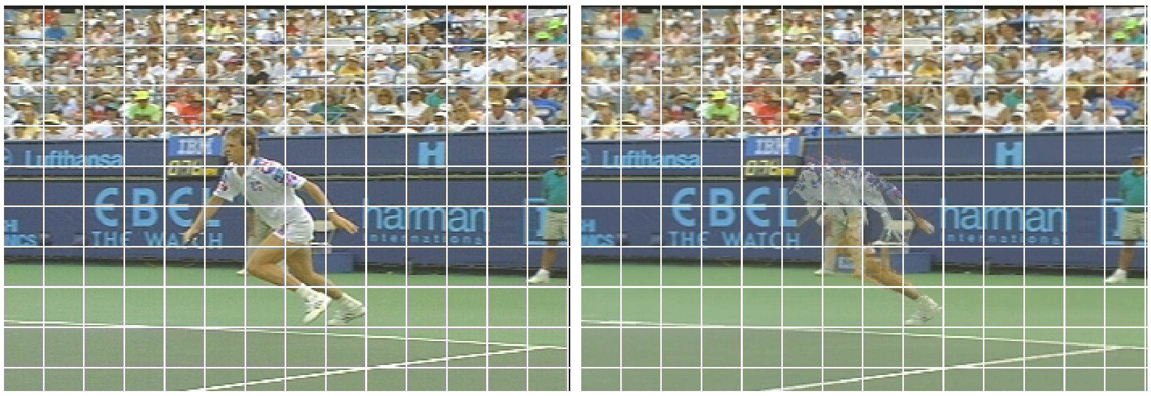
A possible measure for the difference between two images or frames is the root mean square error (RMSE). The RMSE between a reference frame $I_{ref}(x, y)$ and its preliminary local background Sprite $I_{bs,t}(x, y)$ in step t is defined by

$$RMSE_t = \sqrt{\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I_{ref}(i,j), I_{bs,t}(i,j))^2} \tag{2.24}$$

where M and N are the dimensions of the reference frame and the preliminary local background Sprite respectively. The preliminary local background Sprite in step $t = 0$ is the reference frame itself so that $I_{bs,t=0} = I_{ref}$ and $RMSE_{t=0} = 0$. Since the foreground objects vanish step by step the RMSE value increases successively. Therefore, the difference of the RMSE values in two consecutive steps

$$\Delta RMSE_t = RMSE_t - RMSE_{t-1} \tag{2.25}$$

decreases. When the foreground objects are completely eliminated, the values $RMSE_t$ and $\Delta RMSE_t$ change only marginally.



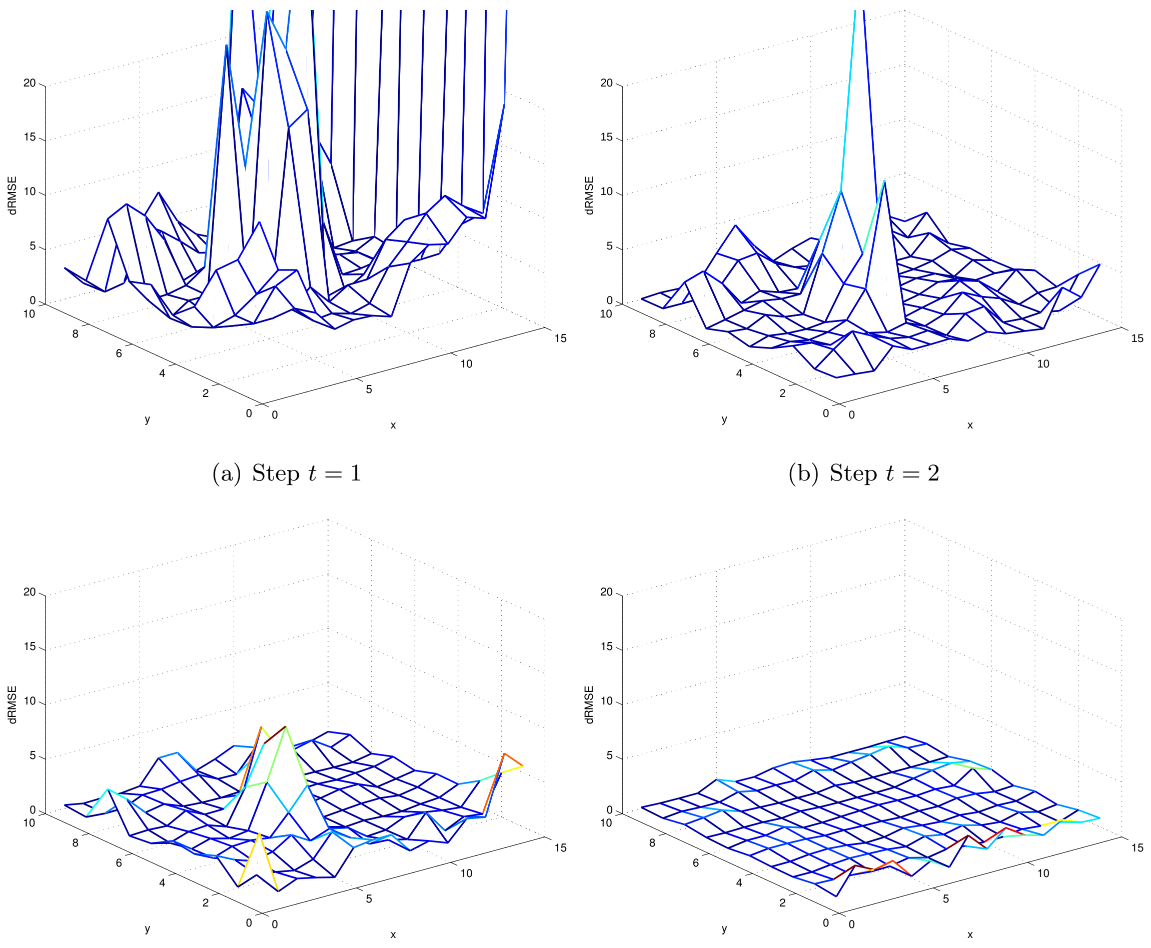
(a) Reference frame (b) Preliminary local background Sprite, $t = 2$

Figure 2.34: Partitioning of images into blocks of size 25×25, sequence “Stefan”, reference frame 230

At the beginning, foreground objects are still present in the preliminary local background Sprite. Change in these areas leads to high values of $\Delta RMSE_t$. After several steps, most of the foreground is eliminated which leads to lower values of $\Delta RMSE_t$. It holds true that $\Delta RMSE_a >= \Delta RMSE_b$ for $a <= b$. Therefore, the value $\Delta RMSE_t$ can be interpreted as a measure for the information about the background that has been added to the local background Sprite in step t.

Using the measure $\Delta RMSE_t$ is not without problems when only small foreground objects are present. Small objects only take a minor percentage of the whole frame. The influence of errors in these areas on the measure is small compared to the sum of errors in the background regions. In this case plotting $RMSE_t$ and $\Delta RMSE_t$ against time produces very flat curves which make a decision on the quality of the preliminary local background Sprite very difficult. Therefore, we define matrices containing the blockwise calculated value $\Delta RMSE_t$, so-called dRMSE-matrices. Reference frame and preliminary local background Sprite are divided into blocks of fixed size, which can be seen in Fig. 2.34. Within this work, various block sizes between 10×10 and 40×40 have been tested. The problem with small block sizes is that the profile of the dRMSE-matrices is very high-frequency. With large block sizes the unwanted effect of averaging of large regions sets in again. In this work a fixed block size of 25×25 is used. The value $RMSE_t$ is then calculated for every block independently. Thus, no averaging over the whole frame takes place. Distinct areas in the preliminary local background Sprite can be evaluated independently. Furthermore, the difference to the block values in the step before is computed using Equation 2.25.

Figure 2.35 shows the corresponding blockmatrices for the example in Fig. 2.33. At the beginning the plot of the matrix is very wavy. With increasing steps t the matrix flattens successively. This means, the more temporally neighboring frames are transformed into the local background Sprite’s coordinate system the less information about the background of the reference frame is gained. The peak in the middle of the matrices in Fig. 2.35 results



(a) Step $t = 1$

(b) Step $t = 2$

(c) Step $t = 3$

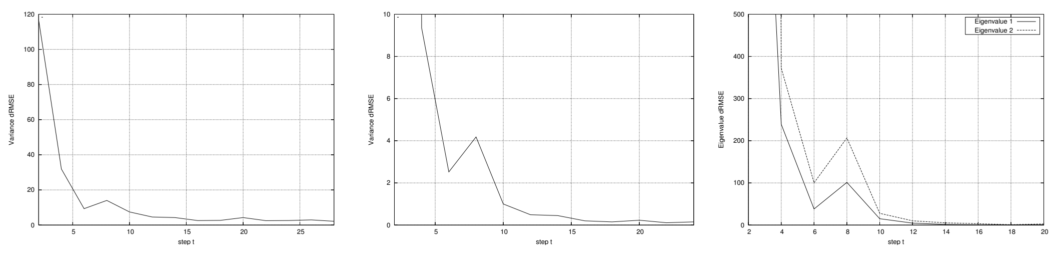
(d) Step $t = 8$

Figure 2.35: dRMSE-matrices using blocks of size 25×25, sequence “Stefan”, reference frame 230

from the moving tennis player and the area behind him that is disclosed. However, the RMSE in the background regions doesn’t nearly change at all. After 8 blending steps (Fig. 2.35 (d)) – 16 frames and the reference frame have been blended together – the matrix is nearly flat in all regions. This corresponds with the results in Fig. 2.33.

The quality of the preliminary local background Sprites now is assessable in a very differentiated way. Assuming the generation of the local background Sprite is to be aborted when there is no more information added in any region, meaning the matrix presented is flat in every region. We present three possible evaluation criteria for dRMSE-matrices.

The easiest way to evaluate the matrices is their maximum value. The maximum provides information about the biggest change of a block in the preliminary local background Sprite. The curve of the maximum plotted against step t can be seen in Fig. 2.36(a). Another way to evaluate the matrices is the variance of their values. The variance provides information about the distribution of values in a dRMSE-matrix. The curve of the variance plotted against step t can be seen in Fig. 2.36(b). The last way to evaluate the matrices is inspired



(a) Maximum (b) Variance (c) Eigenvalues

Figure 2.36: Various evaluation critera for dRMSE-matrices, sequence “Stefan”, reference frame 230

Evaluation criterion	**Threshold**
Maximum	5.0
Variance	0.2
Eigenvalues	5.0

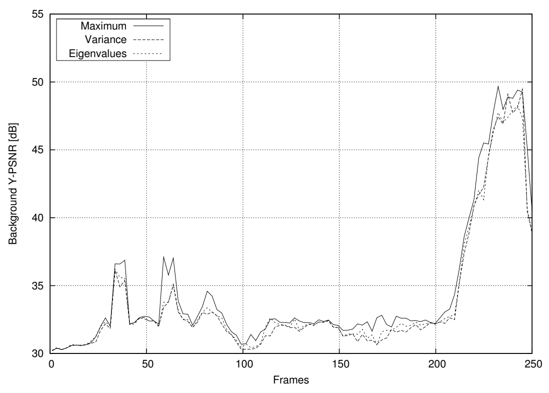
Table 2.6: Thresholds for evaluation criteria

by the window-based feature tracker of Kanade et al. [2]. As already described, in their work, they define that a good feature is one that can be tracked well. This is the case when its texturizing is high. Two large eigenvalues represent any highly textured pattern that can be tracked well. The dRMSE-matrices can be compared to the intensity profile of a feature window. On the contrary, we are looking for a nearly constant profile which means two small eigenvalues. Therefore, in every step t matrix $\mathbf{Z}$ is computed from the gradientes of the dRMSE-matrices. The curves of the eigenvalues plotted against step t can be seen in Fig. 2.36(c).

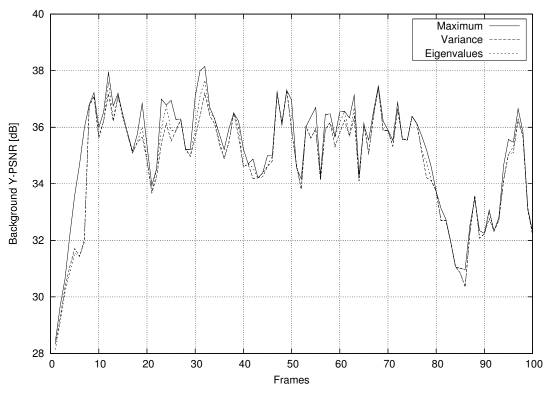
For every evaluation criterion, we assume local background Sprite quality sufficient when maximum, variance and eigenvalues respectively lie below their predefined threshold values. The used thresholds can be see in Table 2.6.

2.3.6 Experimental results

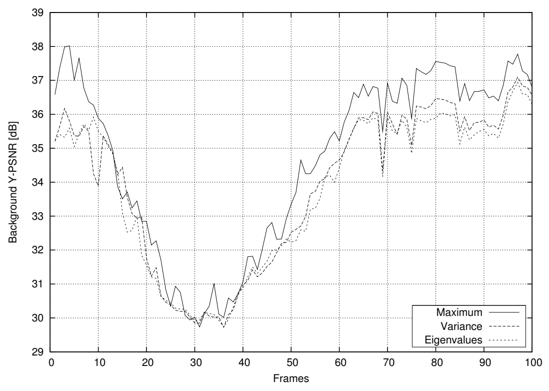
We have evaluated our approach using four test sequences. The first sequence is called “Allstars (Shot 1)” (352×288, 250 frames) and is recorded from a soccer broadcast in german television. It mainly consists of translational motion but is somewhat difficult to handle for global motion estimation because of its low-frequency content (green pitch). The second sequence is called “Mountain” (352×192, 100 frames) and is part of a BBC documentary. It shows a leopard chasing an animal while moving down a hill. This sequence consists also mainly of translational motion but has a high-frequency background (mountains). The third sequence is called “Race1 (View 0)” (544×336, 100 frames) and is part of an MPEG testset



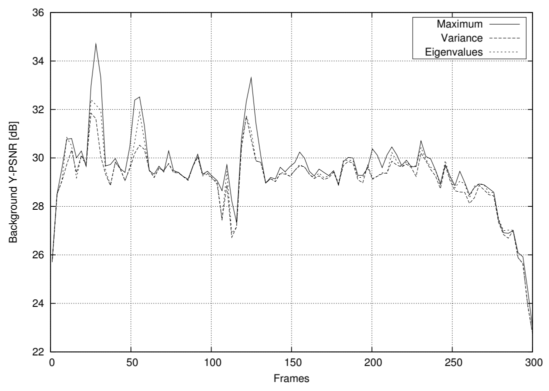
(a) Sequence “Allstars (Shot 1)”



(b) Sequence “Mountain”



(c) Sequence “Race1 (View 0)”



(d) Sequence “Stefan”

Figure 2.37: Background PSNR of local background Sprites using various evaluation criteria for dRMSE-matrices

for multiview sequences. It shows a kart race. The forth sequence is the already introduced “Stefan” sequence (352×240, 300 frames).

For evaluation of background quality we compute the background PSNR of the model using equation

$$PSNR = 10 \cdot \log_{10}\left(\frac{255^2}{MSE}\right) \tag{2.26}$$

for the sequence’s luminance part. We have generated ground truth masks of the foreground objects for all sequences to compute the PSNR only between the background pixels for the original reference frame and the background model.

First, we compare all possible evaluation criteria, i.e. maximum, variance and eigenvalues, for the generation of local background Sprites. Figures 2.37(a) to 2.37(d) show the background PSNR for the background models generated. Table 2.7 shows the mean PSNR values. It can be seen that for every sequence the maximum criterion produces the best background quality (bold values in Table 2.7). This is due to the fact, that local background Sprite generation stops using the least temporally neighboring frames when the maximum criterion is chosen. The high values in the last quarter of sequence “Allstars (Shot 1)”

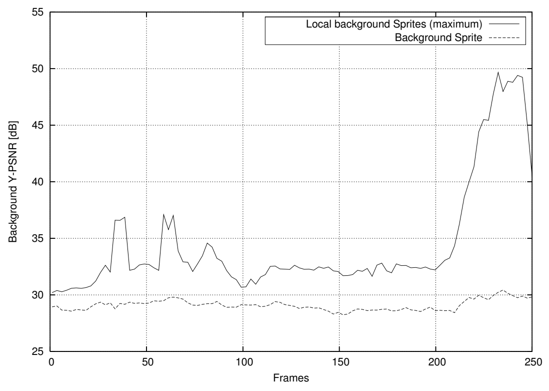
Sequence	**Algorithm**	**PSNR [dB]**
Allstars (Shot 1)	Local background Sprite (maximum)	**34.4345**
	Local background Sprite (variance)	33.7043
	Local background Sprite (eigenvalues)	33.8240
Mountain	Local background Sprite (maximum)	**35.1592**
	Local background Sprite (variance)	34.7299
	Local background Sprite (eigenvalues)	34.8040
Race1 (View 0)	Local background Sprite (maximum)	**34.6666**
	Local background Sprite (variance)	33.9092
	Local background Sprite (eigenvalues)	33.7659
Stefan	Local background Sprite (maximum)	**29.5146**
	Local background Sprite (variance)	29.0908
	Local background Sprite (eigenvalues)	29.2079

Table 2.7: Mean background PSNR values for local background Sprites generated using all evaluation criteria

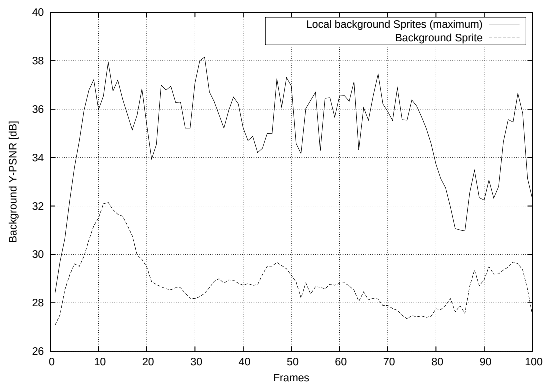
(Fig. 2.37(a)) come from the camera that is static in that part of the sequence. Therefore, the translational initialization of the global motion estimation algorithm is very exact and the gradient descent algorithm produces very small errors. The PSNR profile of sequence "Mountain" (Fig. 2.37(b)) is almost constant. Except for low values at the beginning and at the end of the sequence. These are explainable with errors in registration. The low values around frame 30 of sequence "Race1 (View 0)" (Fig. 2.37(c)) come from a fast camera pan that produces big translational motion. Contrary to the last part of sequence "Allstars (Shot 1)" the translational initialization in that part of the sequence produces higher errors for the global motion estimation. The peaks around frames 25, 50 and 120 and the drop in the last 30 frames of sequence "Stefan" (Fig. 2.37(d)) are explainable alike. The camera motion is static or nearly static in parts with peaks and moves very fast at the end of the sequence.

Next, the local background Sprites are compared to common background models. We use on one hand the local background Sprites provided by the worst criterion – which in all cases is variance, except for sequence "Race1 (View 0)" which is eigenvalues. On the other hand we use reconstructed background sequences from single, multiple and super-resolution Sprites, repspectively. The plots for the comparison are shown in Fig. 2.38(a) to 2.38(d). Table 2.8 shows the mean PSNR values. It can be seen that the new approach outperforms common background models even when the worst evaluation criterion is used at about $2-6$ dB (bold values in Table 2.8).

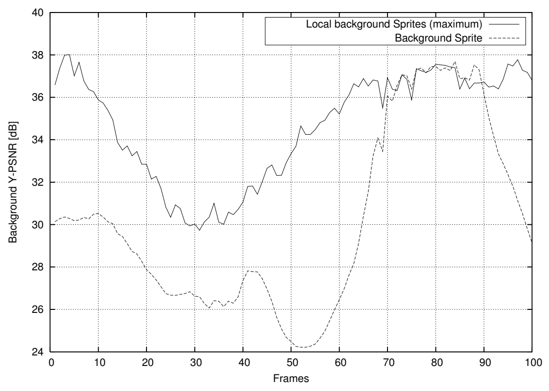
For sequence "Stefan" (Fig. 2.38(d)) only the first 250 frames are taken into account for the mean PSNR value as the super-resolution Sprite existed only for that range.



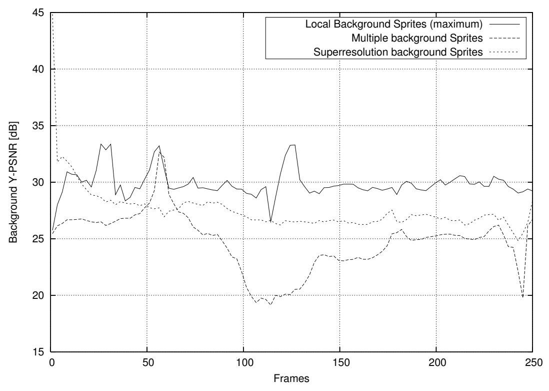
(a) Sequence "Allstars (Shot 1)"



(b) Sequence "Mountain"



(c) Sequence "Race1 (View 0)"



(d) Sequence "Stefan"

Figure 2.38: Comparing background PSNR of reconstructed single/multiple/super-resolution Sprites with local background Sprite (worst evaluation criterion)

2.4 Chapter summary and outlook to open issues

An enhanced global motion estimation algorithm based on gradient descent algorithm has been developed. Several features have been examined, such as a windowing technique to prevent the influence of possible foreground objects and an up-sampling step to prevent errors during the warping process. Using this proposed windowing technique, it is possible to estimate the motion of one object of a video sequence exactly, here the background, without any a-priori knowledge. The technique works very well even with large background occlusions due to large or multiple foreground objects, as has been demonstrated using a number of video test sequences of varying complexity. Test sequences with small foreground objects as well as with larger occluded background are considered. The use of optimal window sizes has been also determined. It has been shown that using this, the overall estimation process can be increased.

The up-sampling in the final registration step has also been investigated. Here, one test sequence without any foreground object was chosen to emphasize the effect using an up-sampling step for preventing aliasing during the image warping. It has been shown that

Sequence	**Algorithm**	**PSNR [dB]**
Allstars (Shot 1)	Single Sprite	29.0938
	Local background Sprite (variance)	**33.7043**
Mountain	Single Sprite	28.8877
	Local background Sprite (variance)	**34.7299**
Race1 (View 0)	Single Sprite	30.1350
	Local background Sprite (eigenvalues)	**33.7659**
Stefan (1-250)	Multiple Sprites	24.7312
	Super-resolution Sprite	27.4404
	Local background Sprite (variance)	**29.4479**

Table 2.8: Mean background PSNR comparing reconstructed single/multiple/super-resolution Sprites with local background Sprites (worst evaluation criterion)

Sequences	“Allstars”	“Biathlon”	“Foreman”	“Stefan”
GME/KLT Bilinear/Up	33.39	32.76	34.90	29.11
GME/KLT Spline	41.65	37.69	37.32	30.58

Table 2.9: Mean Background-PSNR values of short-term frame-by-frame estimation

even for the test sequence without foreground objects and more high-frequency texture, we achieve the highest estimation performance using only the up-sampling step. However, for all other test sequences we also increase the estimation performance except for one test sequence with low-frequency texture in the background.

The biggest challenge for further work is the enhancement of the in-built algorithm that chooses the best window size within one estimation process. The main application for our window-based global motion estimation algorithm is motion-based object segmentation. For that, a robust background motion estimation is strongly needed. Furthermore, the up-sampling issue can also be investigated especially concerning the content of the input video. Finally, initialization techniques (KLT, phase correlation) have been compared experimentally and the final enhanced GME-algorithm (see Fig. 2.21) has been designed. Furthermore, we have found out that using the spline interpolation and upsampling instead of upsampling and bilinear interpolation during the final warping process brings a huge improvement. Table 2.9 shows a first comparison.

The use of the spline interpolation method highly improves the global motion estimation results. Therefore, we use this kind of interpolation during the warping process instead of bilinear interpolation. However, we have seen during the experiments that the spline interpolation is not always the best choice. The main issue for further work here is a comprehensive research finding the optimal interpolation for the warping process.

We have presented a novel approach for background modeling of video sequences. Conventional background Sprites model the background of the sequence in one image that usually is of large dimensions. Other than that local background Sprites are generated for each reference frame individually and are of the same size as the reference. When using conventional background Sprites for applications like image segmentation the frames containing the background information have to be reconstructed from the Sprite image. This step is not necessary when local background Sprites are used. Therefore, distortion is reduced. It has been shown that the quality of the background model using this new technique clearly outperforms conventional background models.

Nevertheless, further work has to be done concerning all fixed thresholds used in this work. The block sizes of the dRMSE-matrices can be adaptively adjusted using a preliminary segmentation that provides information about the mean object size in the reference frame. The threshold values for the evaluation criteria can also be adaptively adjusted. The curves are similar to an exponential path. Curve fitting techniques can estimate the path using past values and adjust the threshold depending on a possible convergence.

Chapter 3

Video Object Segmentation in Sequences with a Moving Camera

Having the two basic techniques, i.e. an enhanced global motion estimation algorithm and local background Sprite generation, we can use these techniques to design an improved object segmentation algorithm. For our purpose, it has to be possible to perform foreground/background segmentation automatically, especially for video sequences with a moving camera. In this chapter, we describe new approaches with use of the pre-processing methods described in Chapter 2.

3.1 Introduction

Object segmentation is a key technique in applications such as object-based video coding and video surveillance. There are many approaches to segment or track foreground objects in scenes recorded by a still camera, such as in surveillance applications. In those scenarios, the background can be modeled and thus, a background subtraction method can be applied. However, the task of finding a pre-processing method for separating the foreground and background in scenes with moving camera, e.g. pan, tilt, roll is more difficult. The use of global motion estimation and Sprite generation techniques is a good base to tackle this problem. If a specific camera motion model, e.g. the perspective model, fits to the real camera motion, it is possible to generate a Sprite from this sequence. As already mentioned, a Sprite is a storage where all aligned frames are summarized in one image [70]. All considered frames are blended into one image controlled by higher-order motion parameters. For the blending process, a filter removing all foreground objects is used. As a result, the Sprite contains the background of a number of frames from this video sequence. The next step is to reconstruct the frames from the Sprite. This generates a video sequence only containing the background. If these frames are used for subtraction with the original frames, ideally, only the foreground object pixels appear in the resulting error frame [12]. During the Sprite generation step it can occur that the blending filter produces several deviations in

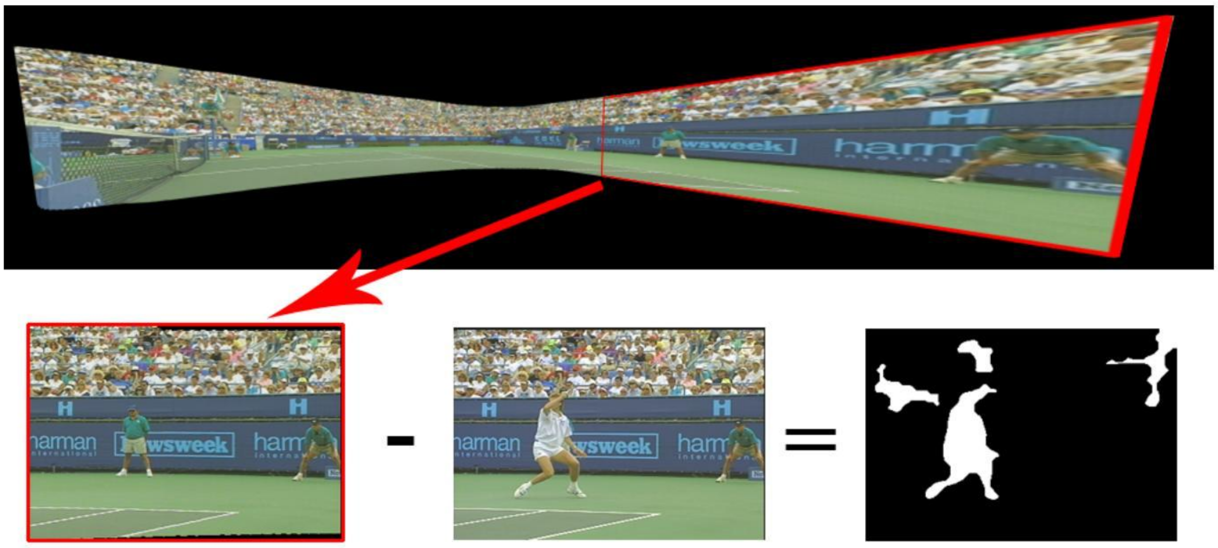


Figure 3.1: Problems using common background Sprites for object segmentation

the reconstructed frames compared to the original frames, possibly without any impact on the subjective quality. However, if the reconstructed background frames are to be used for segmentation, these different pixel values appear in the error frame and influence the segmentation algorithm. Furthermore, the reconstructed frames can be distorted by the Sprite generation process or by a non-static background. We address this problem by applying a second error frame pre-computed by a frame-by-frame short-term global motion estimation algorithm. To achieve this, it is very important to use an accurate method for the precise estimation of the background. Thus, the segmentation algorithm is applied on both error frames.

The segmentation is conducted in several steps. Before the error frame is converted to a binary frame, it is low-pass filtered using anisotropic diffusion. This technique was first proposed in [53]. The advantage of this method is that small edges between adjacent pixel values are blurred whereas large differences of adjacent pixel values remain. The binary image is produced thresholding the filtered error image. In our approach, the threshold is calculated using the average and the maximum of the pixel values. In order to remove small objects and to fill holes in the main objects, several well-know morphological operations are applied on the binary image. The binary masks obtained are combined together.

However, the mapping of pixel content from various frames in a scene into a single Sprite or a collection of multiple Sprites may cause severe geometrical distortion of the background especially in border regions. For reconstruction of the background of a single frame, a second mapping needs to be performed which causes additional distortion which is due to non-ideal interpolation or registration errors. Object segmentation using background subtraction is one application that is degraded by such distiortion. These problems are exemplified in Fig. 3.1. In a second proposed algorithm, local background Sprites are used. A mapping of content from many frames of a scene is performed for each individual frame. In other words, global motion estimation is performed from many adjacent frames into the frame where the background needs to be reconstructed. No backward mapping is required. Thus, our background Sprites are local and there are as many individual Sprites generated as frames exist in a sequence. This will result in a more precise background modeling and therefore background subtraction methods will be improved.

Further, the basic segmentation algorithm, i.e. processing the error frame from background subtraction to a binary mask, is also examined. The proposed scheme is compared with state-of-the-art thresholding algorithms. The use of color spaces is also taken into account for the whole segmentation process. Aspects described in this chapter have been discussed in [16].

3.2 Processing the error frame resulting from a background subtraction method

In Chapter 2, local background Sprites were introduced. We now would like to use this technique for generating background models of each frame of a video sequence to apply background subtraction methods for object segmentation. First, a background model is generated for each frame and second the original frames are subtracted with its corresponding background models. For each frame, which is subtracted with its background model, an error frame results, which is used to calculate the binary object mask. For that, a segmentation method is explained next based on Krutz et al. [37]. Afterwards, different approaches for thresholding methods are stated and their application for object segmentation is examined.

3.2.1 Object Segmentation using Anisotropic Diffusion

The segmentation algorithm described in [37] is based on the error frame resulting from the original frame and its background model. The difference to that approach is to use the local background Sprites for the generation of the background model introduced in Chapter 2. The segmentation approach based on the resulting error frame is shown in Fig. 3.2 and described next.

For a given reference frame taken from a video sequence, an error frame is calculated using the pre-generated background model. Afterwards, the absolute values of the error frame are computed. Ideally, only pixel values of the foreground object regions are different from zero. Due to problems during the background model generation, errors also occur in background regions between the original frame and its background model, here its local background Sprite.

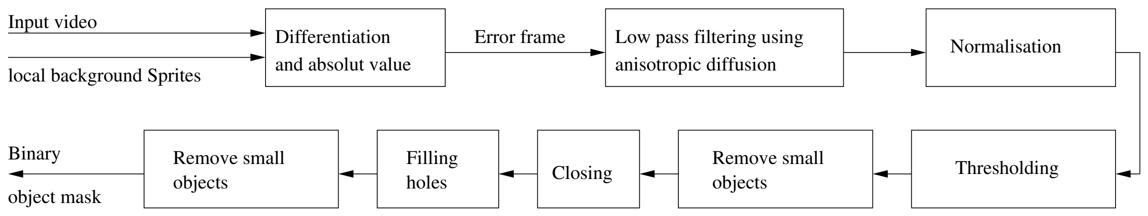


Figure 3.2: Object segmentation using anisotropic diffusion

High values of background regions of the error frame are caused by the error accumulation during the global motion estimation, the non-ideal interpolation of the transformation during the generation of the local background Sprites and applying the blending filter. These distortions are mostly of high-frequency nature. So, it is possible to minimize these errors by use of a low-pass filter. The problem here is that a common low-pass filter, e.g. a gaussian filter, does not differentiate between high-frequency regions caused by noise and high-frequency values causes by object edges. At these edges, pixel values from foreground and background edges are mixed up resulting in a blurred edge. Such object edges occur in the error frame considered. The goal is to keep these edges because they ideally describe the object's borders. An approach to solve this problem is the anisotropic diffusion filter to detect edges in an image first intoduced in [53]. The great advantage of this anisotropic filter is that edges of adjacent pixels with small intensity values are low-pass filtered but adjacent pixels with large intensity values are not low-pass filtered.

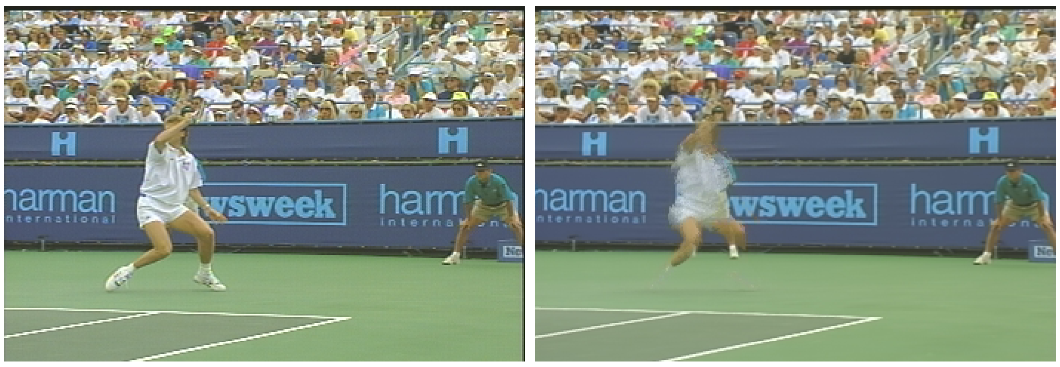
The algorithm is based on the diffusion equation and can be formulated as:

$$I_t = \mathrm{div}(c(x, y)\nabla I(x, y)) \tag{3.1}$$

where $c(x, y)$ describes the diffusion coefficient and $\nabla I(x, y)$ is the gradient of the image $I(x, y)$. If $c(x, y)$ is constant then I_t is reduced to the isotropic diffusion equation. Assuming the object edges are known, the optimal choice of c is $c(x, y) = 1$ for object regions and $c(x, y) = 0$ for object edges. In this case, object regions are low-pass filtered and object edges are not low-pass filtered. Practically, the object edges are not known. Therefore, an estimation is necessary. Such an estimation can be the absolute value of the gradient of the image. Then, $c(x, y)$ is a function $g(\cdot)$ of this estimation:

$$c(x, y) = g(||\nabla I(x, y)||). \tag{3.2}$$

For function $g(\cdot)$, a non-negative monotonically decreasing function is chosen with $g(0) = 1$. That means, in regions with small gradient values, i.e. for
$||\nabla I(x, y)|| \to 0$, $g(||\nabla I(x, y)||) \to 1$. This leads to a more intense smoothing in homogeneous regions. In our case, the following equation for the diffusion coefficient is used:



(a) Original frame (b) Local background Sprite

Figure 3.3: Original frame and local background Sprite, sequence “Stefan”, frame 200

$$
\begin{aligned}
c(x,y) &= g(||\nabla I(x,y)||) \\
&= \frac{1}{1+\left(\frac{||\nabla I(x,y)||}{\kappa}\right)^2},
\end{aligned}
\tag{3.3}
$$

where κ is a constant. We plug this in Equ. 3.1 and the anisotropic diffusion equation results:

$$
\begin{aligned}
I_t &= \operatorname{div}\left(c(x,y)\nabla I(x,y)\right) \\
&= \operatorname{div}\left(\frac{1}{1+\left(\frac{||\nabla I(x,y)||}{\kappa}\right)^2}\nabla I(x,y)\right).
\end{aligned}
\tag{3.4}
$$

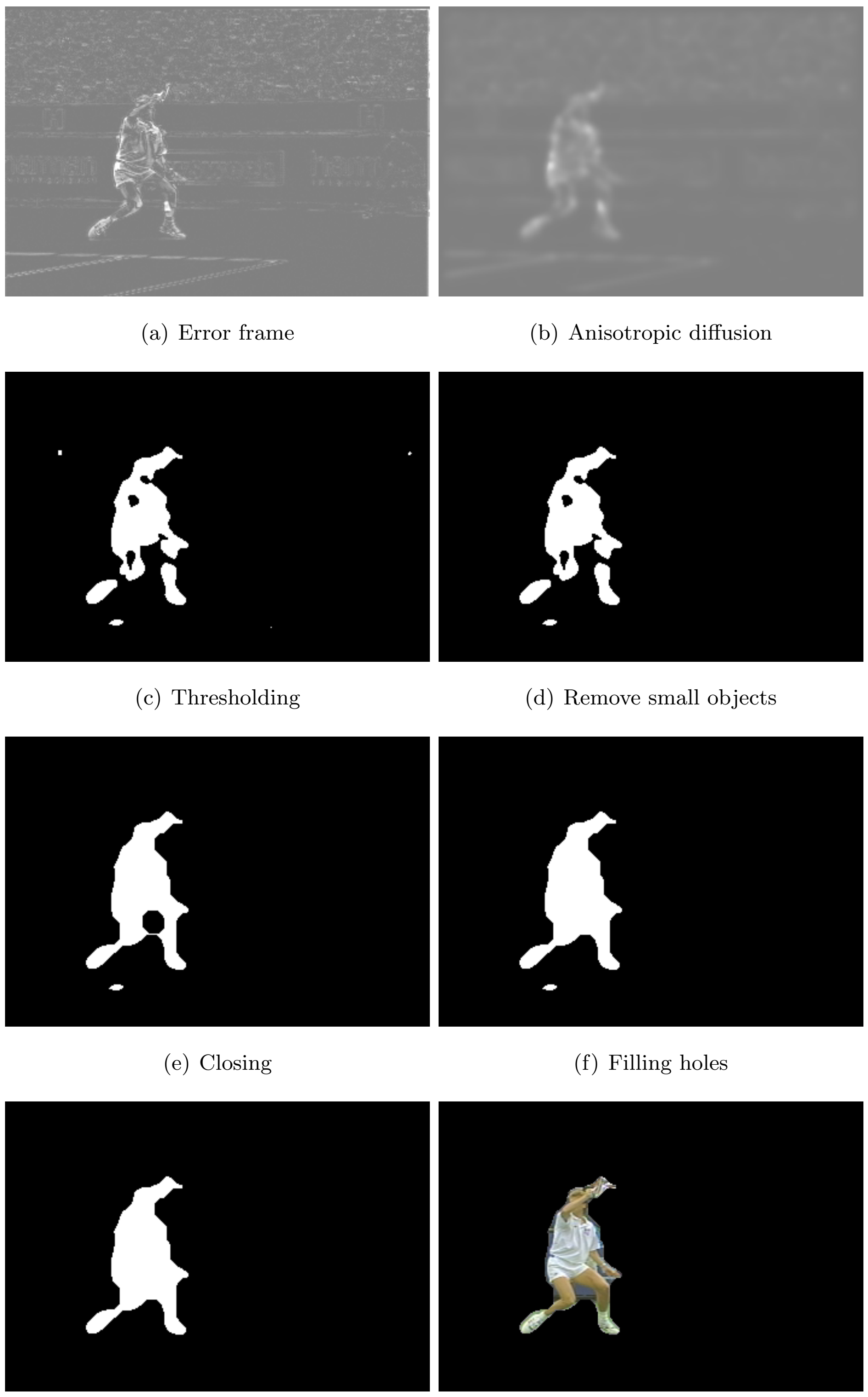
After low-pass filtering of the error frame using the anisotropic diffusion, the error frame is normalized, i.e. scaled to the range of values $[0,1]$, and binarized by thresholding. A first version of an object mask results. The threshold value t is calculated by the following equation:

$$
t = \operatorname{mean}\left(I_{t,norm}\right) + \varsigma \cdot \left(\max\left(I_{t,norm}\right) - \operatorname{mean}\left(I_{t,norm}\right)\right) \tag{3.5}
$$

where $I_{t,norm}$ is the normalized anisotropic filtered error frame and ς is a tuning constant of the threshold.

The resulting object mask is now treated by a number of morphological operators. First, small objects are removed using a pre-defined value of pixels in the region.

Afterwards, a closing is performed to connect two or more closely adjacent objects. Assuming the foreground objects contain no holes, these are filled out. Finally, small objects



(a) Error frame

(b) Anisotropic diffusion

(c) Thresholding

(d) Remove small objects

(e) Closing

(f) Filling holes

(g) Remove small objects

(h) Segmented object

Figure 3.4: Steps of object segmentation using anisotropic diffusion, sequence “Stefan”, frame 200

are removed again with a higher treshold than used previously. The object mask achieved is the final result of the segmentation using anisotropic diffusion.

Every step of the segmentation approach is now visualized at one example. For that, frame 200 taken from the test sequence “Stefan” is chosen for the reference frame shown in Fig. 3.3(a) and its corresponding local background Sprite using the maximum criterion Fig. 3.3(b). It can be seen in the local background Sprite that the foreground object is not completely removed. However, this is not necessary because for the segmentation process only the object edges are of interest. After subtraction and calculating the absolute value of the frames shown in Fig. 3.3, the error frame results (Fig.3.4(a)). The next step is the smoothing operation of the error frame using anisotropic diffusion. The result for the considered example is shown in Fig. 3.4(b). It can be clearly seen that the background is highly smoothed, but the edges of the foreground object are still remaining. The first preliminary object mask is generated using thresholding applying Equ. 3.5 (Fig. 3.4(c)). The remaining problems, which are tackled by the morphological operators, can be seen in this frame. There are small errors in the background area, the legs of the tennis player are not connected with the body and there are also small errors in the body of the player. The results of each step of the morphological operator are shown in Fig. 3.4(d) - 3.4(g). Finally, Fig. 3.4(h) depicts the completed segmentation result.

3.2.2 Thresholding

The subtraction of the background model from an original frame and the following thresholding of the resulting error frame is a means, which is very often used for detecting areas which have changed relative to the background model. The thresholding algorithm proposed above is applied on the filtered version of the error frame by the anisotropic diffusion step. The threshold is a weighting operation and is calculated by :

$$t = \mathrm{mean}\left(I_{t,norm}\right) + \varsigma \cdot \left(\max\left(I_{t,norm}\right) - \mathrm{mean}\left(I_{t,norm}\right)\right), \tag{3.6}$$

where $I_{t,norm}$ is the normalized filtered error frame and ς is a tuning constant. To evaluate the performance of this approach, we compare it with various alternative thresholding algorithms in the prior art. For that, we use the best approaches compared in Rosin et al. [58]. A comprehensive experimental evaluation is performed later in this chapter.

The first thresholding approach proposed by Kapur et al. [23] assumes that the grayscale histogram of an image consists of two probability density functions, i.e. one of the grayscale distribution lower than a threshold and the other taken from the grayscale distribution higher or equal to a threshold. The algorithm maxmimizes the sum of the entropies of both distributions to calculate the threshold t. The maximum entropy, i.e. the maximum mean information, has a grayscale distribution when the relative frequency is the same for

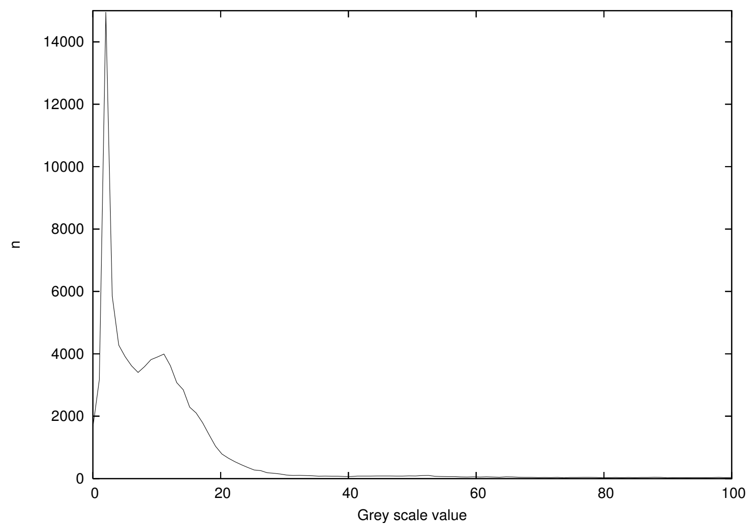


Figure 3.5: Grayscale histogram with unimodal distribution of an anisotropic filtered error frame, sequence “Stefan”, reference frame 230

all grayscale values. The grayscale histograms of the error frames calculated in this work are generally described by a unimodal distribution. Figure 3.5 depicts a typical grayscale histogram of a filtered error frame by anisotropic diffusion. Most of the pixel values are near zero. These pixels correspond to the background areas of the error frame. High error values coming from the foreground areas are very rare. Thus, the maximization of the sum of both distributions requires a threshold, which is very close to the minimum of the unimodal distribution, because in this case all grayscale values from each side of the threshold are uniformly distributed.

The algorithm calculates both distributions in the first step :

$$\begin{aligned} A &: \frac{p_0}{p_s}, \frac{p_1}{p_s}, \dots, \frac{p_t}{p_s} \\ B &: \frac{p_{t+1}}{1-p_s}, \frac{p_{t+2}}{1-p_s}, \dots, \frac{p_{255}}{1-p_s} \end{aligned} \tag{3.7}$$

where $p_0, p_1, ..., p_{255}$ are the probabilities of each grayscale value in the error frame and $p_s = \sum_{i=0}^{t} p_t$ is the sum of the probabilities of the first distribution. For each possible s the entropies of the distributions

$$\begin{aligned} H(A) &= -\sum_{i=0}^{t} \frac{p_i}{p_s} \ln \frac{p_i}{p_s} \\ H(B) &= -\sum_{i=t+1}^{255} \frac{p_i}{1-p_s} \ln \frac{p_i}{1-p_s} \end{aligned} \tag{3.8}$$

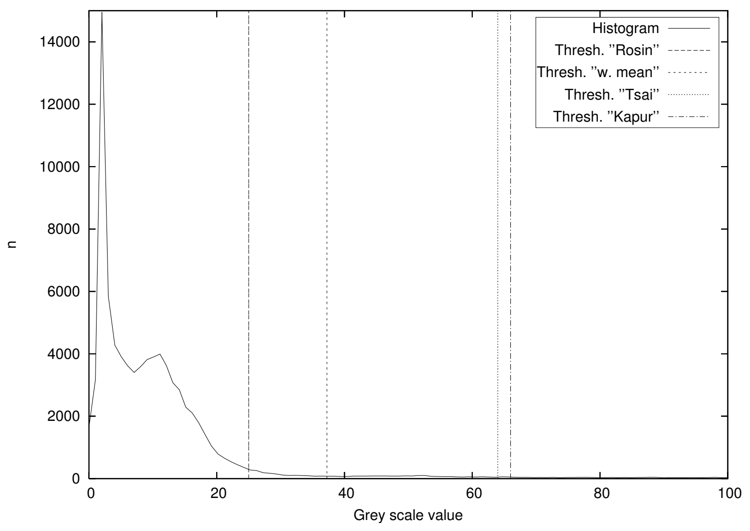


Figure 3.6: Thresholds for grayscale histogram of an anisotropic filtered error frame, sequence “Stefan”, reference frame 230

are calculated. The discrete value t, which maximizes the sum of the entropies $\Phi = H(A) + H(B)$, is the threshold to be found. The method described is applied on the example frame 230 of the test sequence “Stefan”. The resulting threshold can be seen in Fig. 3.6. In comparison to other algorithms, the threshold value $t = 66$ is relatively high, which leads to an under segmentation for this example, as seen in Fig. 3.8(b).

The work of Rosin [57] assumes a unimodal distribution, which means that at the lower bound of the grayscale spectrum is a dominant summit. This summit is caused by small errors of the background areas. The errors of the foreground areas are at the upper bound of the spectrum, i.e. in higher grayscale value ranges. It is not necessary that these errors of the foreground areas produce their own summit. The only constraint is that these errors are as far as possible at the upper bound of the spectrum and thus do not fall under the distribution of the errors of the background. For the calculation of the threshold, first a line is drawn between the summit of the histogram, i.e. between the highest value and the value with the lowest relative frequency. The value of the histogram is described as $H(i)$ for $i \in [0, 255]$. The threshold t is then calculated by maximizing the distance d dropping the perpendicular of the line on the point $(i, H(i))$ of the histogram. Figure 3.7 depicts this geometric method. The threshold obtained for the error frame 230 of the test sequence “Stefan” can be seen in Fig. 3.6. It is obvious that the threshold calculated by Rosin’s algorithm is the lowest comparing to the remaining threshold values. For the chosen example, the segmentation result is good as depicted in Fig. 3.8(c). However, setting the threshold too low leads to an under segmentation.

The last thresholding algorithm introduced in this work has been proposed by Tsai [84]. This algorithm calculates the threshold by keeping the first three momentums of the error frame in a dual significant higher contrast version. A momentum is a term taken from

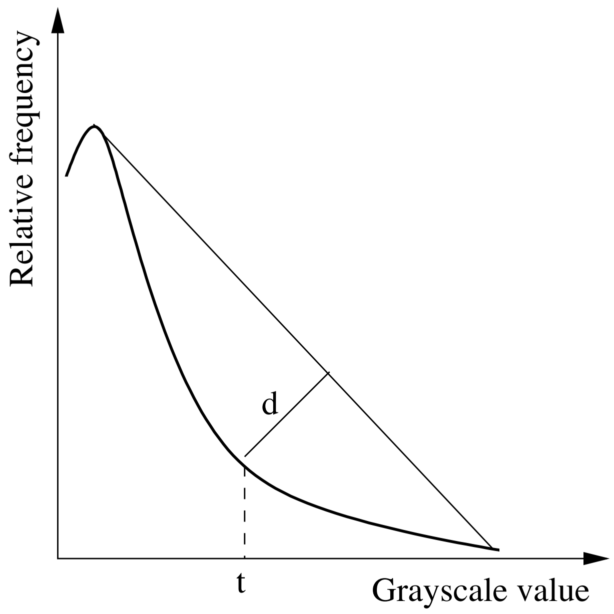


Figure 3.7: Thresholding method of Rosin

statistics. These momentums describe for example expectation value, variance, skewness or kurtosis of a distribution function. The momentum of the k-th order of the random variable X is the expectation value of k to the power of X and thus, is ascertained for the error frame $I(x, y)$:

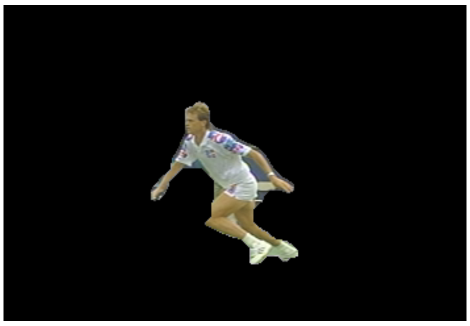
$$m_k = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} I(x_i, y_j)^k, \qquad k = 1,\, 2,\, 3,\, \ldots \tag{3.9}$$

where MxN is the size of the error frame. The momentums can also be calculated from the histogram $H(i)$ of the error frame

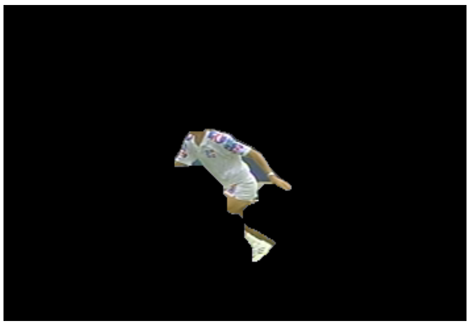
$$m_k = \frac{1}{MN} \sum_{i=0}^{255} H(i) \cdot i^k = \sum_{i=0}^{255} p_i \cdot i^k. \tag{3.10}$$

The variable $p_i = \frac{H(i)}{MN}$ is the probability of a grayscale value being in the error frame. The error frame is considered as a non-sharp version of a dual significant higher contrast version with two grayscale values, i.e. i_{below} and i_{above}. The algorithm chooses a threshold t so that the first three momentums of the error frame in the dual significant version are kept when all values below the threshold are replaced by i_{below} and above the threshold are replaced by i_{above}. Now p_{below} and p_{above} are the probabilities that a pixel value is below or above the threshold t. The first three momentums of the dual significant version are then calculated as:

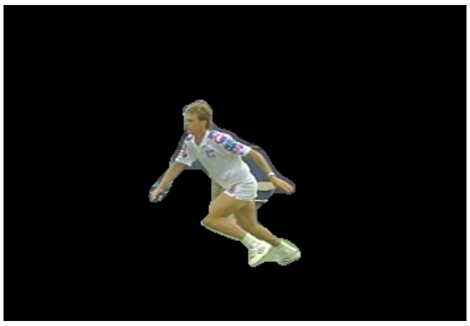
$$m'_k = p_{below} \cdot i_{below}^k + p_{above} \cdot i_{above}^k \tag{3.11}$$



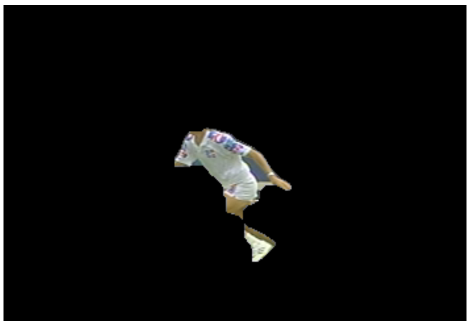
(a) Threshold "weighted mean"



(b) Threshold "Kapur"



(c) Threshold "Rosin"



(d) Threshold "Tsai"

Figure 3.8: Segmentation results using various thresholding methods, sequence "Stefan", reference frame 230

where $p_{below} + p_{above} = 1$ is valid and keeping the first three momentums means that $m'_k = m_k$. The complete formulae are:

$$\begin{aligned}
p_{below} \cdot i^0_{below} + p_{above} \cdot i^0_{above} &= m_0 \\
p_{below} \cdot i^1_{below} + p_{above} \cdot i^1_{above} &= m_1 \\
p_{below} \cdot i^2_{below} + p_{above} \cdot i^2_{above} &= m_2 \\
p_{below} \cdot i^3_{below} + p_{above} \cdot i^3_{above} &= m_3
\end{aligned} \tag{3.12}$$

where $m_0 = 1$. To determine the threshold t Equs. 3.12 have to be solved to calculate i_{below}, i_{above}, p_{below}, and p_{above}. The threshold is chosen so that

$$p_{below} = \frac{1}{MN} \sum_{\forall i \leq t} H(i) \tag{3.13}$$

is valid. The threshold of Tsai's method applied to the error frame 230 of the test sequence "Stefan" is shown in Fig. 3.6. The threshold is relatively high with $t = 64$ like the threshold calculated with Kapur's method. The result of the segmentation can be seen in Fig. 3.8(d).

3.2.3 Segmentation using the YUV color space

To improve the segmentation results we now examine the use of the full YUV color space. Cavallaro et al. [3] have used the full YUV color space for segmentation with color edges. Inspired by this method in this work all three color channels are coupled. It is assumed that some errors cannot be detected in the luminance component, e.g. when local background Sprites are generated and background regions appear with the same luminance values as values of the foreground objects.

Considering a video sequence in a YUV color space, it is assumed that a background model sequence (local background Sprites) is generated and is also transformed in the YUV color space. For the segmentation, the U- and V-components of the reference frame and its corresponding background model are up-sampled. Then, for each color channel a separate error frame is calculated by differentation and calculation of the absolute values.

Each of the error frames is then combined by the following:

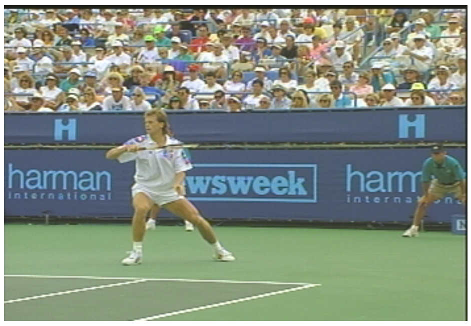
$$I_{error}(x,y) = I_{error,Y}(x,y) + I_{error,U}(x,y) + I_{error,V}(x,y) \tag{3.14}$$

and afterwards scaled to the pixel range $[0, 255]$. Due to this coupling, errors in the foreground regions are increased, because it can be assumed that the chrominance values of the reference frame and background model are highly different. Further segmentation steps are equivalent as described above including an anisotropic diffusion filtering.

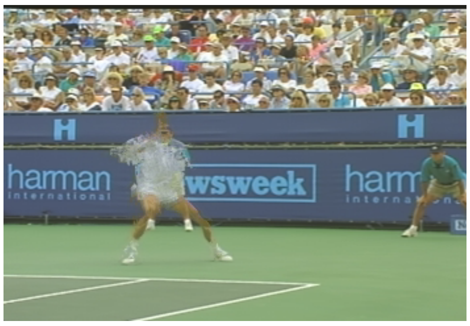
For visualization, an outlier of the segmentation only by the use of the luminance component is taken into account for sequence “Stefan”. The result of using the full YUV color space can be seen in Fig. 3.9. Figure 3.9 (a) and (b) depict the original frame and its corresponding background model of the given example. The error frames of each of the color channels are shown in Fig. 3.9 (c) - (e). The combination of all color space components lead to the result shown in Fig. 3.9 (f). It can easily be seen that the error values are increased in the foreground object region. A comparison between the use of only the luminance component and the full YUV color space for the segmentation process can be seen in Fig. 3.9(g) and (h).

3.3 Object Segmentation using short-term global motion estimation

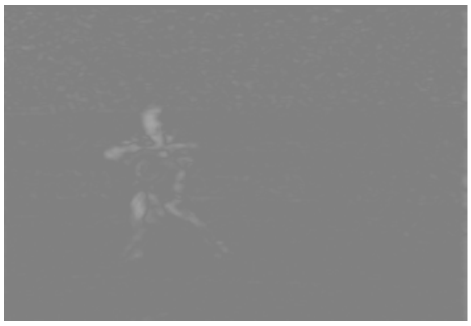
The first approach to apply automatic object segmentation is the use of global motion estimation. Here, global motion estimation means that the background motion of two consecutive frames of a video sequence is estimated. If an error image is calculated using the reference frame and the compensated frame, the background pixels are ideally removed



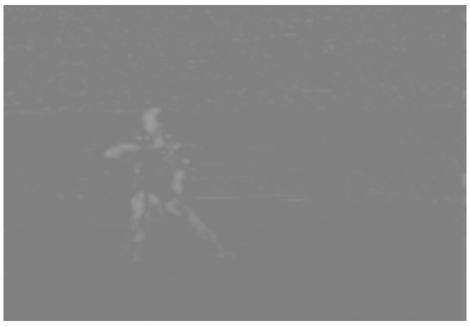
(a) Original frame



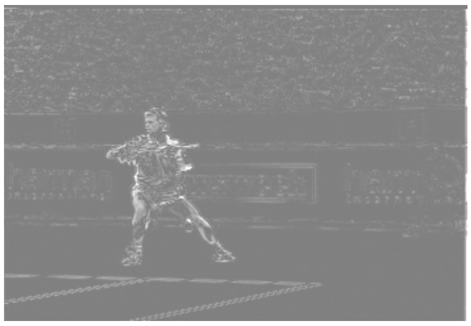
(b) Local background Sprite



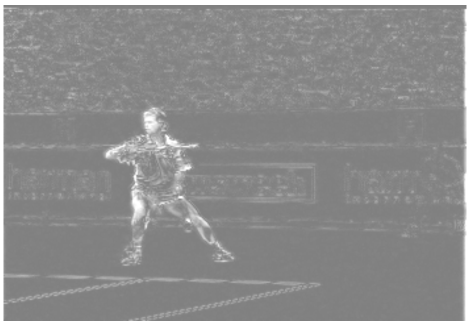
(c) Error frame, U



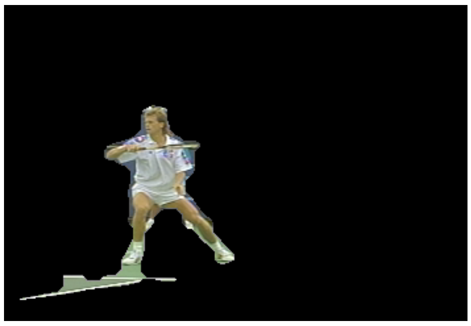
(d) Error frame, V



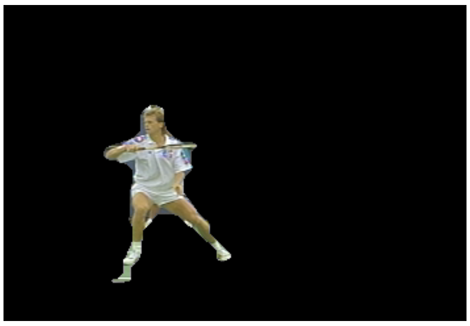
(e) Error frame, Y



(f) Combined error frame



(g) Result (YUV400)



(h) Result (YUV420)

Figure 3.9: Segmentation using the full YUV color space, sequence “Stefan”, reference frame 210

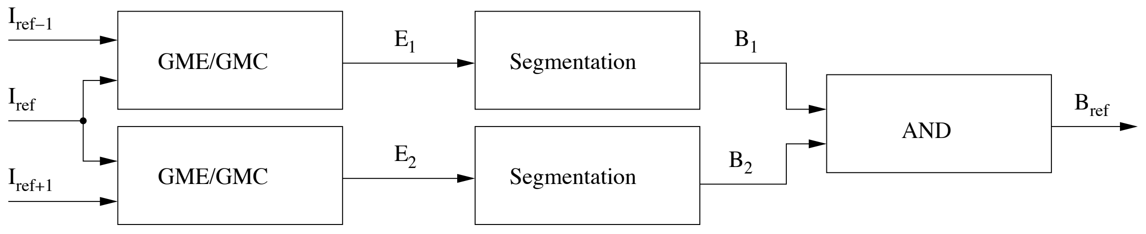


Figure 3.10: Motion-based Object Segmentation using Global Motion Estimation

completely. However, the foreground object regions appear from the reference and the compensated frame in the error image. To extract the foreground objects more precisely, two error images are taken into account: i) the error image calculated by the reference frame and the compensated previous frame, and ii) the error image calculated by the reference frame and the next frame. The segmentation algorithm described above is applied on these two error frames. Two binary masks are obtained and combined by an AND-operator to achieve the final foreground object mask. Figure 3.10 illustrates the whole processing chain. This approach is inspired by earlier work where only global motion estimation techniques where used for the pre-pocessing step [49].

3.4 Segmentation using short-term global motion estimation and background Sprites

The segmentation approach relies on pre-computed error frames. It depends on the significant difference of the pixel values of the foreground and the background object. The use of two error frames improves the segmentation results remarkably. Computing the error image using the reconstructed background frames from the Sprite produces a precise shape of the foreground object. The drawback is that due to Sprite generation further regions of larger values appear in the error frame which are not related to the foreground objects. To prevent this, it is very important to have a very good estimation of the background object. This can be achieved by the use of the error frame by the frame-to-frame image registration. Here the background object can be estimated more precisely while the foreground objects appear twice in the error image. Combining both calculated objects masks using a logical AND-operator removes some drawbacks from both approaches. The block chart of the whole segmentation algorithm is given in Fig. 3.11.

3.5 Object Segmentation using local background Sprites

The third object segmentation algorithm used [29] includes a new background modeling. In practice, accumulated long-term parameters, which set up the Sprite generation process, produce distortions in the background Sprite image. Furthermore, the warping process has

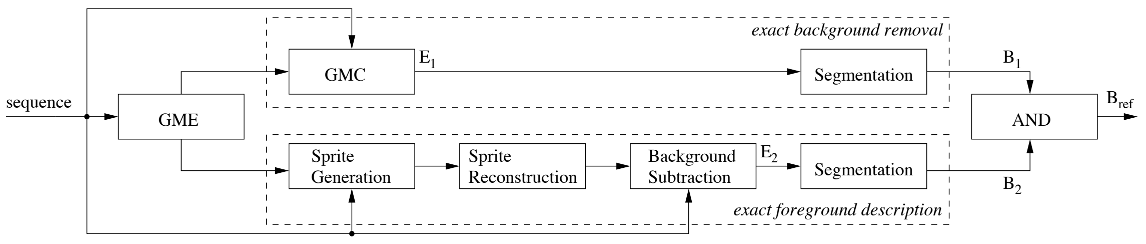


Figure 3.11: Motion-based Object Segmentation using Anisotropic Filtering

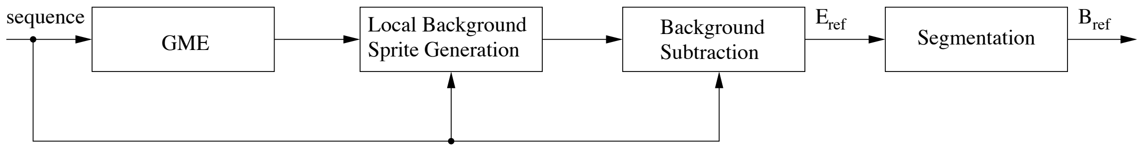


Figure 3.12: Proposed segmentation algorithm

to be applied two times, first to align each considered image to the background Sprite plane and second to reconstruct the image from the background Sprite plane. For the new technique, we generate so-called local background Sprites. That means, we produce background Sprites for each frame of the video. During the generation process, we only consider pixels which fall in the current reference frame. Using this method, we are able to produce accurate background models for each frame of the sequence. The whole segmentation algorithm is then processed in a common background subtraction scheme (see Fig. 3.12).

3.6 Experimental results

To evaluate the object segmentation algorithms described above, object segmentation masks are built using the background models and thresholding algorithms at five test sequences. First, the test sequences used are introduced. Afterwards, the metrics are described which will be used for the objective evaluation.

3.6.1 Test sequences

For evaluation of the background modeling techniques and segmentation algorithms described above, four sequences with different content, features, and camera motion are considered:

- *Allstars (Shot 1)*, size 352x288, 250 frames
- *Mountain*, size 352x192, 100 frames
- *Race1 (View 0)*, size 544x336, 100 frames
- *Stefan*, 352x240, 300 frames

All sequences are known and already used in the previous chapter.

3.6.2 Precision, Recall and F-Measure

Ground truth data is available for all four test sequences named in the previous section. These ground truth masks are generated manually and show the real foreground/background segmentation for all frames of each sequence. By use of these ground truth masks, the well-known F-measure metric can be used to evaluate the performance of the automatic object segmentation algorithms. The F-measure value is calculated as follows:

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{3.15}$$

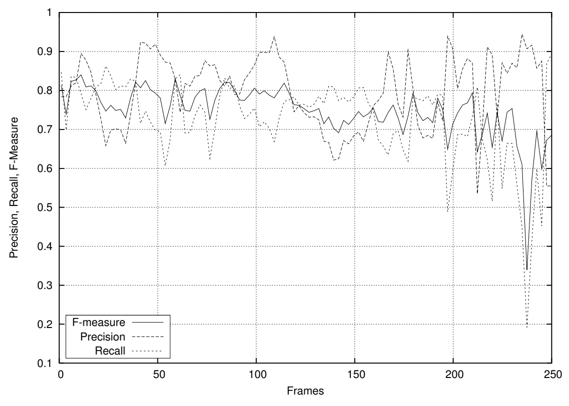
Precision and Recall describe the quality of the foreground and background objects and are calculated :

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN}, \end{aligned} \tag{3.16}$$

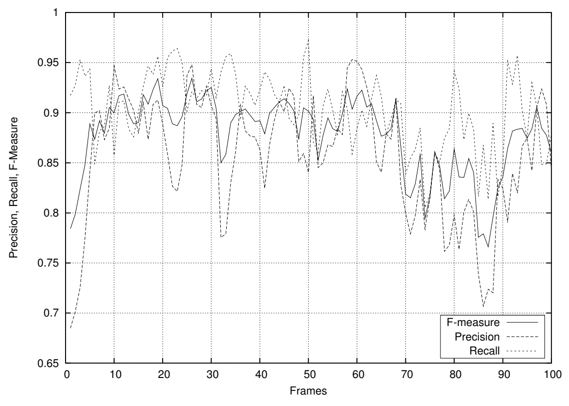
where TP (true positive) is the part of pixels which can be found in the ground truth mask as well as in the automatic object mask to be evaluated. FP (false positive) describes the part of pixels which only appear in the automatic object mask, i.e. wrongly-segmented foreground object pixels. FN (false negative) is the part of pixels which can only be found in the ground truth mask, i.e. foreground pixels not detected by the automatic algorithm.

3.7 Segmentation results

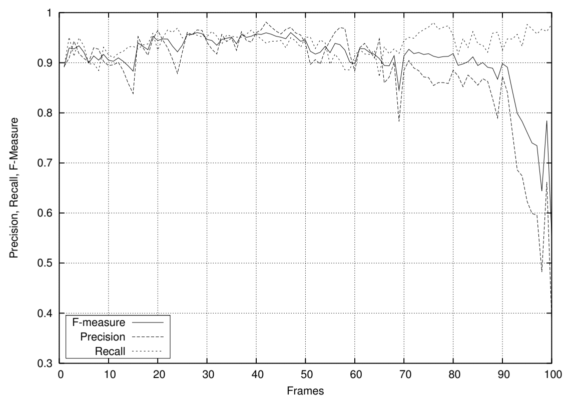
The evaluation of the segmentation and the challenge to find the best combination of background modeling technique and thresholding algorithm are determined as follows. Three background modeling techniques are taken into account. The algorithm proposed by Mech et al. [49] is the first pre-processing algorithm only based on global motion estimation. It is further called “GME-based”. The second algorithm combining the short-term global motion estimation step and global background Sprites (Krutz et al. [36]) is further called ‘GME-/Sprites-based”. The segmentation is determined for each test sequence with different combinations. All various algorithms generating the local background Sprites are used (maximum, variance, eigenvalues). Furthermore, the segmentation is accomplished in color spaces YUV400 (only luminance information) and YUV420 (full YUV color space). For thresholding algorithms, the weighted mean method used in (Krutz et al. [36]), the thresholding algorithm proposed by Kapur et al. [23], the methods by Rosin [57] and Tsai [84] are taken into account. Tables 3.1 - 3.4 show the mean values of Precision, Recall, and F-measure for all combinations of all sequences.



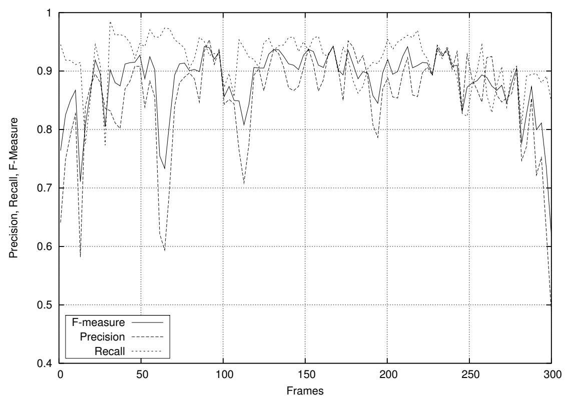
(a) "Allstars"



(b) "Mountain"



(c) "Race1"



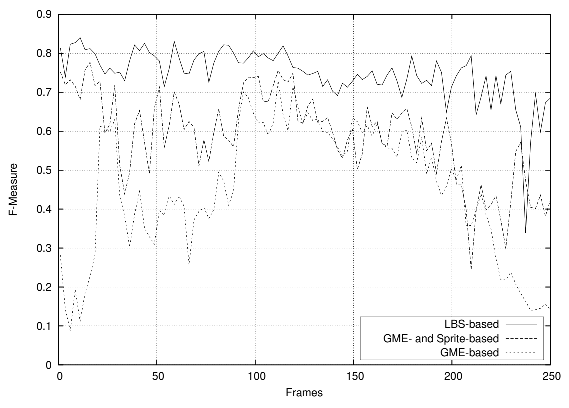
(d) "Stefan'

Figure 3.13: Precision, Recall and F-Measure curves for best segmentation results of all test sequences

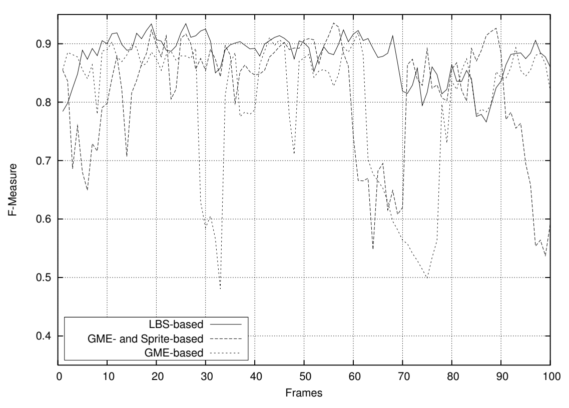
Basically, it can be said that a final decision for choosing one criterion for generating the local background Sprites cannot be made. However, the best segmentation results can be achieved using the maximum or variance criterion. A reason for that is the fixed set of thresholds during the local background Sprite generation process. The generation of local background Sprites using the eigenvalue criterion leads to the largest amount of adjacent frames and therefore causes a big error in the background regions. The use of the full YUV420 color space brings an improvement in every test sequence considered. It can be seen that when the full YUV420 color space is used the mean F-measure values are higher than the results achieved only by use of the luminance component. Considering the thresholding methods, the algorithm proposed by Krutz et al. [36] outperforms the competitive methods used in this work in nearly all cases.

The best configuration of the segmentation algorithm using local background Sprites is now discussed for each sequence considered. Three parts of the whole segmentation process are evaluated in the following order:

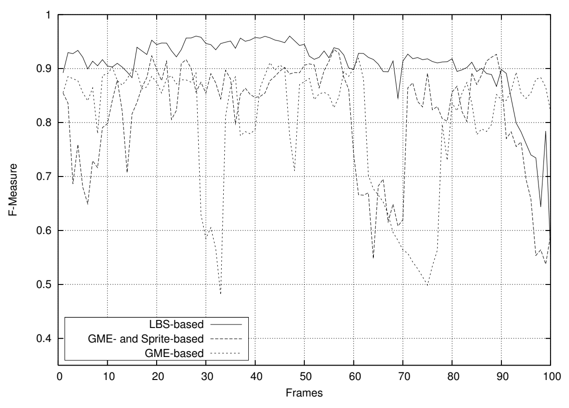
- the breaking criterion during the local background Sprite generation step,



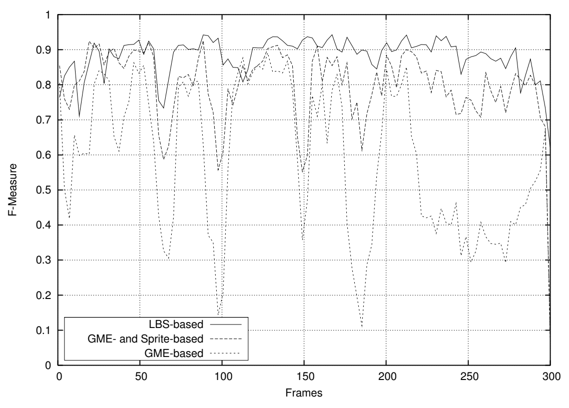
(a) “Allstars”



(b) “Mountain”



(c) “Race1”



(d) “Stefan’

Figure 3.14: F-Measure curves comparing all segmentation algorithms

- the color space, and
- the thresholding method.

The best segmentation configuration for the sequence “Allstars” is “Variance-YUV420-Krutz”. The mean F-measure value is 0.75. This means an improvement of about 16% compared to the GME-/Global background Sprites-based approach. Figure 3.13 (a) shows the curves for Precision, Recall, and F-Measure for each frame of the sequence. The F-measure curves comparing the proposed segmentation algorithm using local background Sprites and the reference algorithms are shown in Fig. 3.14 (a). It is obvious that the proposed algorithm outperforms the reference algorithms at every time of the test sequence. Additionally, it should be mentioned that the logo of the TV-station and the scores inside the frames are foreground objects and are segmented as such by the automatic object segmentation algorithm. However, these regions are not marked as foreground in the groundtruth masks. Therefore, they are removed from the automatic object masks before the calculation of the metrics (P, R, F), because otherwise they would have influenced the results very strong in a negative manner. Examples for the segmentation results from selected reference frames can be seen in Fig. 3.15.

The segmentation results of the sequence “Mountain” are improved with the combination “Maximum-YUV420-Krutz” about 2% in the F-measure value in comparison to the GME-/Global background Sprites-approach. The curves of Precision, Recall, and F-measure along the sequence are shown in Fig. 3.13(b). The F-measure curves comparing the proposed segmentation algorithm using local background Sprites and the reference algorithms using global motion estimation and conventional background Sprites, respectively, are shown in Fig. 3.14 (b). The proposed algorithm outperforms the reference algorithms also at this test sequence, except at the beginning of the sequence and between frames 80 and 90. A reason for this effect at the beginning is the generation of the local background Sprites, which causes a sub-optimal object segmentation. It can be mentioned that we have reached the limit of the object segmentation for this test sequence. Due to the features of the test sequence, the simple translational camera motion and the high-frequency background content, the segmentation is relatively easy. Some segmentation results can be seen in Fig. 3.16.

The test sequence “Race1” can be best segmented using the combination “Maximum-YUV420-Krutz”, like test sequence “Mountain”. The F-measure value of about 0.91 brings out an improvement of 6% compared to the reference algorithm using GME/Global background Sprites. Figure 3.13(c) shows the curves for Precision, Recall, and F-measure along the sequence. The comparison to the reference algorithms is depicted in Fig. 3.14 (c). The dropping of the Precision curve and thus the F-measure curve at the end of the sequence is caused by the fact that the foreground objects become smaller and smaller and very slight differences between the automatically segmented mask and the groundtruth mask cause a dropping of the Precision value. Subjectively, the automatic segmentation at the end of the sequence is very good, which does not comply with the metric in this case. Figure 3.17 shows some examples of segmentation results.

The best segmentation results for the test sequence “Stefan” can be achieved with the combination “Variance-YUV420-Krutz”. It shows a mean F-measure value of 0.88. This means an improvement of 8% compared to the reference approach. The Precision, Recall, and F-measure curves are shown in Fig. 3.13(d). Figure 3.14(d) depicts the comparison of the F-measure curves of each object segmentation considered. The proposed algorithm outperforms the reference algorithms nearly complete, except in the beginning, the end and around frame 60 of the sequence. The reason for that is, as found for test sequence “Mountain”, the sub-optimal generation of the local background Sprites. Examples of segmentation results of this test sequences can be seen in Fig. 3.18.

Finally, we would like to consider problems of calculating the quality metrics Precision, Recall, and F-measure. The calculation of these metrics is strongly constrained to the size of the foreground objects. The reason for dropping of the Precision values in the last part of the sequence “Race1” can be found in these problems. The quality of the automatic segmentation in these parts is subjectively very good as seen in Fig. 3.17(f). Small errors at the border of the foreground objects get a significant higher weighting than the same errors at a larger foreground object. Comparing the relation of foreground object size and frame size between test sequence “Allstars” (1.14%) and test sequence “Stefan” (4.87%) it can be concluded that a mean F-measure value of 0.75 achieved at “Allstars” does not mean that the subjective quality of the object segmentation is worse than at sequence “Stefan” with a mean F-measure value of 0.88.

For complete subjective evaluation of the automatic segmentation results, the test videos are provided along with the results of the three algorithms. Figures 3.19 - 3.22 show all results compared to each other and the original test video. If the Adobe Reader is used for displaying this thesis one can watch the videos by simply clicking on the images.

Criterion	**Color space**	**Threshold**	**P**	**R**	**F**
GME-based			0.425552	0.613733	0.451298
GME-/Sprites-based			0.517529	0.741405	0.586007
Maximum	YUV400	Krutz	0.632925	0.831003	0.709675
		Kapur	0.761088	0.626171	0.649270
		Rosin	0.284743	0.952388	0.428509
		Tsai	0.796993	0.605248	0.665927
	YUV420	Krutz	0.808996	0.665982	0.714954
		Kapur	0.896936	0.441389	0.546008
		Rosin	0.423238	0.900443	0.566131
		Tsai	0.901423	0.439731	0.571272
Variance	YUV400	Krutz	0.631624	0.856823	0.719049
		Kapur	0.758369	0.688658	0.695542
		Rosin	0.290422	0.955762	0.437267
		Tsai	0.800360	0.634944	0.689447
	YUV420	Krutz	0.793833	0.727931	**0.748910**
		Kapur	0.893696	0.495194	0.594747
		Rosin	0.411722	0.920498	0.561170
		Tsai	0.904166	0.476146	0.609195
Eigenvalues	YUV400	Krutz	0.629186	0.853728	0.715855
		Kapur	0.756221	0.684273	0.690021
		Rosin	0.286681	0.955789	0.432585
		Tsai	0.797105	0.631265	0.684176
	YUV420	Krutz	0.795194	0.720551	0.746808
		Kapur	0.898051	0.476174	0.578885
		Rosin	0.414888	0.917109	0.563682
		Tsai	0.902656	0.469036	0.601938

Table 3.1: Mean Precision, Recall, F-Measure for object segmentation, sequence “Allstars”

Criterion	**Color space**	**Threshold**	**P**	**R**	**F**
GME-based			0.733748	0.900068	0.800316
GME-/Sprites-based			0.783993	0.948267	0.856327
Maximum	YUV400	Krutz	0.770076	0.961241	0.853737
		Kapur	0.694412	0.826459	0.707584
		Rosin	0.643054	0.990416	0.778694
		Tsai	0.885102	0.771904	0.820279
	YUV420	Krutz	0.855914	0.903186	**0.877445**
		Kapur	0.762742	0.758881	0.698691
		Rosin	0.715913	0.978063	0.825341
		Tsai	0.921701	0.642259	0.752090
Variance	YUV400	Krutz	0.921701	0.642259	0.752090
		Kapur	0.707893	0.810795	0.698722
		Rosin	0.634746	0.991563	0.772697
		Tsai	0.885424	0.788879	0.830073
	YUV420	Krutz	0.843243	0.914051	0.875362
		Kapur	0.762035	0.749941	0.688992
		Rosin	0.707239	0.979992	0.819982
		Tsai	0.922463	0.669266	0.769790
Eigenvalues	YUV400	Krutz	0.761602	0.964695	0.849515
		Kapur	0.701561	0.808902	0.698581
		Rosin	0.636625	0.991541	0.774351
		Tsai	0.883551	0.784268	0.826489
	YUV420	Krutz	0.843179	0.913005	0.874728
		Kapur	0.762495	0.752122	0.693151
		Rosin	0.709452	0.979653	0.821560
		Tsai	0.920605	0.661771	0.764060

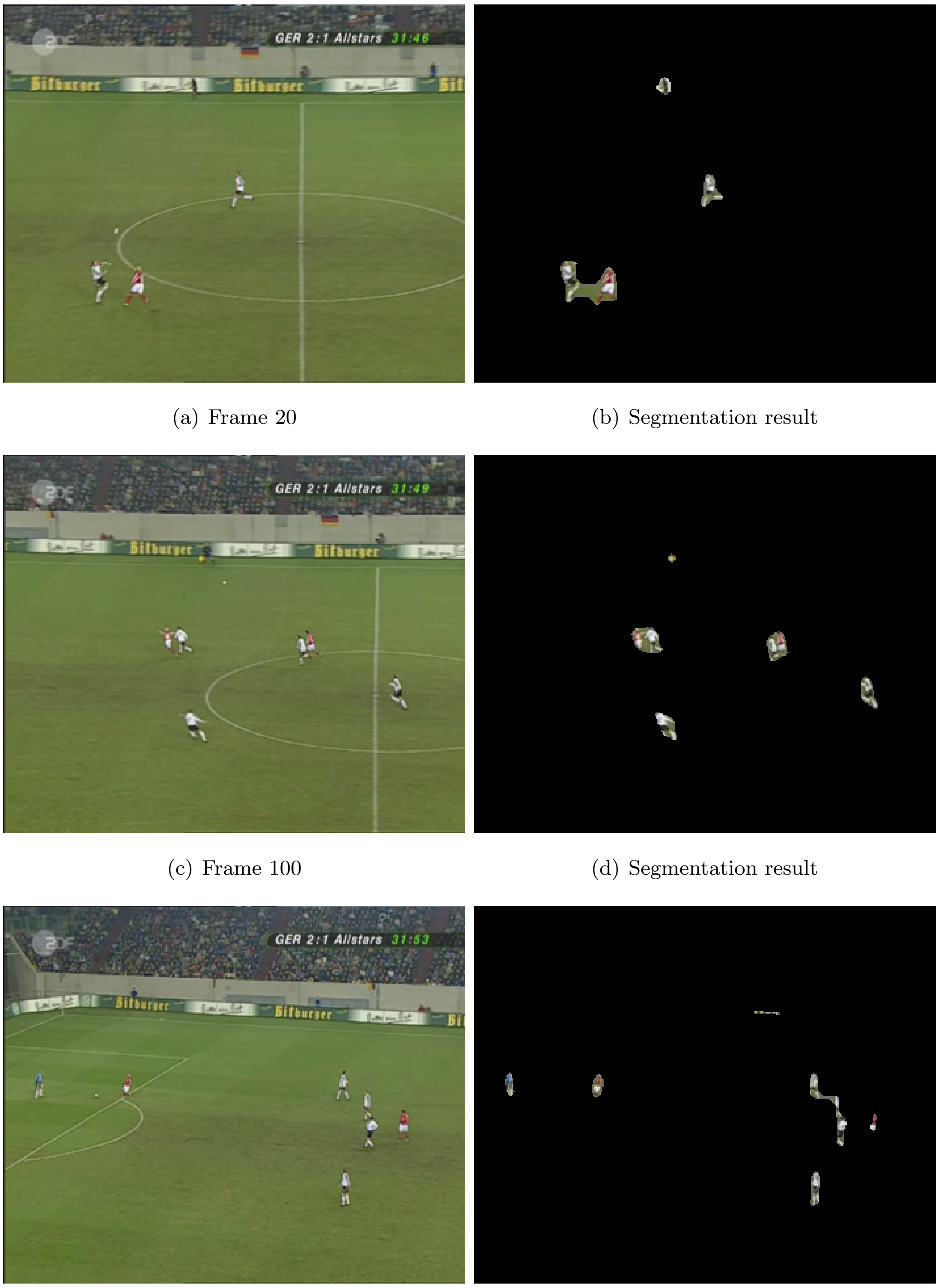
Table 3.2: Mean Precision, Recall, F-Measure for object segmentation, sequence “Mountain”

Criterion	**Color space**	**Threshold**	**P**	**R**	**F**
GME-based			0.472543	0.683493	0.549631
GME-/Sprites-based			0.839296	0.885550	0.850559
Maximum	YUV400	Krutz	0.822186	0.957921	0.880759
		Kapur	0.812560	0.876156	0.809376
		Rosin	0.561431	0.989673	0.692919
		Tsai	0.900430	0.845904	0.859218
	YUV420	Krutz	0.884592	0.939906	**0.907293**
		Kapur	0.888402	0.881022	0.873881
		Rosin	0.636176	0.982101	0.743254
		Tsai	0.946222	0.785953	0.853319
Variance	YUV400	Krutz	0.816535	0.958245	0.876954
		Kapur	0.821745	0.902617	0.836857
		Rosin	0.521364	0.989644	0.651138
		Tsai	0.879804	0.864830	0.856836
	YUV420	Krutz	0.876588	0.941778	0.903865
		Kapur	0.884690	0.870430	0.864744
		Rosin	0.648681	0.978772	0.749662
		Tsai	0.945157	0.817859	0.872767
Eigenvalues	YUV400	Krutz	0.819260	0.959035	0.878961
		Kapur	0.824505	0.903194	0.839262
		Rosin	0.536411	0.989248	0.667206
		Tsai	0.879196	0.867436	0.858064
	YUV420	Krutz	0.874992	0.943230	0.903558
		Kapur	0.886283	0.867872	0.863155
		Rosin	0.651912	0.978294	0.751454
		Tsai	0.944521	0.827196	0.877920

Table 3.3: Mean Precision, Recall, F-Measure for object segmentation, sequence “Race1 (View 0)”

Criterion	**Color space**	**Threshold**	**P**	**R**	**F**
GME-based			0.551941	0.661785	0.583526
GME-/Sprites-based			0.841140	0.785205	0.799715
Maximum	YUV400	Krutz	0.774365	0.915589	0.826461
		Kapur	0.889283	0.579552	0.657599
		Rosin	0.611475	0.969369	0.697793
		Tsai	0.825880	0.766011	0.759616
	YUV420	Krutz	0.815003	0.901533	0.844746
		Kapur	0.877864	0.764429	0.789190
		Rosin	0.743152	0.939380	0.799347
		Tsai	0.850498	0.800061	0.803932
Variance	YUV400	Krutz	0.817421	0.925477	0.862515
		Kapur	0.906797	0.637512	0.707281
		Rosin	0.684111	0.969292	0.772982
		Tsai	0.875173	0.762779	0.793374
	YUV420	Krutz	0.851775	0.917638	**0.880358**
		Kapur	0.884015	0.844527	0.852015
		Rosin	0.799943	0.947930	0.859054
		Tsai	0.893729	0.807952	0.839400
Eigenvalues	YUV400	Krutz	0.809881	0.923845	0.855927
		Kapur	0.904823	0.635296	0.705743
		Rosin	0.672835	0.969798	0.760352
		Tsai	0.867620	0.760805	0.786189
	YUV420	Krutz	0.844948	0.915854	0.874357
		Kapur	0.880520	0.829132	0.838942
		Rosin	0.791414	0.947099	0.849422
		Tsai	0.885797	0.806072	0.832551

Table 3.4: Mean Precision, Recall, F-Measure for object segmentation, sequence “Stefan”



(a) Frame 20

(b) Segmentation result

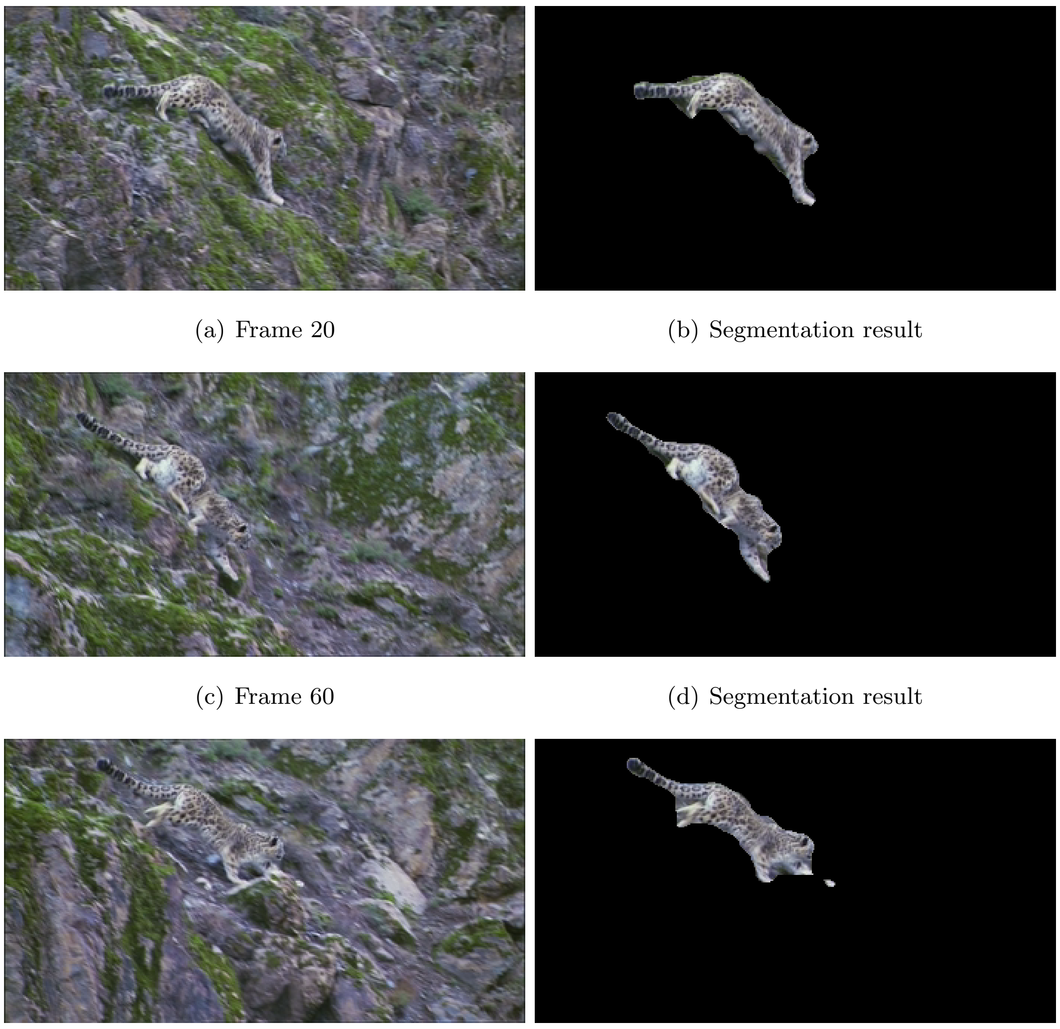
(c) Frame 100

(d) Segmentation result

(e) Frame 200

(f) Segmentation result

Figure 3.15: Example segmentation results, sequence "Allstars"

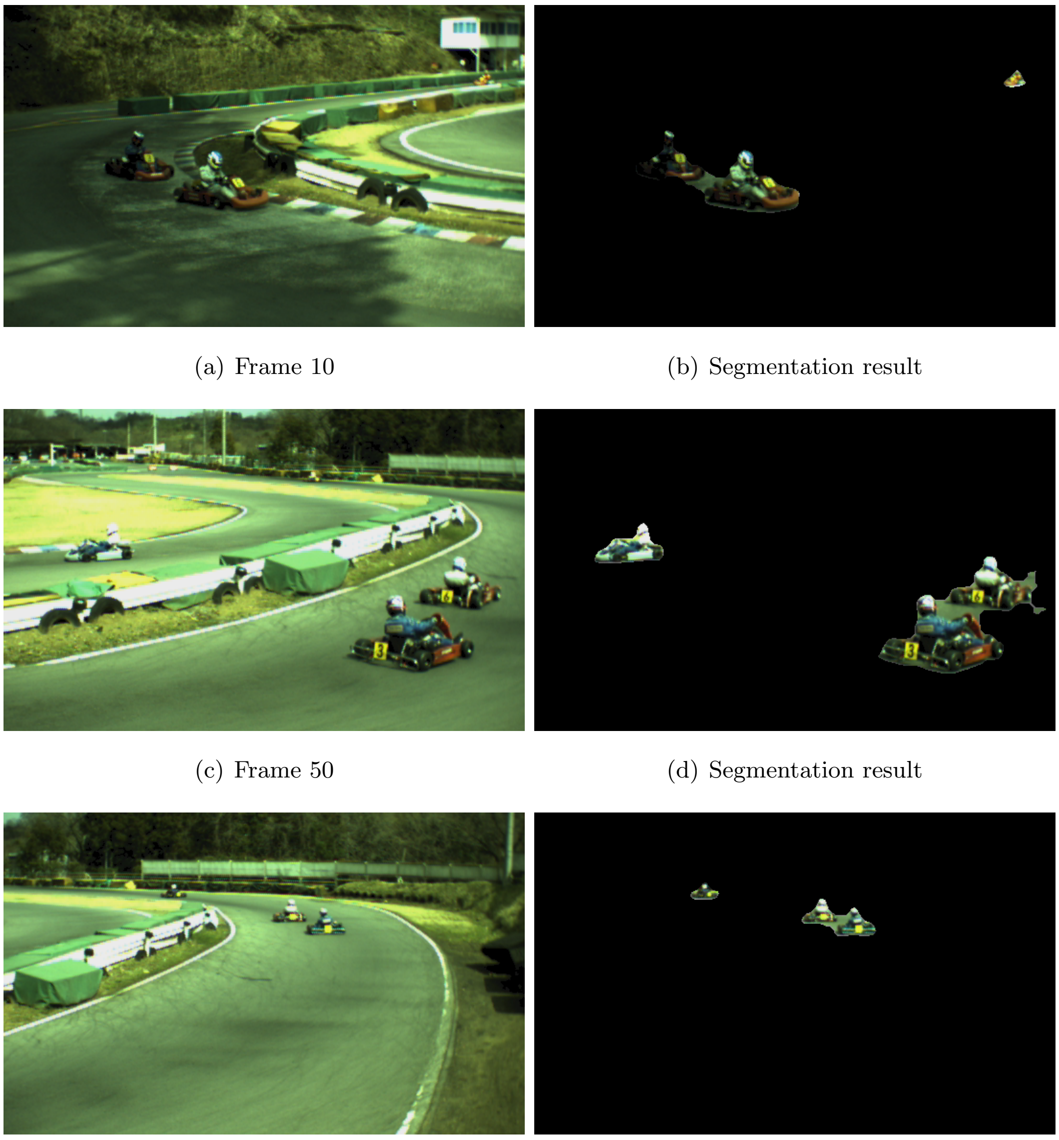


(a) Frame 20 (b) Segmentation result

(c) Frame 60 (d) Segmentation result

(e) Frame 80 (f) Segmentation result

Figure 3.16: Example segmentation results, sequence “Mountain”

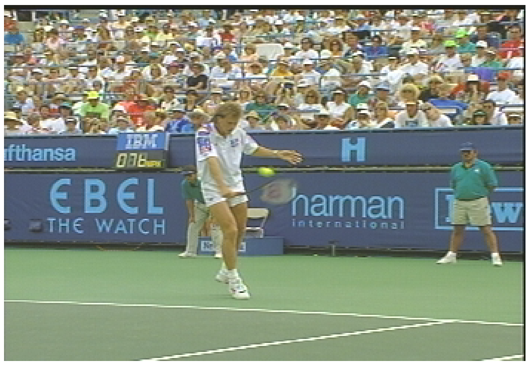


(a) Frame 10 (b) Segmentation result

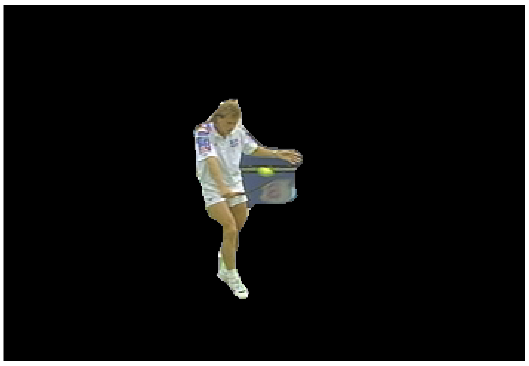
(c) Frame 50 (d) Segmentation result

(e) Frame 90 (f) Segmentation result

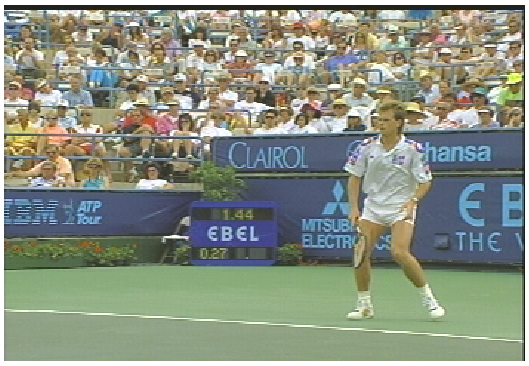
Figure 3.17: Example segmentation results, sequence “Race1 (View 0)”



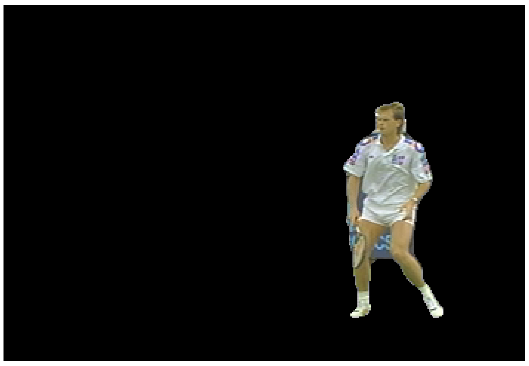
(a) Frame 20



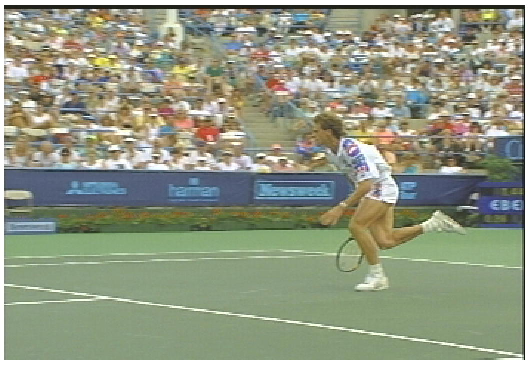
(b) Segmentation result



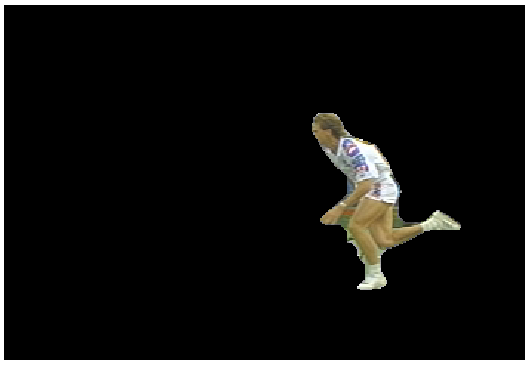
(c) Frame 130



(d) Segmentation result

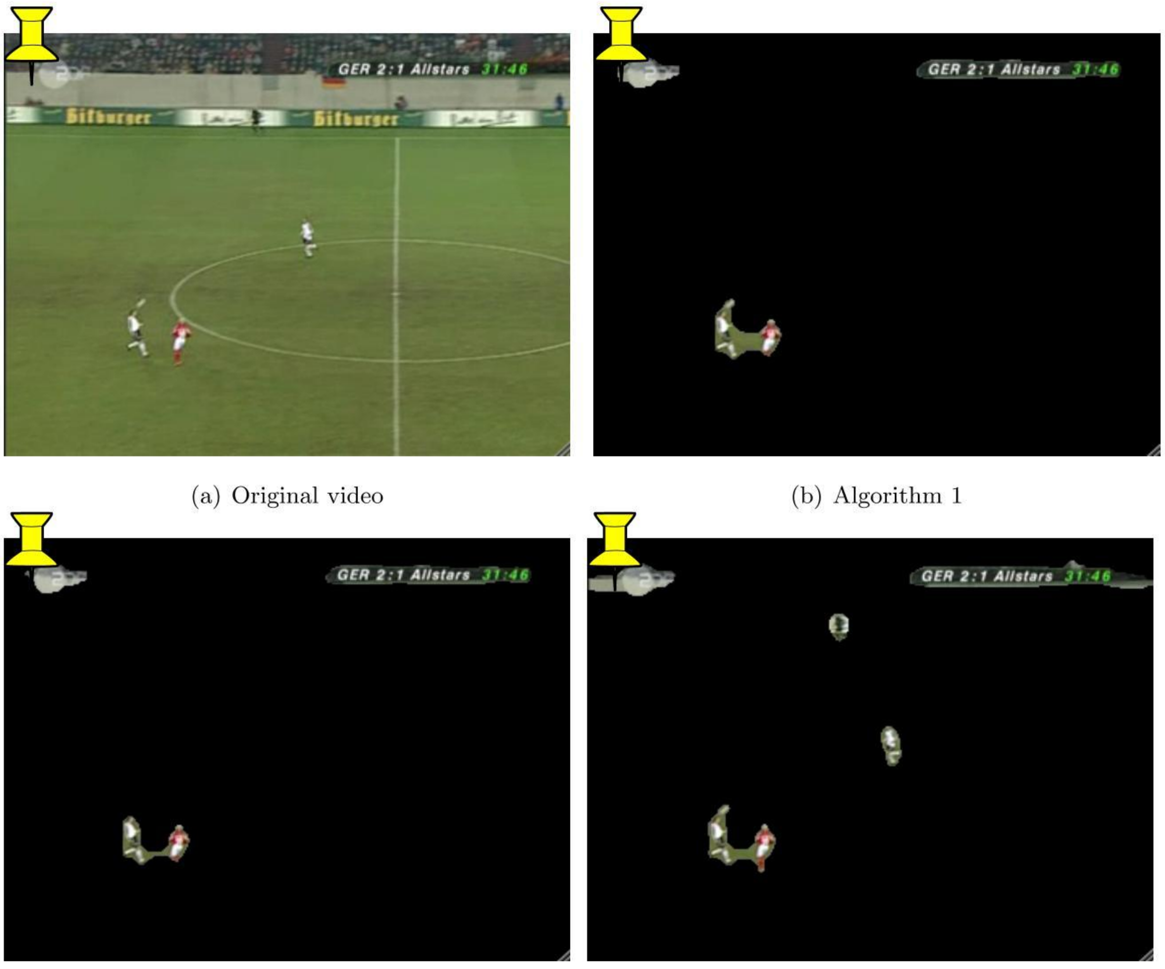


(e) Frame 250



(f) Segmentation result

Figure 3.18: Example segmentation results, sequence "Stefan"



(a) Original video

(b) Algorithm 1

(c) Algorithm 2

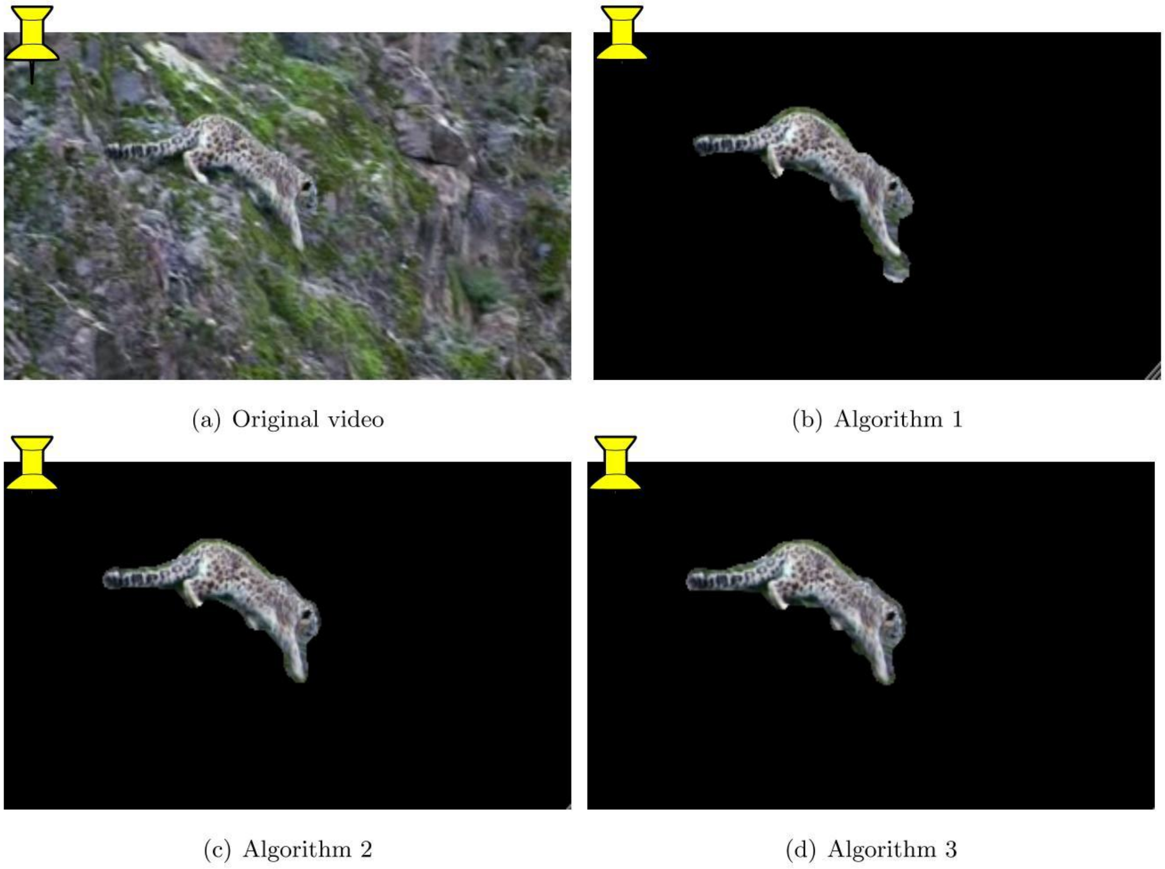
(d) Algorithm 3

Figure 3.19: Subjective comparison of various automatic object segmentation algorithms "Allstars" (Click on the full images while using *Adobe Reader*)

3.8 Chapter summary and outlook

In this chapter a new automatic object segmentation algorithm based on local background Sprites introduced in the previous chapter has been proposed and evaluated. The segmentation procedure has been portioned into three parts, the breaking criterion of the local background Sprite generation, the use of different color spaces, and the thresholding method. For each part, various methods have been taken into account and the behavior of each combination has been evaluated regarding the performance of object segmentation in video sequences with a moving camera.

Additionally, the best combination has built the new proposed object segmentation method and has been compared with reference algorithms using global motion estimation only and GME/Global background Sprites for the pre-processing step. In both reference algorithms only the luminance component of the video data was considered (YUV400 color space) and the algorithm proposed by Krutz et al. [36] was used for the thresholding step.



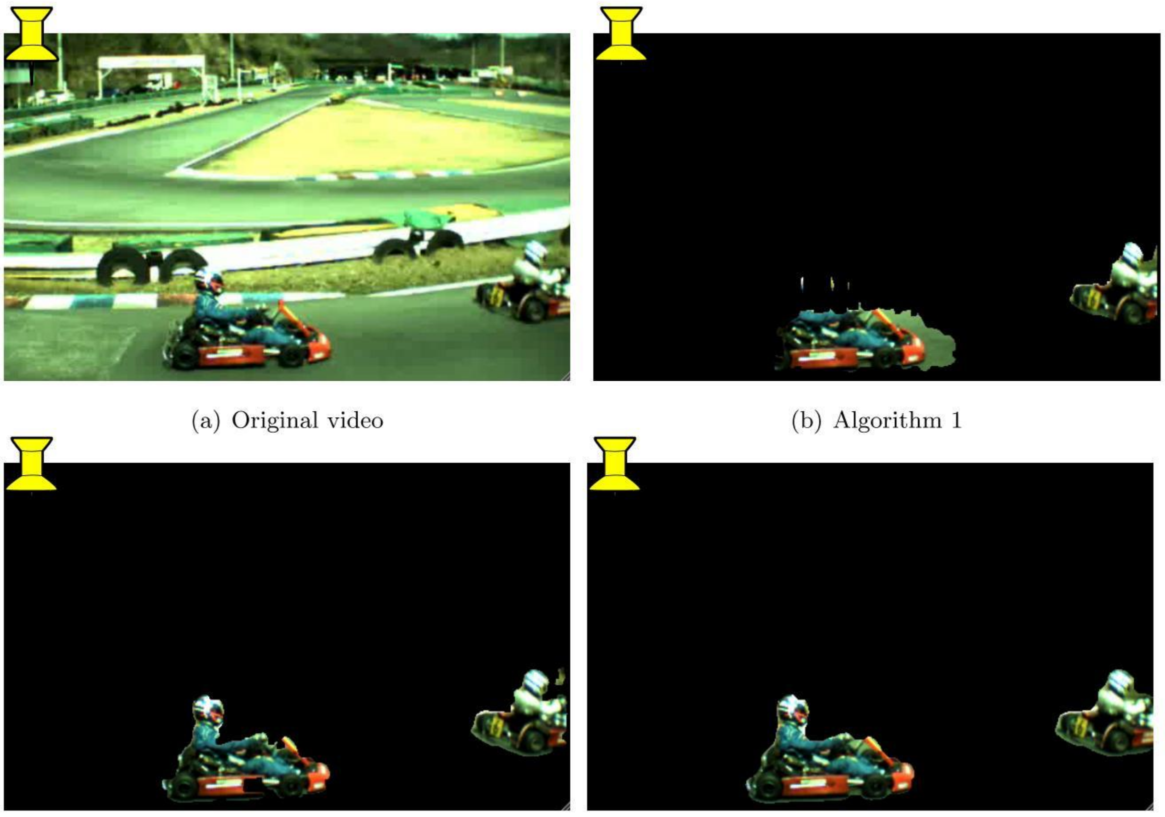
(a) Original video (b) Algorithm 1 (c) Algorithm 2 (d) Algorithm 3

Figure 3.20: Subjective comparison of various automatic object segmentation algorithms “Mountain” (Click on the full images while using *Adobe Reader*)

Comprehensive experiments have been conducted to find the best configuration. The outcome is that, regarding the generation of the local background as a pre-processing for object segmentation, the “variance” and the “maximum” criterion have shown the best performance. The method using the “eigenvalues” has led to worse object segmentation results. The use of all components of the YUV color space has brought an overall significant improvement. Finally, we have evaluated various thresholding methods and selected the three most known algorithms [23], [84], [57] and compared their performance inside the proposed algorithm to our own thresholding method already proposed in [36].

It should be mentioned that the choice of the thresholding method should be kept as simple as possible is this work. Therefore, we haven’t considered more sophisticated thresholding methods, like algorithms using fuzzy-logic or supervised algorithms. Our main application is content-adaptive video coding and the object segmentation method is placed in a pre-processing part at the encoder and thus should work fast and online.

The evaluation shows that the method proposed in [36] outperforms all considered competitive algorithms for all test sequences. This means that we have found already a very good configuration regarding the thresholding aspect with the considered test sequences. We will show later on that, with some considered test sequences, we have found nearly the optimum object segmentation.



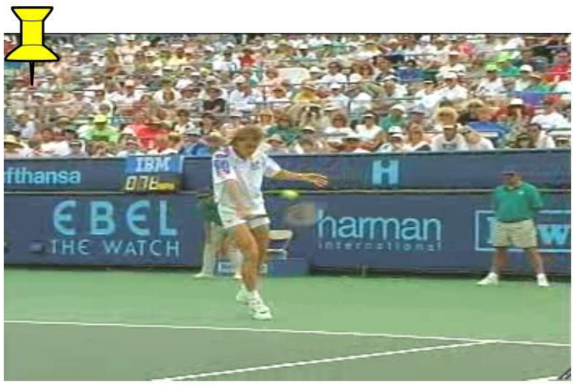
(a) Original video (b) Algorithm 1

(c) Algorithm 2 (d) Algorithm 3

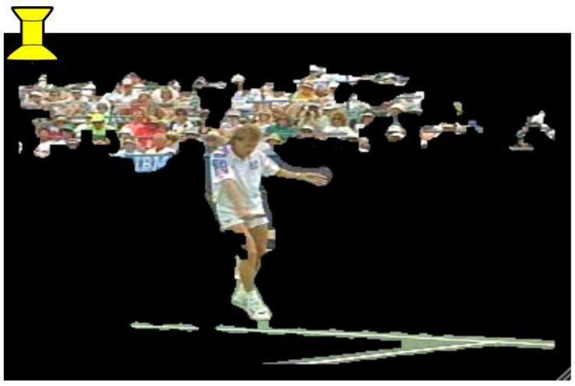
Figure 3.21: Subjective comparison of various automatic object segmentation algorithms "Race 1" (Click on the full images while using *Adobe Reader*)

So it comes out that the biggest challenge regarding the automatic object segmentation is to find the best algorithm for generating the background model, in our case the local background Sprites. Here, several aspects can be considered and still be optimized, e.g. using an arbitrary window size and/or finding a content-adaptive breaking criterion. Furthermore, we have only considered test sequences with continous camera motion. Algorithms have to be found when test sequences are taken into account with changing in the camera motion, e.g. moving-still-moving. Tracking algorithms can be used here and/or a rough pre-characterization of the camera motion to adapt the following object segmentation algorithm.

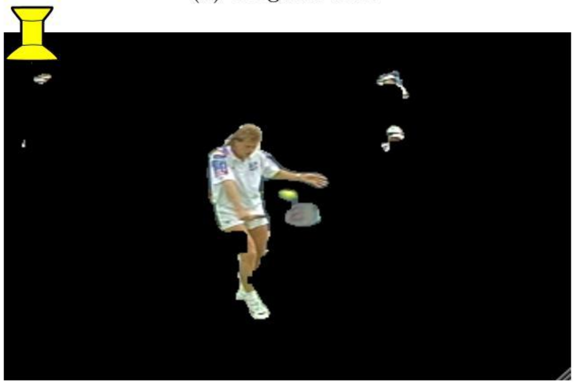
This chapter finalizes the basic pre-processing steps of a content-adaptive video coding approach. First, the global motion of an input sequence is analyzed and second the background during a considered test sequence is separated from the foreground objects. We would like to emphasize here that one goal of this work is, comparing the previous approaches, to set-up an in-built fully-automatic object segmentation method using the already found global motion parameters. We can say that a very good segmentation approach has been found inspired by previous work and newly developed methods. The next chapters of this thesis will show the performance of this pre-processing algorithms inside content-adaptive video codecs.



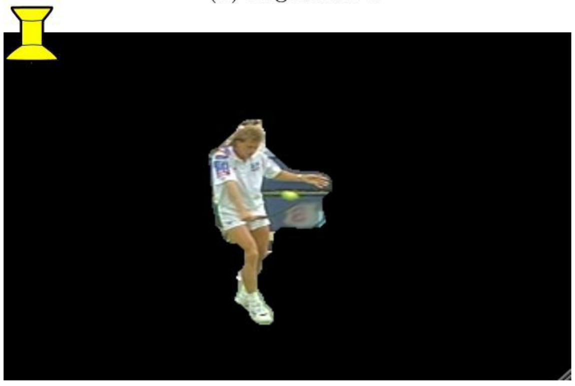
(a) Original video



(b) Algorithm 1



(c) Algorithm 2



(d) Algorithm 3

Figure 3.22: Subjective comparison of various automatic object segmentation algorithms "Stefan" (Click on the full images while using *Adobe Reader*)

Chapter 4

Automatic Sprite Coding for Single- and Multiview

Now we can set up an automatic Sprite coding scheme using the pre-processing techniques developed in the previous chapters. The main challenge of the Sprite coding idea has been the foreground/background segmentation of the video sequence to be coded. During the MPEG-4 standardization, it was assumed that the segmentation was performed beforehand. In this work, we design an in-built automatic object segmentation in a Sprite coding environment. We use the MPEG-4 Sprite coding technique as well as a new model-based video coding scheme based on the Sprite coding idea with use of H.264/AVC.

4.1 Introduction

Sprite coding has been established almost a decade ago. It was developed within the MPEG-4 activities dealing with object-based video coding approaches. The general idea is to segment the input sequence into foreground and background objectes. The background object is generated by aligning all images of a certain number of frames to one larger sized image. This image is called background Sprite. The aligned images are blended together such that all moving foreground objects are removed. On the other hand, there is the foreground objects sequence. Both foreground and background objects are then coded separately. At the decoder, the background frames are reconstructed from the Sprite and merged with the foreground objects to build the orignial video sequence. Figure 4.1 shows a simplified Sprite coding scheme. In [14] Sprite coding within MPEG-4 is described in more detail. The motivation for pursuing this approach has been outlined in [67]. Here, the Sprite coding idea is taken into account to design a model-based coding scheme with automatic in-built object segmentation.

Over the last years the whole processing chain of this approach has been researched comprehensively. The two main pre-processing techniques are global motion estimation and Sprite generation. Very early techniques in this field can be found in [20], [50], [7], and [5].

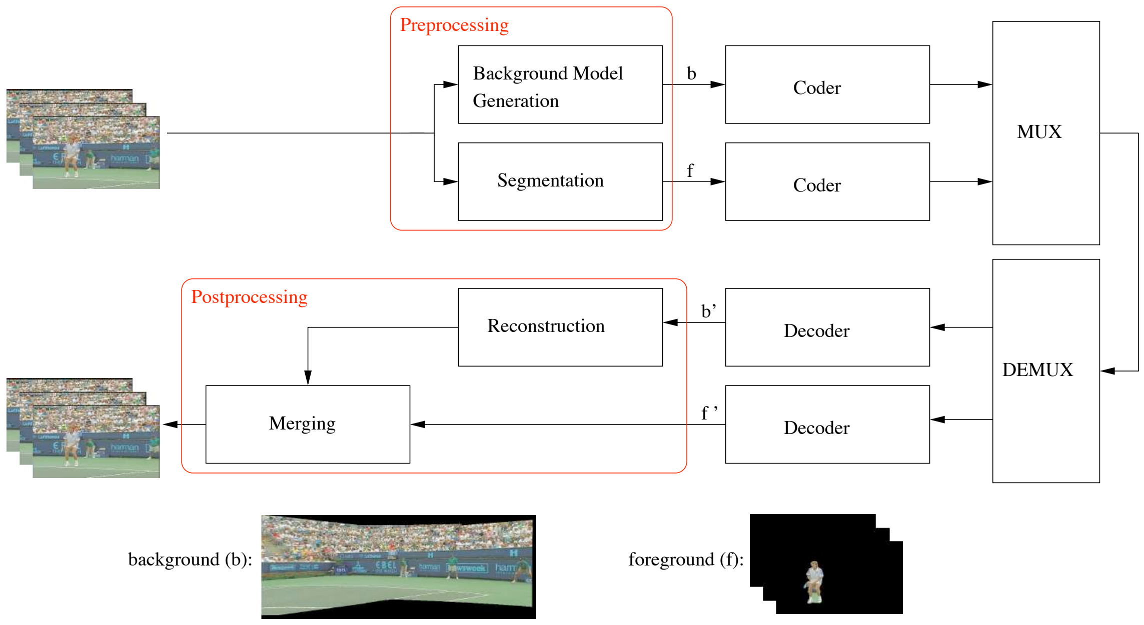


Figure 4.1: Simplified Sprite coding scheme

Further work on improved global motion estimation and Sprite generation algorithms have been published in [70], [8], [22], [46], [4], [24], [18], and [11].

The goal of this work is to build a complete video coding system based on these earlier works. This video coding system based on the Sprite coding idea is designed towards more general usability. Having the enhanced GME-algorithm of Chapter 2 and the improved object segmentation algorithm of Chapter 3, it is possible to tackle the important object segmentation issue. Towards a more practical Sprite codec, it would be very useful if this part can be performed "in-built". It has been shown in [48] and [12] that it is possible to use the already calculated pre-processing techniques, GME and Sprite generation, for an automatic segmentation. Based on these works, improved automatic object segmentation algorithms have been proposed in [36], [30]. In [33], these methods are evaluated within a real video coding environment. The pre-processing with in-built object segmentation produces the Sprite-based data representation. For coding, the MPEG-4 Visual reference codec has been used. As a result, the automatic object segmentation algorithms perform very good in comparison to groundtruth masks for the considered test sequences.

Thus, we are able to perform an "in-built" object segmentation using global motion estimation and the background Sprites already generated. Considering the latest standardized hybrid video codec, H.264/AVC [90], our goal is to find a way to improve the coding efficiency of this codec using the Sprite-based representation. For that, we apply our enhanced pre-processing techniques including the "in-built" object segmentation first. Having the segmented video data, we use the reference H.264/AVC encoder to code all the segments of the video and multiplex the streams together to one bitstream. At the decoder, the bitstream is demultiplexed, decoded and the segments are then merged together to the original video. It

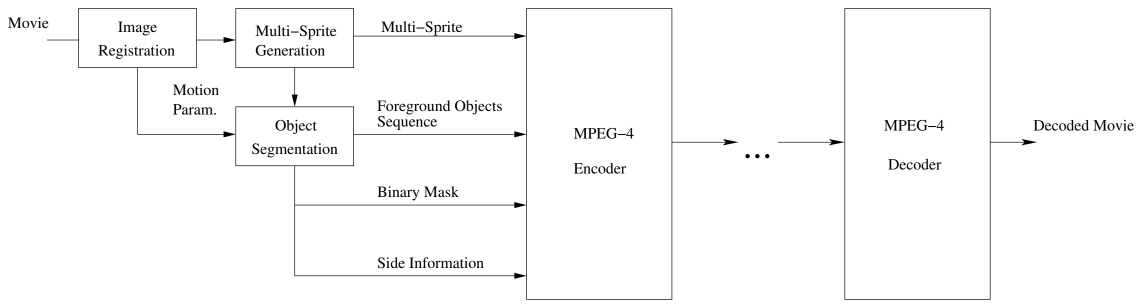


Figure 4.2: MPEG-4 Sprite coding using in-built object segmentation

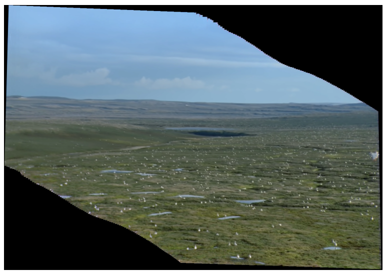
Sequence	Source	Resolution	Frames	FPS [Hz]
Allstars	ZDF (German TV (Channel 2))	352 × 288	250	25
BBC-Pan12	BBC (Docu. *Planet Earth*)	720 × 576	185	25
BBC-Pan13	BBC (Docu. *Planet Earth*)	720 × 576	110	25
Biathlon	ARD (German TV (Channel 1))	352 × 288	200	25
Mountain	BBC (Docu. *Planet Earth*)	352 × 192	100	25
Stefan	MPEG	352 × 240	300	30

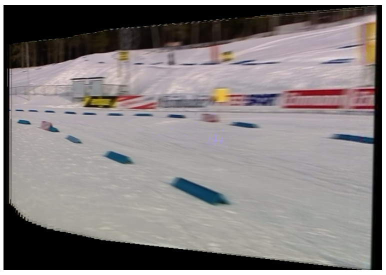
Table 4.1: Test sequences used for comparison of MPEG-4 Visual Sprite coding using various automatic foreground object segmentation approaches.

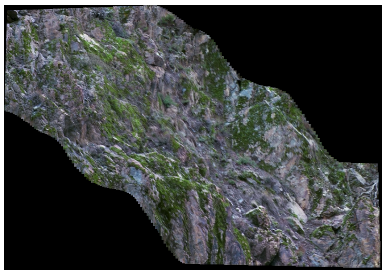
has been shown in [26] and [41] and [39], where the design of this approach was developed in collaboration, that this approach significantly improves the common use of H.264/AVC for single- and multi-view. In this chapter, we would like to describe this enhanced Sprite-based video codec using H.264/AVC in more detail. Comprehensive experiments show the very good performance of our improved model-based coding scheme.

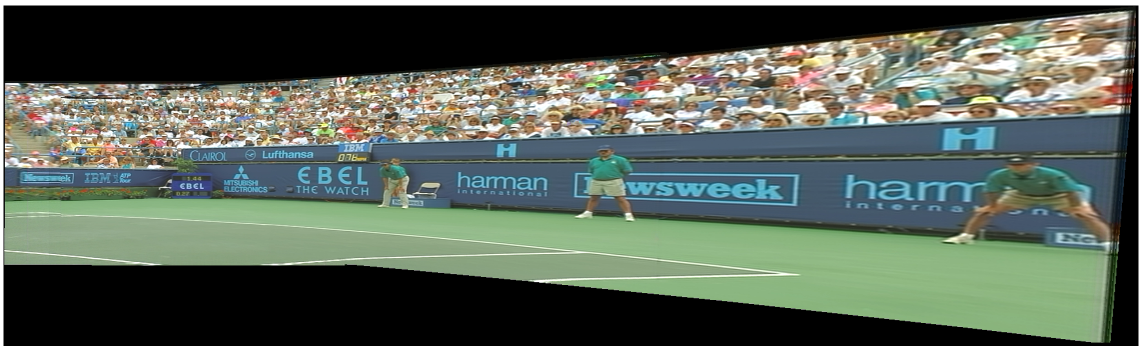
4.2 Automatic Sprite Coding within MPEG-4

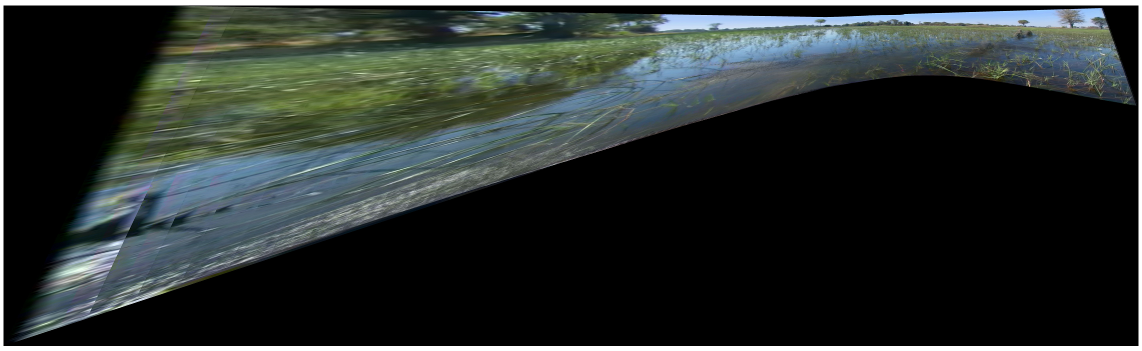
The Sprite coding part in MPEG-4 Part 2 Visual was developed for an object-based representation of the input video using background Sprites. Our automatic segmentation algorithms are mainly developed for this coding purpose. Therefore, we would like to evaluate these algorithms in a real object-based coding environment. For that, we use our pre-processing algorithms and connect them to the MPEG-4 codec. This is visualized in Fig. 4.2. We use the global motion estimation and Sprite generation techniques developed in Chapter 2 and the object segmentation algorithms from Chapter 3. Thus, the MPEG-4 codec is fed with the foreground objects sequence, the foreground object mask obtained automatically using our algorithms, the background Sprite image and the motion parameters. The output is a coded file and an already decoded and reconstructed version of the input video.


(a) Single Sprite for “BBC-Pan13”


(b) Multiple Sprite 3/4 for “Biathlon”


(c) Single Sprite for “Mountain”


(d) Multiple Sprite 1/3 for “Stefan”


(e) Single Sprite for “BBC-Pan12”

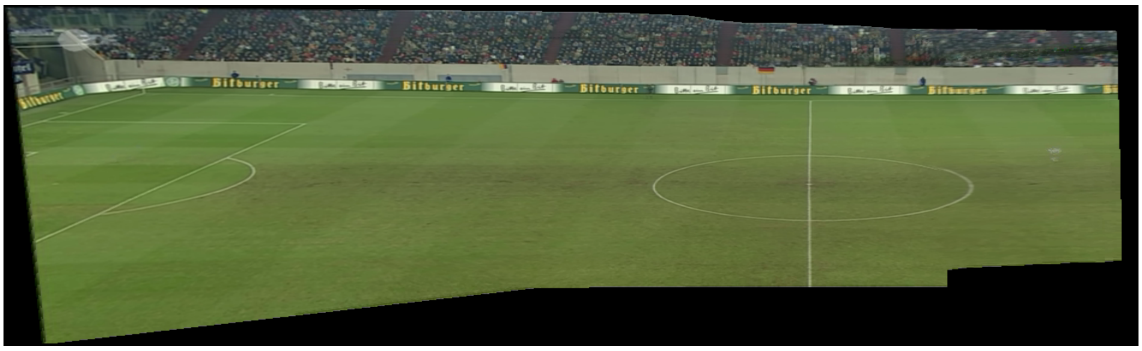
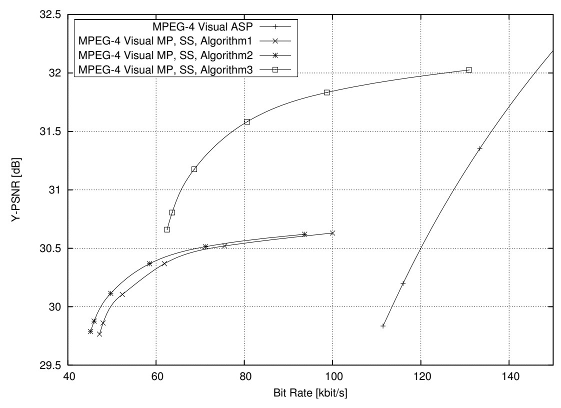
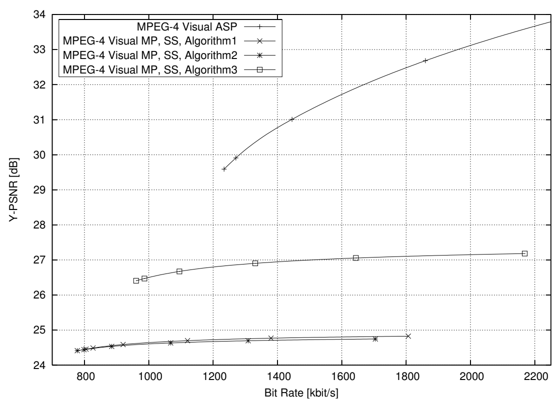

(f) Single Sprite for “Allstars”

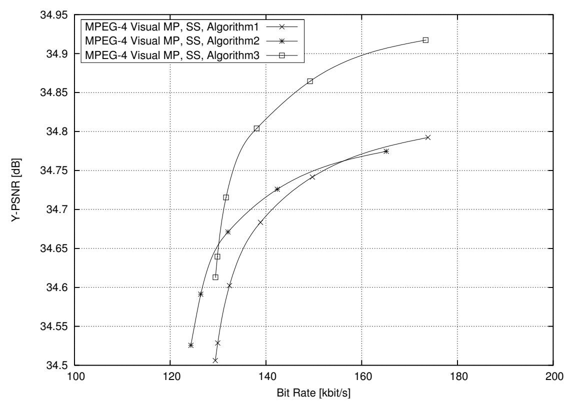
Figure 4.3: Examples of background Sprite images generated using the approaches from Chapter 2.



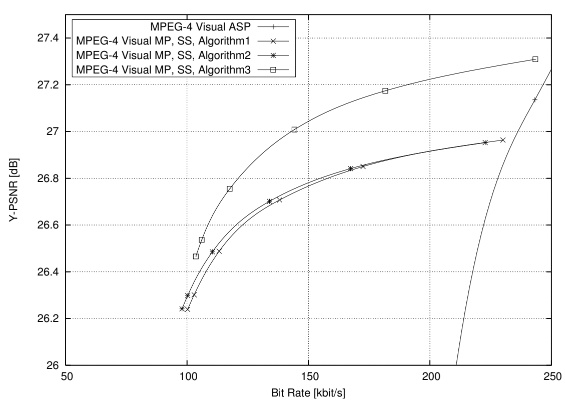
(a) Sequence *Allstars*



(b) Sequence *BBC-Pan12*



(c) Sequence *BBC-Pan13*



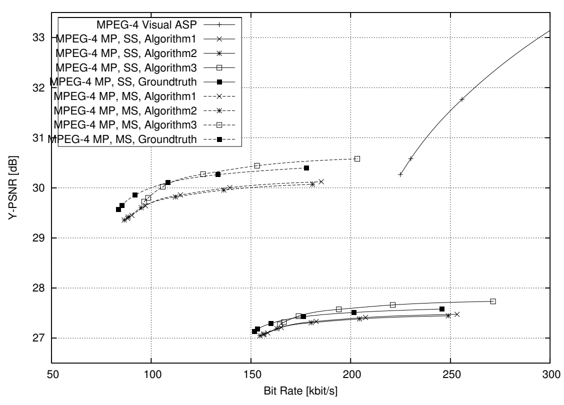
(d) Sequence *Mountain*

Figure 4.4: Comparison of rate-distortion performance for coding using MPEG-4 Visual ASP with Sprite coding (MP) applying different types of segmentation algorithms.

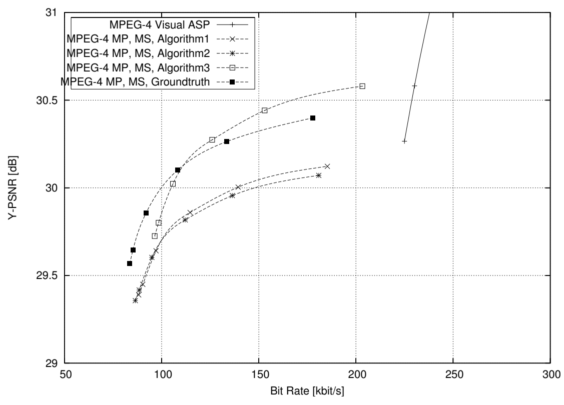
4.2.1 Intrinsic evaluation of automatic object segmentation algorithms

For the experimental evaluation, we considered six test sequences that are listed in Table 4.1. First, background Sprite images have been generated for all test sequences using the approaches presented in Chapter 2. However, multiple Sprites have only been generated for "Biathlon" and "Stefan", since the camera pan in all other sequences is too narrow to make this necessary. In the multiple Sprites case, the sequence "Biathlon" is divided into four partitions (frames 0 to 9, 10 to 22, 23 to 46, and 47 to 199) and "Stefan" is divided into three partitions (frames 0 to 244, 245 to 261, and 262 to 299). Examples of background Sprite images are depicted in Fig. 4.3.

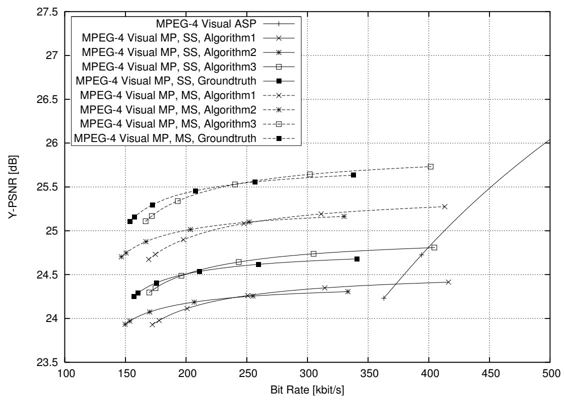
Segmentation has been performed using all test sequences from Table 4.1 and all given algorithms described in Chapter 3, i.e. *Algorithm1* based on short-term global motion compensation, *Algorithm2* based on background subtraction using single and multiple Sprites, and *Algorithm3* based on background subtraction using local background Sprite modeling. Additionally, for the sequences "Biathlon" and "Stefan", groundtruth masks for the foreground objects were available. Therefore, we also used these as an ideal segmentation case for Sprite



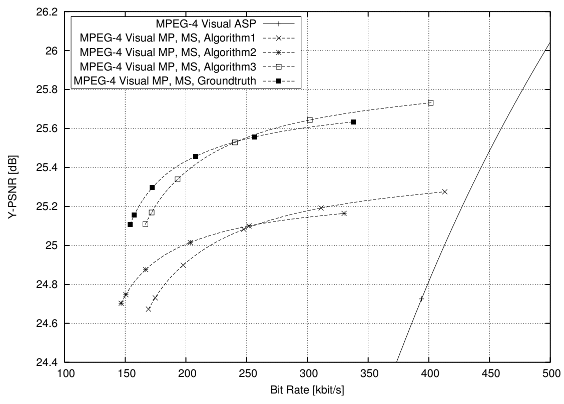
(a) Sequence *Biathlon*



(b) Sequence *Biathlon* (zoom in)



(c) Sequence *Stefan*



(d) Sequence *Stefan* (zoom in)

Figure 4.5: Comparison of rate-distortion performance for coding using MPEG-4 Visual ASP with Sprite coding (MP) applying different types of background Sprites and segmentation algorithms.

coding to evaluate the influence of correct segmentation in terms of coding performance. The sequences were coded using the Sprite coding approach presented in the previous section. The MPEG-4 Visual reference coder software MoMuSys (Mobile Multimedia Systems) has been used applying the Main Profile (MP) for Sprite coding. Additionally, all sequences were coded as one rectangular video object using the Advanced Simple Profile (ASP) for comparison. The quantization parameter (QP) has been kept constant ($\mathrm{QP}_{\mathrm{bg}} = 14$) for the background model for all test sequences. The foreground object sequences have been coded using one of several quantization parameters, i.e. $\mathrm{QP}_{\mathrm{fg}} \in \{7, 10, 14, 21, 28, 31\}$, as is the case for the ASP. It has to be stated explicitly that the choice of $\mathrm{QP}_{\mathrm{bg}}$ is purely random, i.e. no optimization in terms of setting the best combination of $\mathrm{QP}_{\mathrm{bg}}$ and $\mathrm{QP}_{\mathrm{fg}}$ has been done. The prediction structure has been set to IPPP with a GOP size of 16 for Sprite coding as well as for the ASP to ensure comparability. Quarter-pel motion vector accuracy has as well been enabled for both profiles.

Fig. 4.4 and 4.5 show rate-distortion results for all test sequences used. It can be seen in all curves that the segmentation approach *Algorithm3*, i.e. segmentation based on background subtraction using local background Sprites, outperforms all other segmentation techniques. Additionally, when multiple background Sprites are used (cf. Fig. 4.5(a) and Fig. 4.5(c)), the performance is better than compared to using one single background Sprite image as a model.

For the sequence “BBC-Pan12” (cf. 4.4(b)), the Sprite coding approach performs worse than coding the sequence using MPEG-4 ASP. This can be explained with the content of the sequence. It shows a group of monkeys wading through water, which is moving and takes a large part of the video frame. Dynamic textures, e.g. water, have to be assumed as foreground objects in a video, since their movement differs from the global motion. Such a dynamic texture cannot be modeled correctly by a static background Sprite. However, all segmentation approaches presented in this work define the water as background. Therefore, the decoded video frames differ strongly from the original sequence in these parts after reconstruction of background Sprite image and foreground object sequence. This results in a lower PSNR compared to MPEG-4 ASP, which correctly reconstructs the dynamic texture.

In Fig. 4.4(c), the rate-distortion curves for the test sequence “BBC-Pan13” are shown. Here, no curve for coding the sequence with the MPEG-4 ASP is depicted, since the bit rate needed for coding this sequence using the ASP is in the range of about 800 kbit/s to 1200 kbit/s, depending on the coarseness of quantization. The reason for the low bit rate range when using Sprite coding compared to the ASP is the content of the sequence. The foreground objects, i.e. a group of flying birds, are very small. Therefore, the background model already shows nearly the complete sequence. In other words, the bit rate needed for coding the foreground object sequence is only 1-2 times the bit rate needed for coding the background model. E.g. for the sequence “Allstars”, the bit rate for the foreground object sequence is 3-11 times higher than that needed for the background model, depending on the combination of quantization and segmentation approach. Therefore, the “BBC-Pan13” sequence is an excellent example of the capabilities of the Sprite coding principle.

The usage of a single background Sprite image for the sequence “Biathlon” (cf. 4.5(a)) did not perform as well as expected. It turned out that the generation of the background model introduced some errors. This can also be explained with the content of the sequence. Fig. 4.3(b) shows one of its multiple Sprites, which gives an impression of its content. Most part of it is snow, i.e. homogeneous content. In background Sprite generation, long-term motion compensation is performed. This means that a mapping of pixel content between temporally remote frames has to be done. This is challenging when large homogeneous areas are present and may lead to bad results, as is the case here. Additionally, the camera moves very fast, which makes background Sprite image generation even harder. Due to these problems, the quality of the background model for this sequence is not very good, which leads to a bad rate-distortion performance.

For *Algorithm3*, up to 1.4 dB PSNR-gain has been achieved compared to the competing segmentation algorithms. For the sequences where groundtruth is available, *Algorithm3* even outperforms the groundtruth-mask in the upper bit rate range. We thus can say that we have found a nearly optimal automatic object segmentation method for this kind of coding approach.

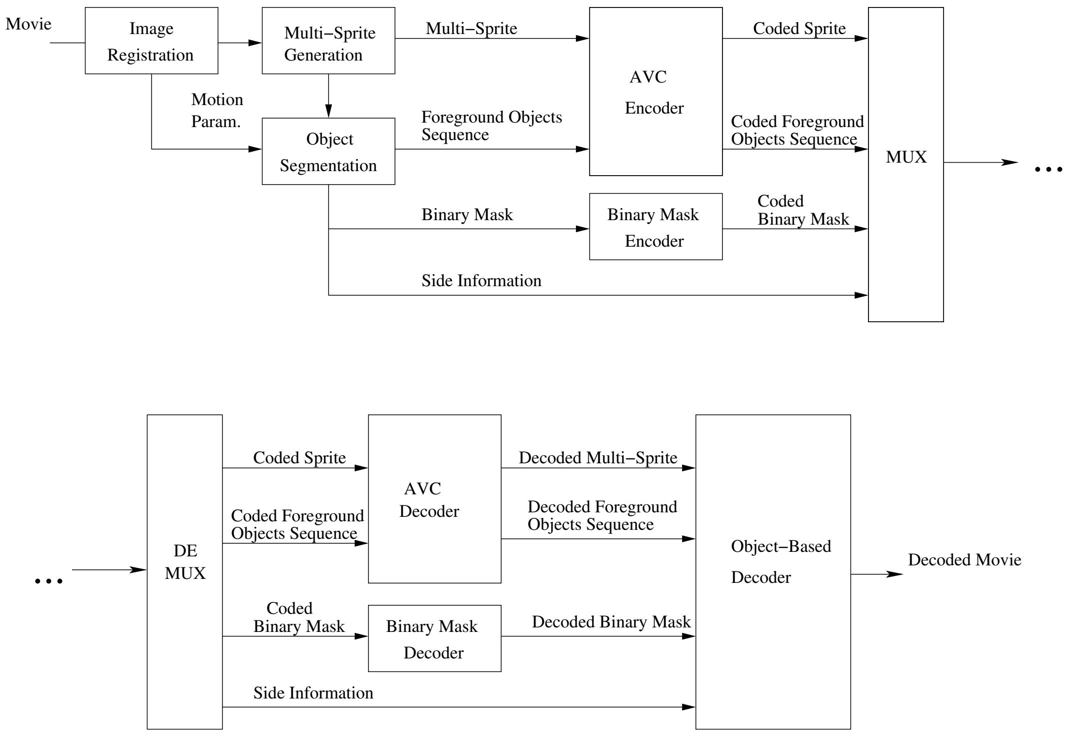


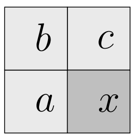
Figure 4.6: Automatic Sprite coding system using H.264/AVC

4.3 Automatic Sprite Coding using H.264/AVC

The basic idea of this coding approach is to transfer the input video sequence into foreground and background parts before encoding. A block chart of the Automatic Sprite coding approach using H.264/AVC is given in Fig. 4.6. The H.264/AVC encoder is treated as a black box. The coding process of each of the data segments is described next in more detail.

4.3.1 Texture and Binary Mask Coding

The textures of the Sprite and the foreground objects are independently coded using H.264/AVC. We used the reference software JSVM version 9.1. To efficiently code the foreground, the objects are expanded to fit to macroblock structure. We applied coding in a hierarchical B picture scheme IbBb... with a GOP length of 15 frames [62]. The Sprite itself is coded as I picture. The binary mask coding scheme uses the binary arithmetic coder “M-Coder”, which is specified in the H.264/AVC standard. Eight contexts are initialized to model 8 possible spatial states for every pixel to code as shown in Fig. 4.7. In advance to the coding, the mask is divided into 16x16 blocks. The transformation parameters are coded as four frame corner point correspondences using 3 bytes per corner point and build the side information.



$$\longrightarrow \quad ctx(x) = a \cdot 2^0 + b \cdot 2^1 + c \cdot 2^2$$

Figure 4.7: Context assignment for $bit(x)$ in Binary Mask Coding

4.3.2 Applying blocks on the segmented foreground objects

Having the segmented foreground objects sequence, we would like to use the common H.264/AVC standard for coding. Because of the block-based hybrid video coding approach, it is useful to lay a 16x16 block grid over a foreground objects frame. Figure 4.8 shows two examples. Block grids have two advantages. Firstly, due to the automatic segmentation, errors can occur in foreground objects. We apply the blocks in the way that every block which contains pixels from the foreground is declared as foreground. This means that falsely segmented regions of the foreground objects appear again. Secondly, this approach prevents losing coding efficiency because a block is either background and contains only zeros or it is foreground. We have no blocks which contain both, background (zeros) and foreground pixels. This would lead to additonal cost in bit rate because of the high-frequency edge. A disadvantage of this approach is that falsely segmented background regions (background is segmented as foreground) increases and the coding performance of the OBVC decreases. Therefore, it is still very important to have a very accurate segmentation algorithm as a pre-processing step.

4.3.3 Experimental comparison of OBVC against common use of H.264/AVC

We evaluate the overall coding performance of the automatic Sprite codec called “OBVC”. For that, we use a test data set of 8 video sequences of different genres, characteristics and resolution. The test data set is depicted in Tab. 4.2.

We only compare the OBVC with H.264/AVC because it is well-known that the advanced coding tools of H.264/AVC significantly outperform the MPEG-4 Part 2 Visual. We use H.264/AVC inside the OBVC and we would like to show if it is possible that we outperform the common hybrid encoder using our object-based representation including the automatic object segmentation. For the experiments, the latest developed and evaluated object segmentation is used. As already mentioned, we consider an objective quality measurement,

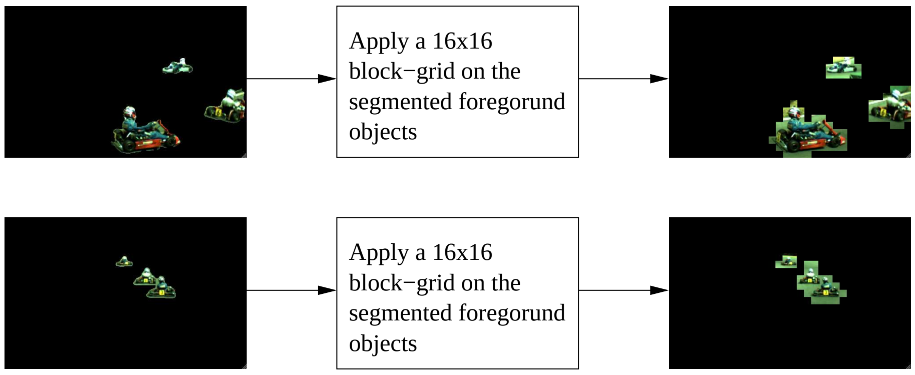
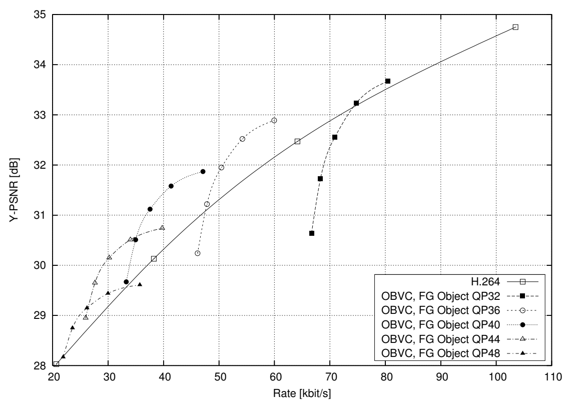

Apply a 16x16 block−grid on the segmented foregorund objects
Apply a 16x16 block−grid on the segmented foregorund objects

Figure 4.8: Generation of blocked foreground objects

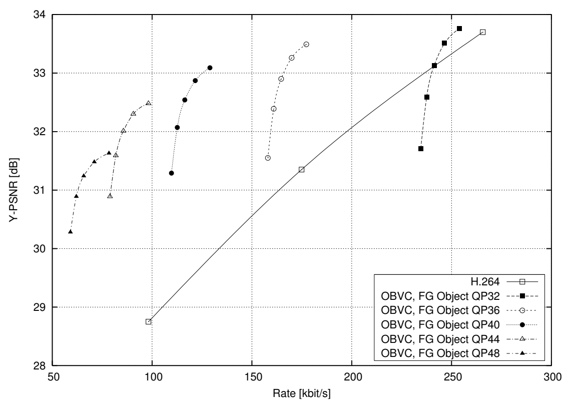
Table 4.2: Test video sequences

Sequence	Source	Resolution	Frames	FPS [Hz]
“Allstars” (cif)	ZDF (German TV (Channel 2))	352×288	250	25
“Allstars”	ZDF (German TV (Channel 2))	704×576	250	25
“Biathlon”	ARD (German TV (Channel 1))	352×288	200	25
“Entertainment”	VCEG	352×288	250	25
“Mountain”	BBC (Docu. *Planet Earth*)	352×192	100	25
“Race1”	MPEG	544×336	100	25
“Stefan”	MPEG	352×240	300	25
“Tempete”	VCEG	352×288	260	25

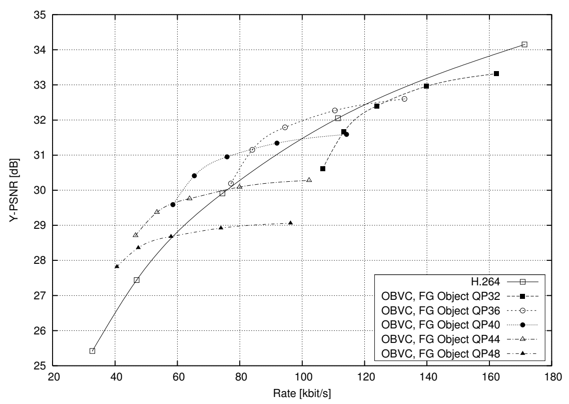
the very common peak signal-to-noise ratio (PSNR). For the H.264/AVC codec, we use the reference software JSVM v.9.1. The encoder settings are hierarchical B-frames for the prediction and the GOP-size is 15 frames. For the calculation of the bit rate, we assume a frame rate of 25 frames/s. To set up a fair comparison, we keep these settings fix for encoding the whole video sequence as well as encoding the foreground objects sequence in the OBVC. We take all eight test sequences into acount for this evaluation. The rate-distortion curves for all sequences are shown in Fig. 4.9-4.10. It can be seen that the OBVC obtains higher coding gain in comparison to common use of H.264/AVC for all sequences. We achieve bit rate savings up to 50 %. It is also obvious that the coding efficiency of our coding approach increases with the resolution of the test sequences. This means that the OBVC is able to encode higher resolution of certain video sequences at lower bit rates, which makes it suitable e.g. for internet applications.



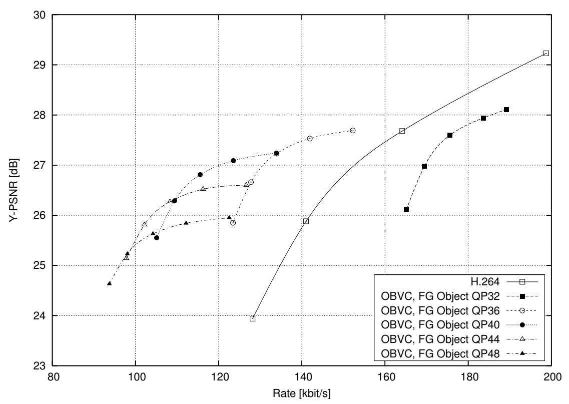
(a) "Allstars" (cif)



(b) "Allstars"



(c) "Biathlon"



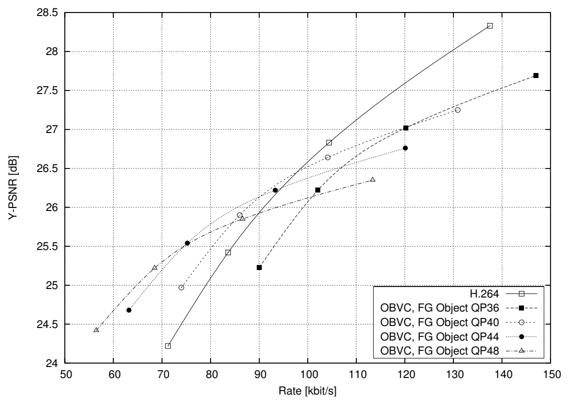
(d) "Entertainment"

Figure 4.9: Rate-distortion curves comparing OBVC and H.264/AVC (1)

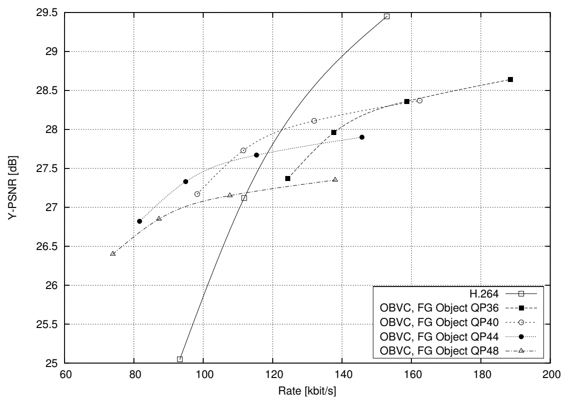
4.4 Extension to multi-view video: Automatic Sprite Coding using MVC

The model-based coding approach described and evaluated in the previous sections can be simply extended to the multi-view case. For that, a set of n views from the multi-view video data has been handled separately in the pre-processing, i.e. the global motion estimation, Sprite generation and automatic object segmentation. A simplified schematic of the proposed automatic Sprite coder using MVC is shown in Fig. 4.11. Figure 4.13 shows the coding structure for the multi-view background Sprites. The Sprites for each view and the associated foreground objects sequences are coded using a multi-view prediction scheme shown in Fig. 4.12. Here, an *IbBb...* frame structure is used with $GOP = 15$. Although, Fig. 4.12 shows the scheme for only two-views, it can be extended to more views easily.

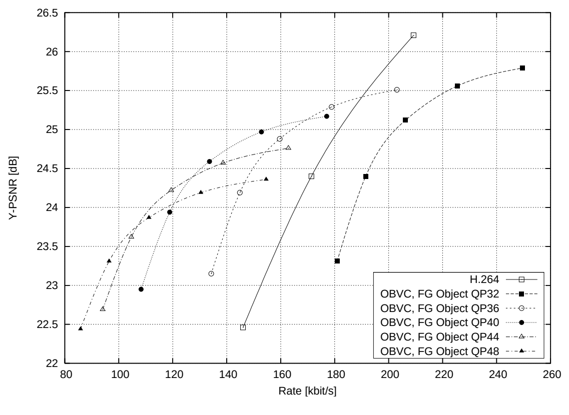
For the first experimental results, we now add the second view of the "Race1" sequence to evaluate our approach for two views. The same coding settings as for the single-view case were applied using the MVC. Figure 4.14 (a) shows the coding results. It can be seen that our codec performs better than the common use of the MVC. The Sprite sequence contains



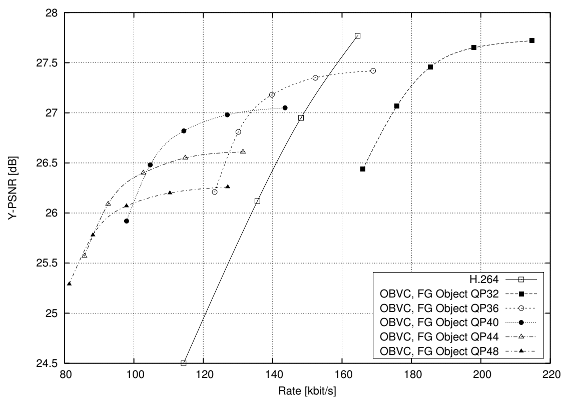
(a) "Mountain"



(b) "Race1"



(c) "Stefan"



(d) "Tempete"

Figure 4.10: Rate-distortion curves comparing OBVC and H.264/AVC (2)

two frames of Sprites, which are coded predictively and yield a higher coding gain. Thus, we have a better performance in the higher quality curve in comparison to the single-view case. To evaluate the subjective improvement of the coded video content, two examples are shown below. Figures 4.15 (a) and (b) shows the comparison of the sequence "Race1 view 1" coded with the object-based coder and MVC. The frames show the difference in quality between the background objects reconstructed using two coding approaches. It can be observed that the lower the bit rate, the higher are the artifacts in the MVC. The background, reconstructed from the Sprites, is almost free of these errors. This result illustrates one of the key advantages of the Sprite coding approach. Furthermore, Fig. 4.15 (c)-(e) show a zoom-in of an example frame, which shows the foreground object. It can be seen that even the object sequence is reconstructed with a better quality than using the MVC. Thus, the object based approach significantly outperforms common hybrid codecs for the considered test sequences in both objective and subjective performance assessment.

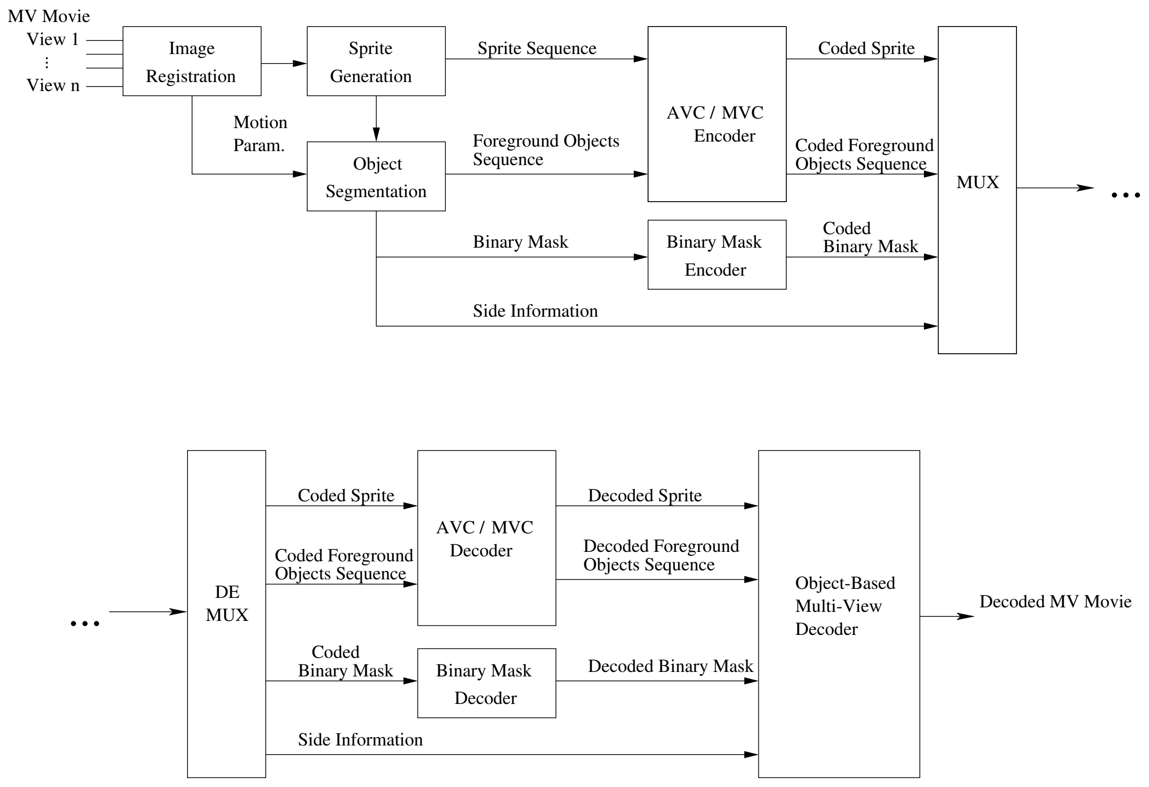


Figure 4.11: Object-based Multi-View Video Codec (OBMVC)

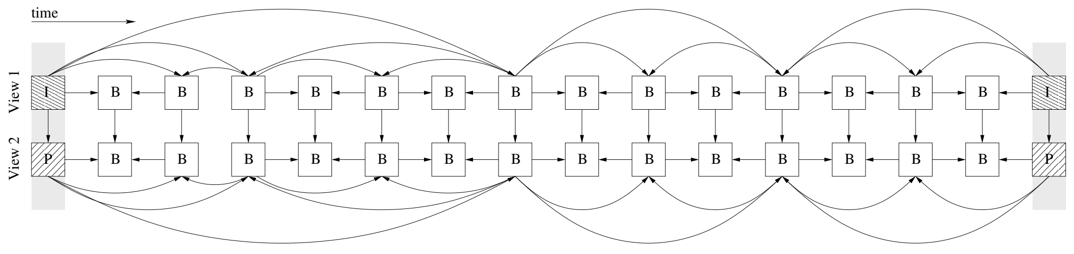


Figure 4.12: Multi-View Prediction Scheme for the foreground object sequence ($GOP = 15$)

4.4.1 Multiple Multi-view Sprite Prediction

Now we consider the case when more than only one background Sprite containing all the background information of the sequence is used. The advantage is, as described above, that if there is a wide camera pan during the sequence a single background Sprite becomes also very large. Applying a multiple Sprite results in smaller background Sprites, which reduces the bit rate. Furthermore, multiple Sprites provide more accurately reconstructed background frames because of smaller distortions at the border of the Sprite. Additionally, we will show that multiple background Sprites are also advantageous if more than one direction of a camera pan appears during the sequence considered (e.g. “Race1”). Using single Sprites, we approached in the previous section the single background Sprites of each view as a video sequence. Having multiple Sprites, we now can use common prediction schemes inlcuded in MVC to code the background Sprites over the views and along the single sequence.

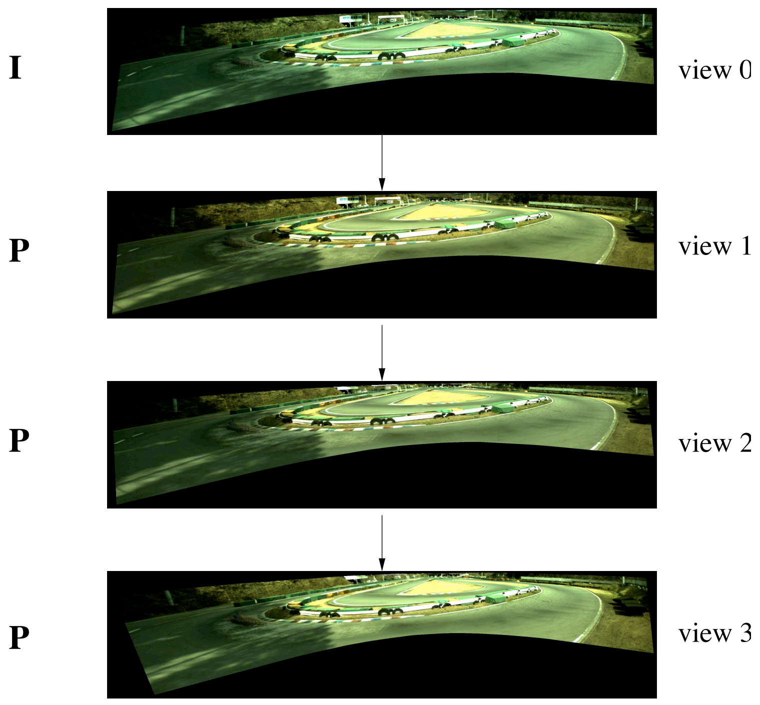


Figure 4.13: Coding structure of a Multi-view background Sprite sequence, “Race1”

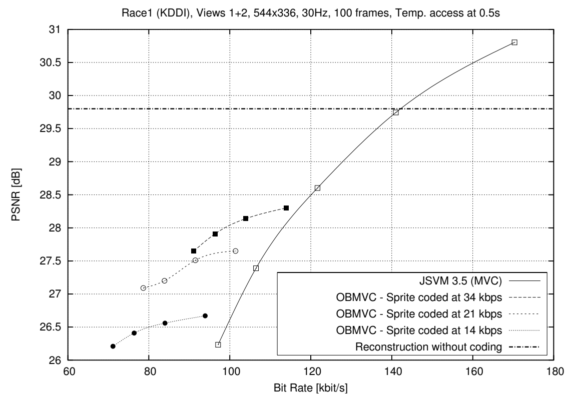
The coding performance was examined of the single- and multiple Sprite generation methods. For multiple Sprites, we used the generation technique using the estimation of the real pan angle and camera motion characterization (CMC) introduced in Chapter 2. We coded the different background Sprites and produced a rate-distortion comparison. In Fig. 4.14 (b) the curves are drawn for the multi-view case. As expected from the direct PSNR-value comparison we achieve much higher coding performance using the new multiple Sprite generation technique based on CMC. Overall, we can state that using multiple multi-view Sprite prediction improves the coding behavior compared to the technique, which uses a single multi-view Sprite prediction.

4.4.2 Outlook towards the enhanced OBMVC

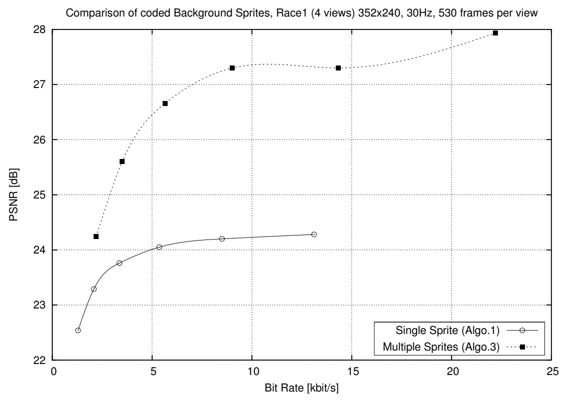
Having the mutiple multi-view Sprite prediction scheme we can outlook towards an enhanced OBMVC. Figure 4.16 shows the block diagram using the new features described above. We can say that a significant coding improvement will be expected in comparison to the object-based MVC described in this work. An experimental evaluation of that is the first future task when pursuing in this area.

4.5 Chapter summary and outlook

We have presented an object-based video coding system using background Sprites. The main difference, compared to previous work in this field, is the automatic in-built object segme-



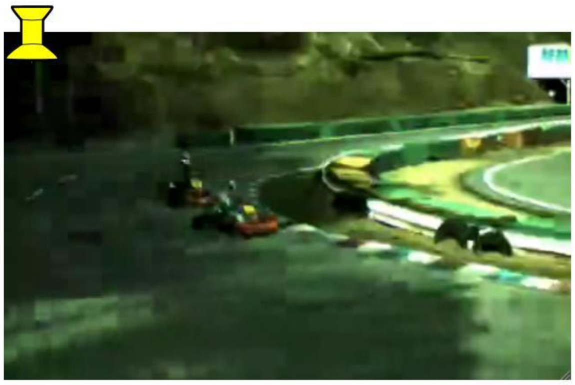
(a) Coding results for multi-view video (average per view)

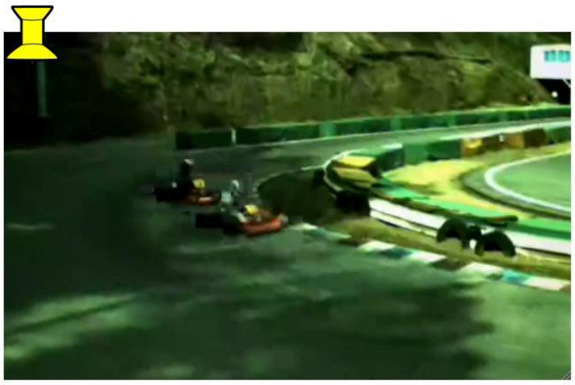


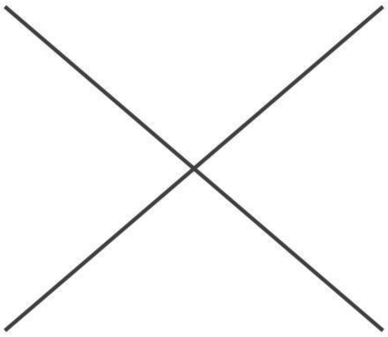
(b) Rate-distortion curve of the considered Sprite generation methods, "Race1", 4 views, 530 frames per view

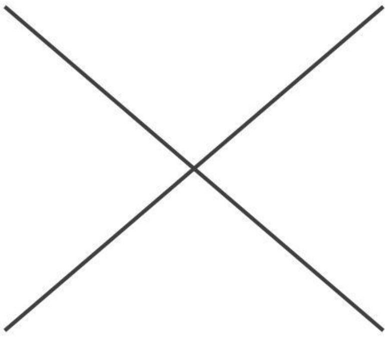
Figure 4.14: OBMVC coding and Multiple multi-view Sprite prediction results

nation and encoding the object-based represented video data with H.264/AVC. Beside this, we have shown the performance of new advanced automatic object segmentation algorithms in sequences with higher-order camera motion including fast pan and strong zoom. We evaluated these algorithms in a real coding environment using the MPEG-4 Visual Main Profile encoder. It has been shown that the performance of the automatic object segmentation algorithms is very close or equal to the manually segmented ground truth object representation with the considered test sequences. Furthermore, we have used the latest developed object segmentation algorithm and an enhanced multiple background Sprite generation algorithm as pre-processing for the automatic Sprite codec using H.264/AVC (OBVC). We compared the OBVC with H.264/AVC using a number of different test sequences and it can be seen that for a certain bit rate range, the OBVC outperforms H.264/AVC up to 50 %. It is also obvious that the encoder of the OBVC has several new settings, e.g. the choice of the QP of the background Sprite and the QP of the foreground object sequence. A next step would be finding an optimal setting of these parameters in a rate-distortion sense. Due to a possible different behavior of this coding approach, it is necessary to design a new optimization method, which is the biggest task and will be discussed in the next chapter. Further, we have extended our model-based coding approach to the multi-view case. It has been shown that having a set of views, where background Sprites are suitable, siginificant coding gain can be achieved due to the enhanced prediction using the Sprites connected with the MVC. It has also been indicated that using enhanced Sprite generation techniques, such as multiple Sprites, further enhanced prediction schemes are possible to bring even more coding performance. The task for future work will be further investigation how this enhanced long-term prediction method can increase the coding efficiency of a multi-view video either with a Sprite representation or other new approaches.


(a) MVC (98 kbit/s)


(b) OBMVC (97 kbit/s)


(c) original


(d) MVC (98 kbit/s)

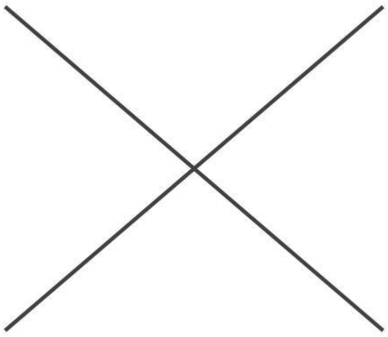
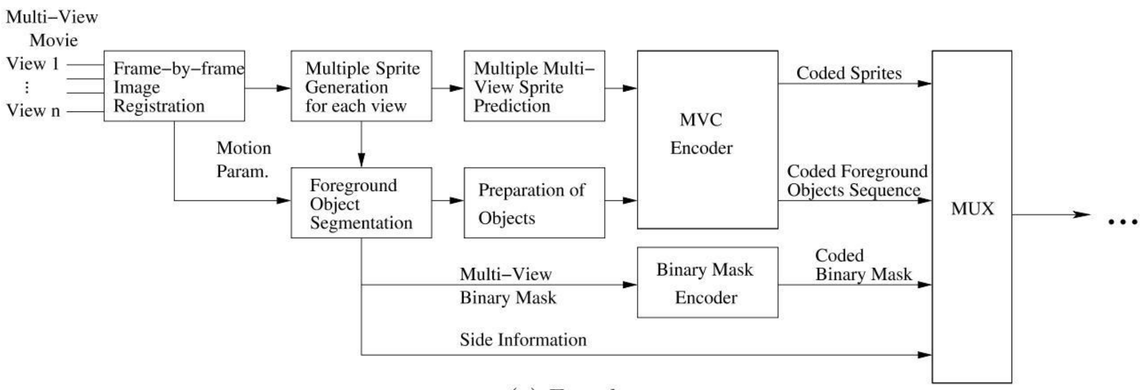

(e) OBMVC (97 kbit/s)

Figure 4.15: Subjective comparison of decoded test video “Race1 view 1” (Click on the full images while using *Adobe Reader*)


Multi−View Movie
View 1
⋮
View n
Frame−by−frame Image Registration
Multiple Sprite Generation for each view
Multiple Multi−View Sprite Prediction
MVC Encoder
Coded Sprites
Motion Param.
Foreground Object Segmentation
Preparation of Objects
Coded Foreground Objects Sequence
MUX
...
Multi−View Binary Mask
Binary Mask Encoder
Coded Binary Mask
Side Information
(a) Encoder

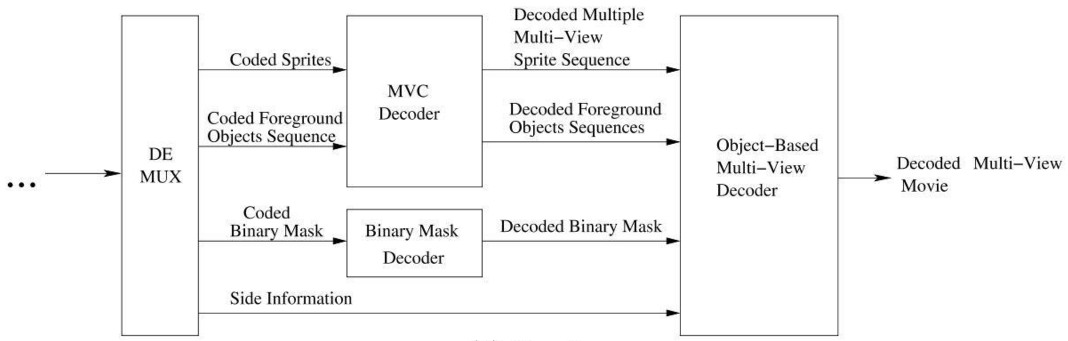

...
DE MUX
Coded Sprites
Coded Foreground Objects Sequence
Coded Binary Mask
Side Information
MVC Decoder
Binary Mask Decoder
Decoded Multiple Multi−View Sprite Sequence
Decoded Foreground Objects Sequences
Decoded Binary Mask
Object−Based Multi−View Decoder
Decoded Multi−View Movie
(b) Decoder

Figure 4.16: Object-based Multi-View Video Codec using Multiple Sprites

Chapter 5

Rate-Distortion optimized Automatic Sprite Coding

5.1 Introduction

As already mentioned in the previous chapter, Sprite coding was developed within the MPEG-4 activities dealing with object-based video coding approaches. In [14] Sprite coding within MPEG-4 is described in more detail. The motivation for pursuing this approach has been outlined in [67] and in this work. Automatic Sprite coding has been introduced recently as an extension of the latest video coding standard H.264/AVC [26], [41], [35]. It has been shown that for a certain kind of video sequences, it is possible to outperform the hybrid video coding approach up to 50% in bit rate savings. In our approach, we consider the H.264/AVC-encoder as a black box. The video content is pre-processed in a way that all the background information of a number of consecutive frames of the input video is aligned into one single image, which is called background Sprite. The remaining foreground objects are segmented automatically and stored in a separate video file. For this, background subtraction algorithms are applied where the already generate background Sprite is used as a background model for all frames of the sequence considered [36], [31]. This is the main novelty in comparison to Sprite coding in MPEG-4 where the segmentation mask of the objects were treated as given. We code the different objects, the foreground object sequence and the background Sprite image, using H.264/AVC independently.

At the decoder, the background sequence is reconstructed from the decoded background Sprite image and the original video sequence is generated by merging the background frames sequence and the foreground object sequence. As mentioned above, both parts are coded separatly using H.264/AVC. This means also that each of them are encoded with the optimized coding parameters set by the in-built rate-distortion optimization of the encoder. Our goal now is to examine the optimization of the whole automatic Sprite codec (OBVC) introduced in the previous chapter. The behavior of the OBVC is different in comparison to common hybid codecs because of several issues, e.g. coding the background pixels inside a background Sprite but measure the quality of reconstructed background frames and varying the in-built object segmentation parameters. We discuss how these parameters of the OBVC can be set optimally in a rate-distortion sense. There is a number of work done here concerning rate control of object-based coding. However, all this related work refers to the MPEG-4 object-based approach and the use of background Sprites were not considered [51], [56]. Our approach is closer to the H.264/AVC-encoder. Therefore, we consider the optimization techniques used for hybrid video codecs.

A lot of work has been done in rate-distortion optimization of video codecs. For a hybrid video encoder, the state-of-the-art approach is an optimization using the Lagrangian cost function. The principle is to minimize the function $J(I|X,\lambda)$:

$$J(I|X,\lambda) = D(I|X) + \lambda R(I|X), \tag{5.1}$$

where D is the distortion of the input signal after encoding and decoding and R is the allocated bit rate. The variable I stands for the input signal (row of images), X includes the encoding parameters and λ is the Lagrangian multiplier. It has been shown that the prediction modes including the costs of motion estimation are optimzed, both at macroblock level. A block chart of a hybrid video codec including rate control at the encoder is shown in Fig. 5.1.

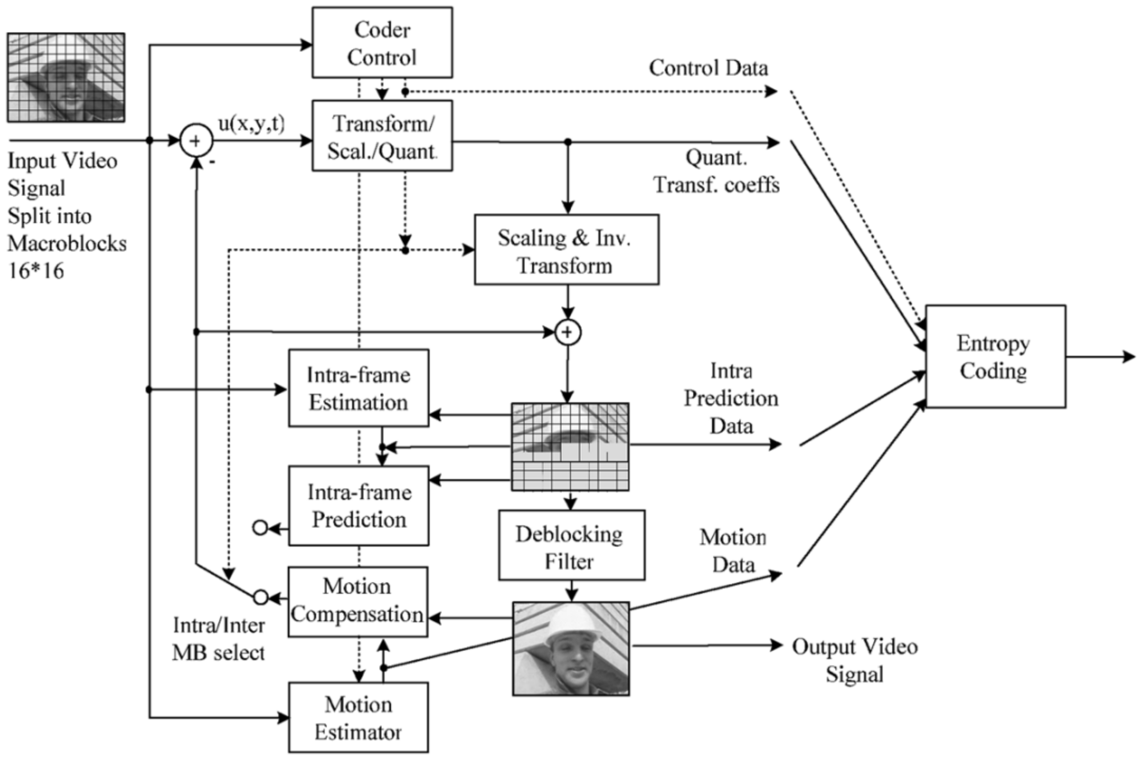


Figure 5.1: Hybrid Video Coder including rate-distortion optimization [77]

A comprehensive description and evaluation has been published in [76] and [86]. Using this method, a critical point is to assign the Lagrangian parameter λ. In [76] a relationship between λ and the quantization parameter QP was found experimentally:

$$\lambda = 0.85 \cdot \mathrm{QP}^2 \tag{5.2}$$

which was further examined in [85]. A theoretical approximation, which lead to this relationship has been also shown [76] and [85]. Furthermore, Equation 5.2 is valid for H.263 and MPEG-4. The same experiment was examined for the latest hybrid video coding standard, H.264/AVC. Here, the relationship between λ and QP is [86]:

$$\lambda = 0.85 \cdot 2^{\frac{QP-12}{3}} \tag{5.3}$$

Based on these investigations, we take into account the issue of encoder optimization using the Lagrangian approach for the OBVC introduced in the previous chapter. As already mentioned, the OBVC encoder has different behavior in comparison to a common hybrid video encoder. Therefore, we have to re-design the optimization process for our purpose. This chapter is organized as follows. In Section 5.2 the Lagrangian rate-distorion optimization approach is described in more detail. Especially, assigning the Lagrangian multiplier is discussed. Section 5.3 includes the optimization method applied to our coding approach. Experimental results are shown in Section 5.4 and the last section gives an outlook to further steps, which can be investigated in this issue.

5.2 Rate-Distortion Optimization for Hybrid Video Coding

The Lagrangian cost function for optimal allocation of bits was firstly introduced in [64]. Applying this technique for video coding was shown in e.g. [75], [89], [76], and [86]. Until today, it has become "state-of-the-art" for optimizing encoder parameter settings in hybrid video encoders, such as the optimal choice of prediction modes and motion estimation. A brief insight is provided into the technique to set up the next section.

For choosing the optimal mode for each macroblock, a Lagrangian cost function J_{mode} is minimized [86]:

$$J_{\mathrm{mode}}(\mathbf{S}_k, I_k|Q, \lambda_{\mathrm{mode}}) = D_{\mathrm{REC}}(\mathbf{S}_k, I_k|Q) + \lambda_{\mathrm{mode}} R_{\mathrm{REC}}(\mathbf{S}_k, I_k|Q), \tag{5.4}$$

where $\mathbf{S}_k$ are the macroblocks, I_k are the coding modes, Q is the quantization paramter, D_{REC} is the distortion, R_{REC} is the rate after entropy encoding, and λ_{mode} is the Lagrangian multiplier. For the distortion measurement, the sum of squared difference (SSD) is used. Refering to [86], possible coding modes for different video coding standards are e.g.:

- MPEG-2 : INTRA, SKIP, INTER-16x16
- H.263 and MPEG-4 : INTRA, SKIP, INTER-16x16, INTER-8x8

- H.264/AVC : INTRA-4x4, INTRA-16x16, SKIP, INTER-16x16, INTER-16x8, INTER-8x16, INTER-8x8

The same approach is also applied to motion estimation. Here, another Lagrangian cost function is examined regarding to motion estimation type (pixel accuracy, half-pixel accuracy, quarter-pixel accuracy). The critical point of optimizing a constrained problem using the Langrangian cost function is the right choice of assigning the multiplier λ. In [76] and [85], a theoretical derivation has been given of a relationship between λ and the quantization parameter Q. The first derivative of the Lagrangian function

$$J(R) = D + \lambda R, \tag{5.5}$$

has to be calculated and set to zero to find the optimal λ:

$$\frac{dJ(R)}{dR} = \frac{dD}{dR} + \lambda = 0, \tag{5.6}$$

which results in the well-known equation:

$$\lambda = -\frac{dD}{dR}. \tag{5.7}$$

This means that the optimal Lagrangian multiplier λ is the slope of the distortion-rate function. Now, a typical high-rate approximation function for entropy-contrained scalar quantization can be assumend as:

$$R(D) = a \log_2 \left(\frac{b}{D} \right). \tag{5.8}$$

The parameters a and b define the relationship between rate and distortion. For the distortion-quantizer function, it is assumed that at sufficiently high rates, the source probability distribution can be approximated as uniform within each quantizer interval. The function can be written as:

$$D(Q) = \frac{Q^2}{3}. \tag{5.9}$$

Having this distortion-quantizer function, a rate-quatizer function can be assembled by inserting Equ. 5.8 in 5.9:

$$R(Q) = a \log_2 \left(\frac{3b}{Q^2} \right). \tag{5.10}$$

There are two functions of distortion D and rate R, which are a function of the quantizer Q. We have the relation between λ, D, and R, so we calculate the derivatives of both functions:

$$\frac{dD(Q)}{dR(Q)} = \frac{\left(\frac{Q^2}{3}\right)'}{\left(a\log_2\left(\frac{3b}{Q^2}\right)\right)'} = \frac{\frac{2Q}{3}}{\frac{-2a}{Q\ln(2)}} = \frac{\ln(2)}{3a}Q^2, \tag{5.11}$$

with

$$\left(a\log_2\left(\frac{3b}{Q^2}\right)\right)' = a\left(\frac{3b}{Q^2}\right)' \cdot \left(\log_2\left(\frac{3b}{Q^2}\right)\right)' = \left(-\frac{6b}{Q^3}\right) \cdot \left(\frac{Q^2}{\ln(2)3b}\right) = \frac{-2a}{Q\ln(2)}. \tag{5.12}$$

To this end, we have the relationship between the Lagrangian multiplier and the quantizer value:

$$\lambda = -\frac{dD}{dR} = \frac{\ln(2)}{3a}Q^2, \tag{5.13}$$

where $\frac{\ln(2)}{3a}$ is set to the parameter c. This parameter was experimentally assigned to 0.85 for H.263, MPEG-4 and H.264/AVC, where for H.264/AVC the relation of λ and the quantizer value is different (shown in Equ. 5.2) due to a different assumption for the distortion-quantizer relation ($D \approx 2^{\frac{Q-3}{12}}$ instead of $D \approx Q^2$ for H.263 and MPEG-4).

This brief overview sets up the next section, where we consider our Sprite-based coding approach for optimizing the encoder.

5.3 Rate-Distortion optimization for OBVC

This section describes the approach for optimizing the object-based video codec using Sprites (OBVC). We start with a general overview to define the problem. Afterwards, the method developed is applied to encode the foreground and background objects in an optimal way. Comprehensive experimental results are conducted to prove this approach, which are shown in the next section.

5.3.1 General approach

We design the rate-distortion optimization for our object-based codec using background Sprites. It has been outlined in the previous section that optimization using the Lagrangian cost function is very suitable for optimizing video encoders with a set of parameters. Our basic coding approach is to merge the impact of background Sprite representation and the powerful encoding tools of the latest video coding standard H.264/AVC. First, we apply enhanced global motion estimation and background Sprite generation techniques as a pre-processing step. We use the results for an in-built automatic object segmentation to extract the foreground objects from the input sequence considered. This is necessary because in the final background Sprite image, only pixels which correspond to the background are stored. We then have the following data representation:

- Background Sprite image (contains all background information of the input sequence)
- Foreground objects sequence (contains only the foreground objects of the input sequence)
- Foreground/Background mask sequence (automatically generated during the pre-processing; needed for the decoder)
- Higher-order long-term motion parameters (side information; control the warping process of images into and out of the background Sprite)

All these items are coded separately. The background Sprite image and the foreground objects sequence are coded using the H.264/AVC encoder. Here, the Sprite image is considered as a video sequence with one frame and coded as an intra frame. It is also possible to code the background Sprite image with other image encoders, e.g. JPEG or JPEG2000. However, we keep encoding with H.264/AVC to be uniform. Furthermore, the motivation of our coding approach is to be an extension of H.264/AVC. The mask sequence is coded by using a binary mask encoder [6]. We do not encode the higher-order motion parameters at this stage. For the bit rate calculation, we reserve 4.8 kbits/s for transmitting these motion parameters (we assume 3 byte per parameter $\rightarrow$ 8 parameter per frame $\rightarrow$ 24 byte/frame with 25 frames/s $\rightarrow$ 4.8 kbit/s.).

After encoding, we multiplex all bit streams together and the final encoded file results. We emphasize that we treat the H.264/AVC-encoders as black boxes. The mean QP-value is the parameter which is set from outside. All others are kept fix. For the prediction scheme, we use hierarchical B-frames to achieve the best coding performance [62].

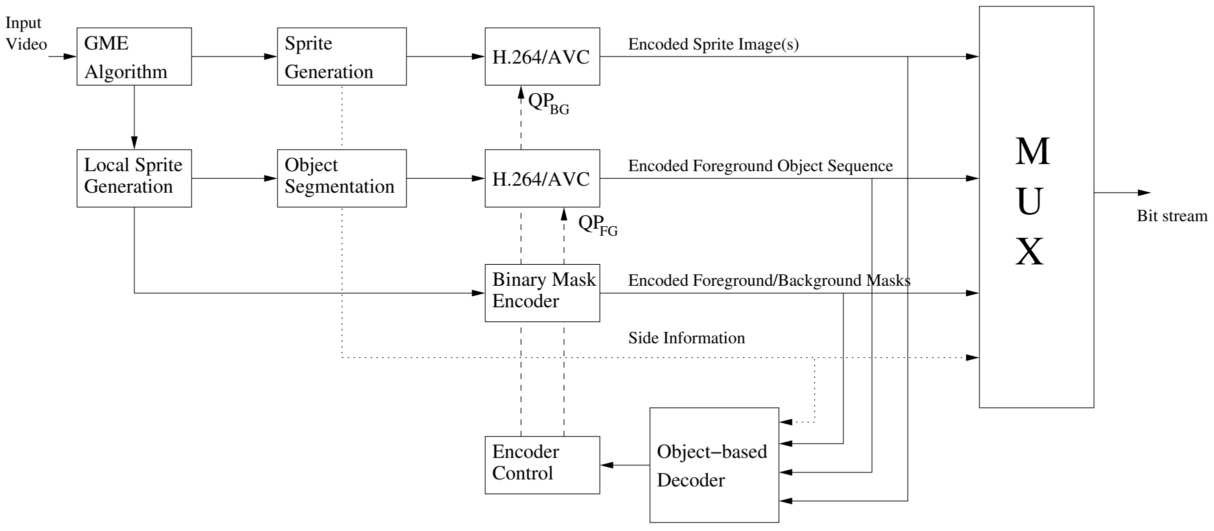


Figure 5.2: Rate-distortion optimized OBVC

Having this set up, we consider our approach as a single video encoder. We have two coding parameters, two QP-values of the foreground objects sequence and the background

Sprite. The question is how the encoder can set these QP-values optimally. Due to the very good performance of the approach using the Lagrangian cost function we also take this into account. The Lagrangian cost function which is to be minimized for our case is:

$$J(I_n|(QP_{fg},QP_{bg}),\lambda_{fg/bg}) = D_{rec}(I_n|(QP_{fg},QP_{bg})) + \lambda_{fg/bg}R_{rec}(I_n|(QP_{fg},QP_{bg})) \quad (5.14)$$

with the following parameters:

I_n - Input image sequence (n frames)

QP_{fg} - QP for foreground object sequence

QP_{bg} - QP for background Sprite image

D_{rec} - Distortion between input images and corresponding reconstructed output images

$\lambda_{fg/bg}$ - Lagrange multiplier

R_{rec} - bit rate of whole encoded image sequence

This means for the encoder that we have to deliver a decoded and reconstructed version of the input image sequence to the encoder control unit. A block chart is given in Fig. 5.2, which depicts the encoder of the controlled object-based approach. The difference here in comparison to previous coder control methods using the Lagrangian approach is that we apply this on the whole images of the video sequence instead of on macroblock level. There exist two questions:

1. Can we compare this coding environment with common well-know hybrid encoding?
2. Can we adopt the theoretical derivations made finding the optimal assignment for the Lagrangian multiplier?

To answer the first question, we have to consider that the foreground object sequence is coded and evaluated directly. However, the process of coding the background information is different. We first generate the background Sprite where all background information is gathered by warping and blending all frames of the considered input sequence into one image. Then, the background Sprite image is coded and afterwards the background image sequence is reconstructed from the decoded background Sprite by another inverse warping process. This means that we transform the background information into another representation, code it and transform it back. The quality evaluation is then measured on the reconstructed frames. One of the theoretical assumptions deriving a formula for assigning the Lagrangian multiplier is the relation between the quantizer value and the distortion of the signal as outlined in the previous section. In case of coding the background information using a Sprite representation, we have no direct relationship between the quantizer of the encoder and the quality of the reconstructed video signal. Figure 5.3 visualizes this issue regarding the quantizer and the input signal. If we would compare S and S' we would have the common coding environment. However, in our case, I and I' is compared to each other.

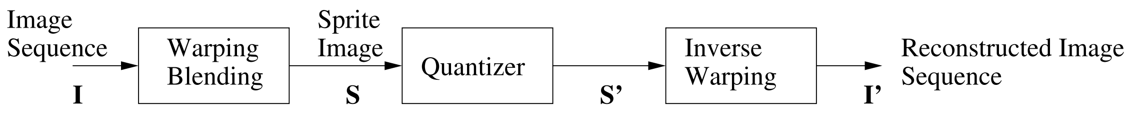


Figure 5.3: Abstracted processing chain for Sprite coding

Therefore, we can say that our coding environment has a different behavior to the common hybrid video coding approach. This means that we can not rely on the theoretical assumptions made towards defining the Lagrangian multiplier. However, due to the similarity, the video signal is kind of filtered before quantization and afterwards, we expect an analog relationship between the Lagrangian multipler and the QP's.

5.3.2 Lagrange Multiplier Selection for optimal Foreground/Background Coding

Beside the assumption of the relation between the distortion and the quantizer (Equ. 5.9 for H.263 and MPEG-4) we can not consider the approximation made for a rate-distortion analytic function (Equ. 5.8) because it is valid for high rates. In our case, we have previously shown [26], [41] that our approach outperforms the common H.264/AVC-codec at low bit rates. In this work, objective measurements are used for performance evaluation and due to the warping, blending and inverse warping process while generating the background Sprite, the uncoded reconstructed video sequence is not equal to the input video sequence which produces a max. PSNR-limit. Thus, we would like to optimize our codec especially in the bit rate ranges where it outperforms common H.264/AVC. This means that we have to derive the Lagrange multiplier selection completely experimentally like the factor 0.85 was found for H.263 and MPEG-4 (Equ. 5.2) and the relation for H.264/AVC (Equ. 5.3).

In [76] and [85] a set of λ's was assigned and the coding behavior as well as the performance were measured to find an appropriate value for the constant c in Equ. 5.13. For our purpose, it has been found that the opposite way is useful.

Considering the computational cost, we define a QP_{fg}-value for the foreground object sequence. Then, the background Sprite is coded with different QP_{bg}'s. Analogous to Equ. 5.4, the differently coded background Sprite images are our "modes". The sequences are reconstructed using the differently coded background Sprites and the values for D_{rec} and R_{rec} are computed. The MSE (mean-squared-error) of each image pair (original and reconstructed) is chosen for the value of D_{rec}. The resulting equation is:

Table 5.1: QP-values chosen for encoding foreground objects and background Sprite

QP_{fg}	24	28	32	36	40	44	48
QP_{bg}	24	28	32	36	40	-	-

$$D_{rec} = \overline{\frac{1}{MN}\sum_{M}\sum_{N}(I_n - I'_n)^2}, \tag{5.15}$$

where I_n is the original frame, I'_n is the reconstructed frame after object-based decoding, n is the number of frames of the input sequence, and $M \times N$ is the size of the images. R_{rec} is the overall rate, which is needed for the whole sequence including the binary mask and motion parameters. For one fix $\mathrm{QP}_{\mathrm{fg}}$, one D_{rec}-R_{rec}-curve is achieved. Figure 5.4 shows an example.

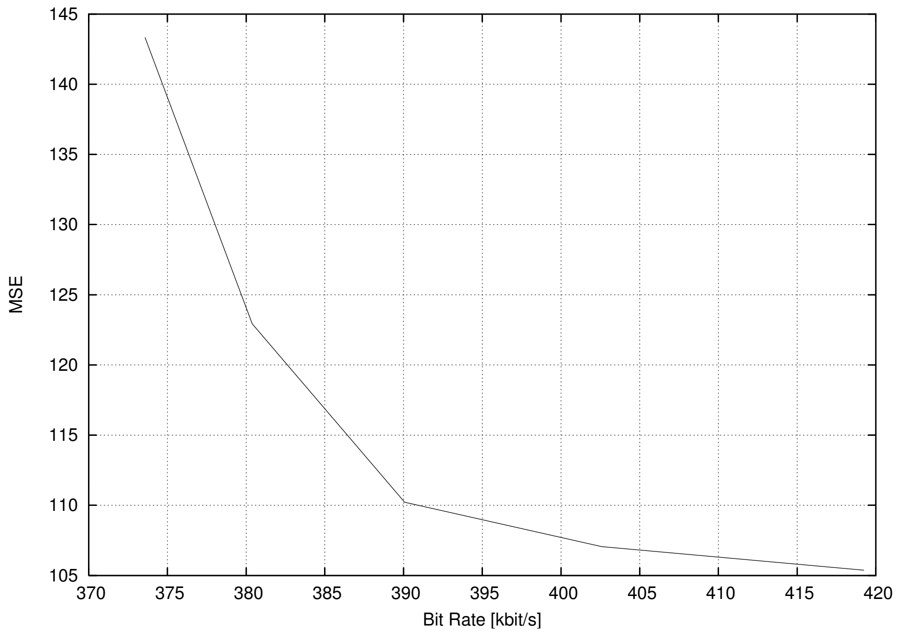


Figure 5.4: D-R-curve for $\mathrm{QP}_{\mathrm{fg}} = 24$ connected with several $\mathrm{QP}_{\mathrm{BG}}$'s, sequence "Tempete"

We have to find out which point of the D-R-curve is optimal in a rate-distortion sense regarding our coding environment. It is assumed that for every considered sequence, the behavior of our codec is similar. For finding the equation for λ, we choose four test sequences with different content and genres, i.e. "Allstars", "Entertainment", "Race1", and "Tempete". Several coded background Sprites are compounded with each of the $\mathrm{QP}_{\mathrm{fg}}$'s. As mentioned above, the main goal is to optimize our codec in the lower bit rate ranges. That means, we consider the values from Tab. 5.1 for each of the QP's .

It has been found experimentally that this setting is most appropriate for our coding environment. We encode the test sequences with each combination of the QP-values shown

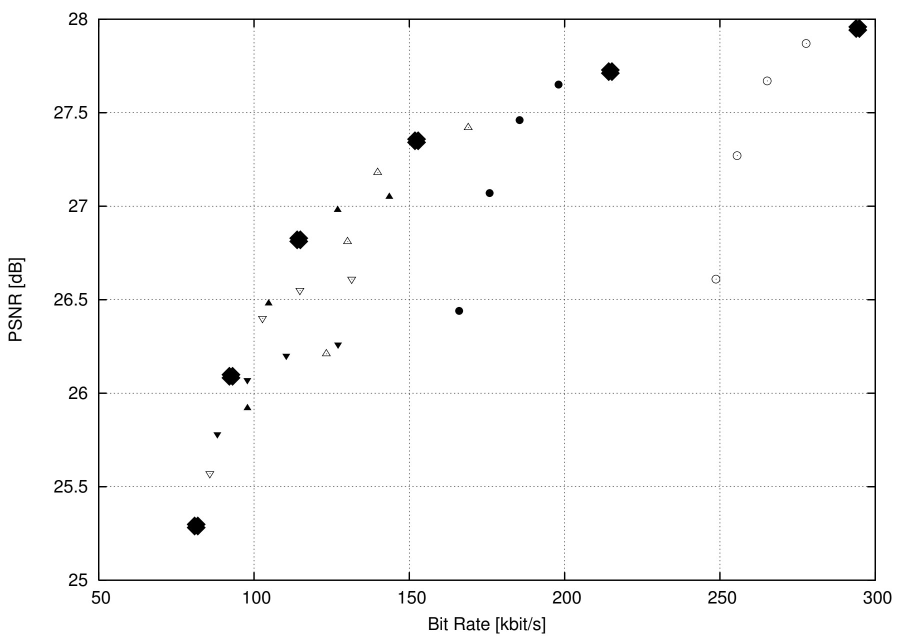


Figure 5.5: Optimal points chosen for each QP_{fg}, sequence "Tempete"

Table 5.2: Optimal λ-values for each $\mathrm{QP}_{\mathrm{fg}}$

$\mathrm{QP}_{\mathrm{fg}}$	24	28	32	36	40	44	48
λ_{opt} ("Allstars (cif)")	0.025	0.05	0.1	0.2	0.5	2.25	4.5
λ_{opt} ("Entertainment")	0.05	0.1	0.15	0.25	1.15	2.9	6.75
λ_{opt} ("Race1")	0.01	0.05	0.075	0.1	0.2	2.5	5
λ_{opt} ("Tempete")	0.05	0.1	0.15	0.25	0.75	2	4

above. Figure 5.5 shows the result for the test sequence "Tempete". Having these points we obtain for each QP_{fg} a D-R-curve with five samples comes from the differently coded background Sprites. We can see from Fig. 5.5 that for each QP_{fg} there exists one optimal point in the rate-distortion sense. Each of these optimal points is marked. Considering the D-R-curve, the optimal point is chosen and the connected Lagrangian multiplier which leads to that point is recorded. Table 5.2 shows the optimal λ-value for each QP_{fg} with the four test sequences considered.

Figure 5.6 shows the λ-QP_{fg}-curve experimentally found. Having these results, we can find a formula, which is for general use. A polynomial curve fitting technique is applied to the experimental curve to approximate it. This leads us to the following equation:

$$\lambda_{\mathrm{opt}}(\mathrm{QP}_{\mathrm{fg}}) = 0.0012 \cdot 2^{\frac{\mathrm{QP}_{\mathrm{fg}}-12}{3}} + 0.2566 \tag{5.16}$$

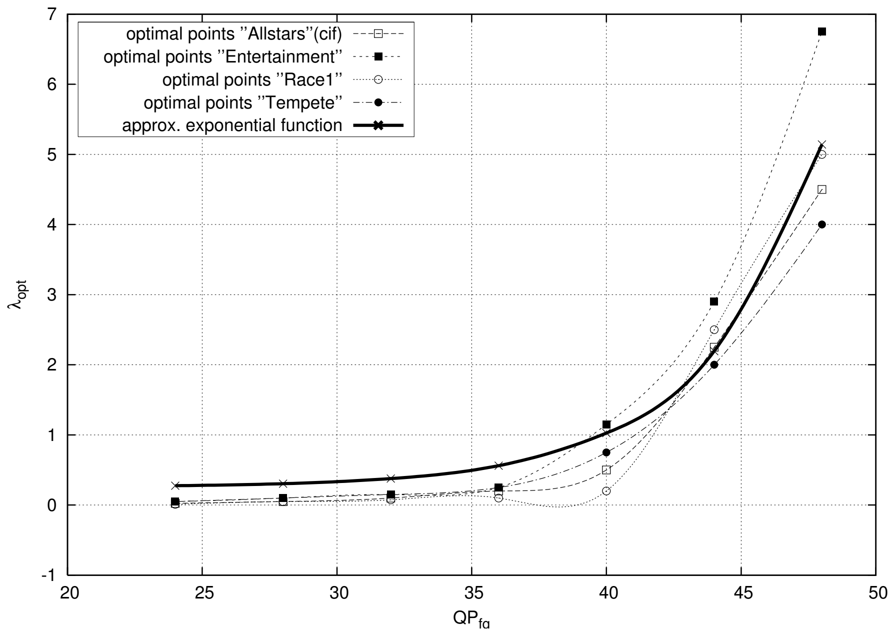


Figure 5.6: λ_{opt}-QP_{fg}-curve, experimentally found and approximated analytic function

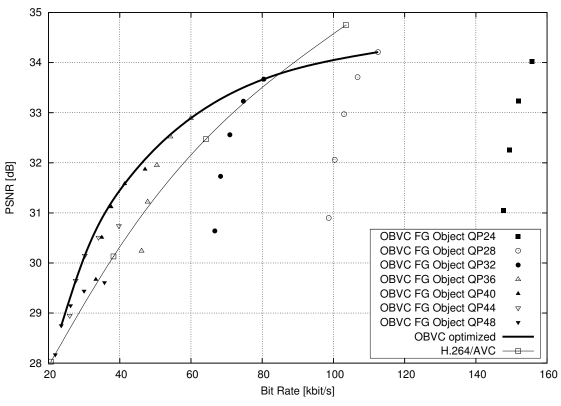
To prove this relationship, a comprehensive experimental evaluation is conducted in the next section.

5.4 Experimental results

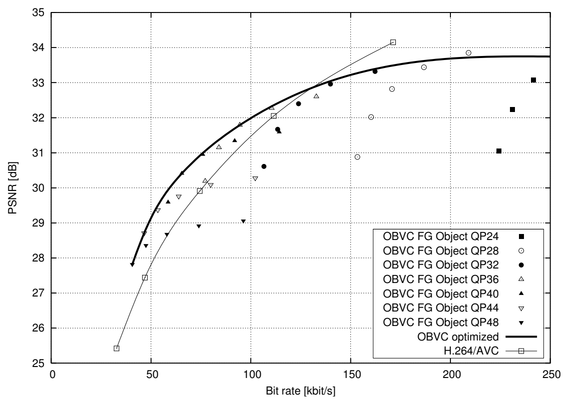
We evaluate the rate-distortion optimization technique derived above using a number of different test video sequences. As already mentioned, we consider an objective quality measurement, the very common peak signal-to-noise ratio. For the H.264/AVC codec, we use the reference software JSVM v.9.1. The encoder settings are hierarchical B-frames for the prediction and the GOP-size is 15. For the calculation of the bit rate, we assume a

Table 5.3: Test video sequences

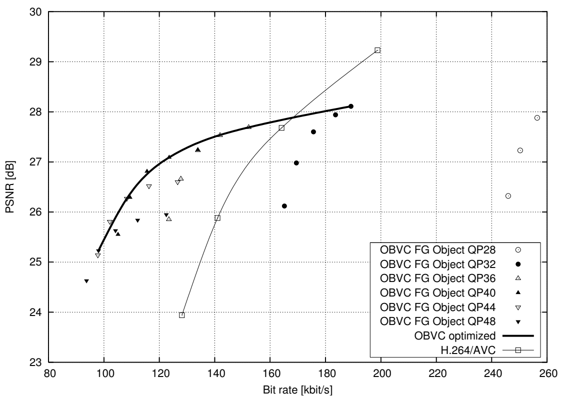
Sequence	Source	Resolution	Frames	FPS [Hz]
"Allstars" (cif)	ZDF (German TV (Channel 2))	352×288	250	25
"Allstars"	ZDF (German TV (Channel 2))	704×576	250	25
"Biathlon"	ARD (German TV (Channel 1))	352×288	200	25
"Entertainment"	VCEG	352×288	250	25
"Mountain"	BBC (Docu. *Planet Earth*)	352×192	100	25
"Race1"	MPEG	544×336	100	25
"Stefan"	MPEG	352×240	300	25
"Tempete"	VCEG	352×288	260	25



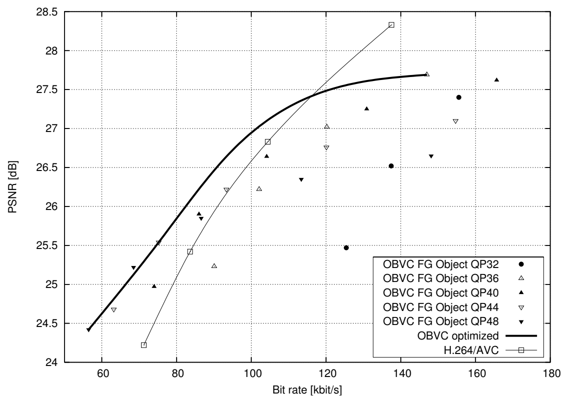
(a) “Allstars”(cif)



(b) “Biathlon”



(c) “Entertainment”

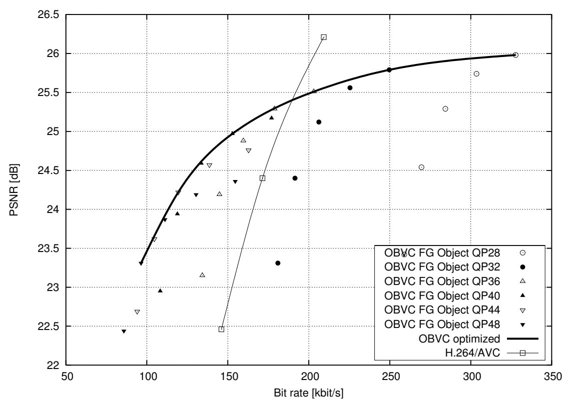


(d) “Mountain”

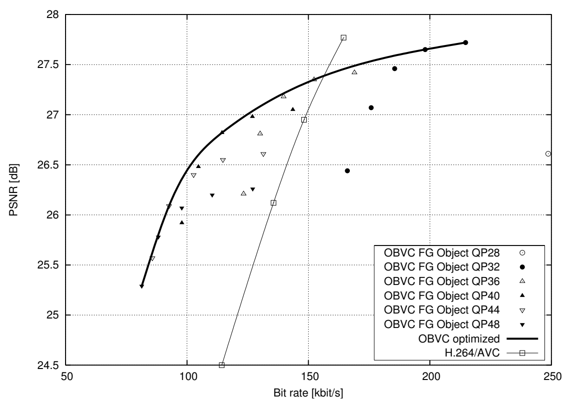
Figure 5.7: Results for optimizing OBVC (1)

frame rate of 25 frames/s. To set up a fair comparison, we keep these settings fixed for encoding the whole video sequence as well as encoding the foreground objects sequence in the OBVC. Several different kinds of test sequences are taken into account to cover a wide range of genres and content. Of course, all sequences have complex camera motion because this is the main feature we tackle with our Sprite-based technique. However, the test sequences have been chosen from various genres, e.g. sport, documentation, entertainment. We consider different resolutions to analyze if this impacts the coding performance as well. Table 5.3 lists the test sequences used. In the first part, it is examined how the optimization technique works. For that, we first figure, which points from all combinations are picked up. Then, the optimized curve of the OBVC-encoder is compared to H.264/AVC.

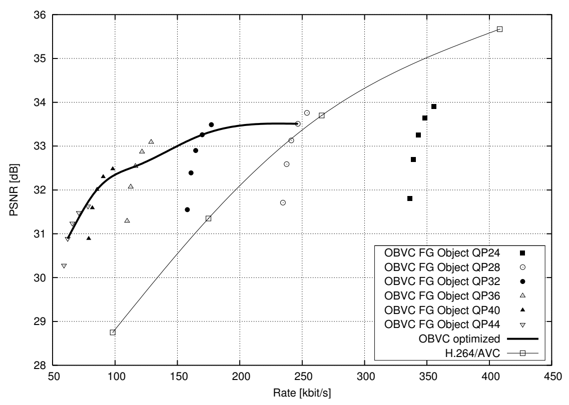
Figures 5.7 and 5.8 show the results for the test sequences considered. For the test sequences with CIF/SIF-resolution, the rate-distortion optimization scheme works very good. Nearly the optimal combination of the foreground and background QP in the rate-distortion sense are chosen. However, the equation for the assignment of λ works suboptimal on the test sequences with higher resolution. The reason is that, as descibed earlier in the chapter, the Lagrangian multiplier is the slope of the rate-distortion curve. For sequences with a



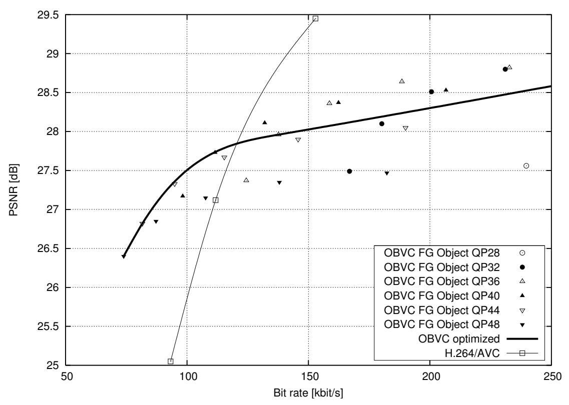
(a) "Stefan"



(b) "Tempete"



(c) "Allstars"

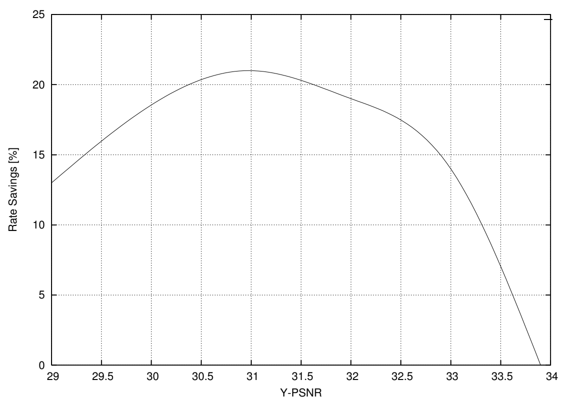


(d) "Race1"

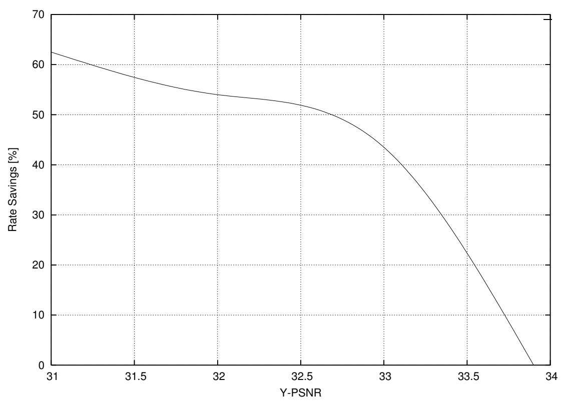
Figure 5.8: Results for optimizing OBVC (2)

unique resolution, it is possible to find an equation experimentally, which can be applied for all video material having the same resolution. By increasing the resolution, the spatial information to code increases as well, which leads to a higher bit rate, while the distortion changes slightly (compare "Allstars (cif)" and "Allstars", cf. Fig. 5.7(a) and Fig. 5.7(c)). This yields a change of the slope of the RD-curve. Therefore, the equation found for the lower resolution does not apply on the sequences with higher resolution. Thus, new coefficients have to be found when the OBVC is used for coding sequences with higher resolutions, which is a big task for future work.

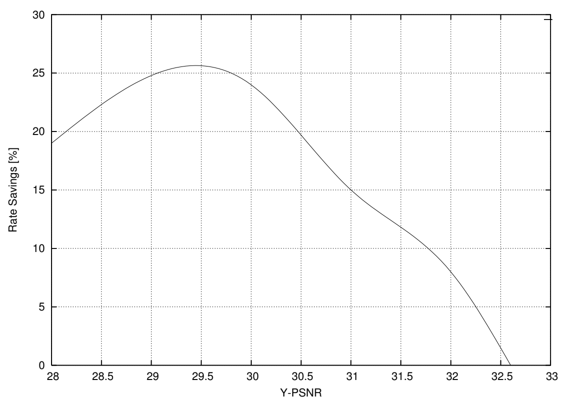
In general, it can be seen that in sequences with spatial detail as well as complex camera motion including fast pan and strong zoom, the OBVC outperforms common H.264/AVC up to 50% for the lower bit rate ranges. Another fact is that in higher resolution the bit rate saving increases as well. This means that the potential of Sprite-based coding techniques is very high for TV-resolution and even for the next step, HD-TV. A drawback here is the increasing computational complexity at the encoder during the pre-processing steps (global motion estimation and background Sprite generation).



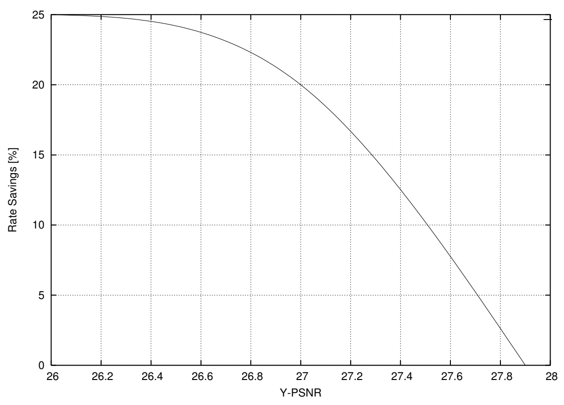
(a) “Allstars”(cif)



(b) “Allstars”



(c) “Biathlon”

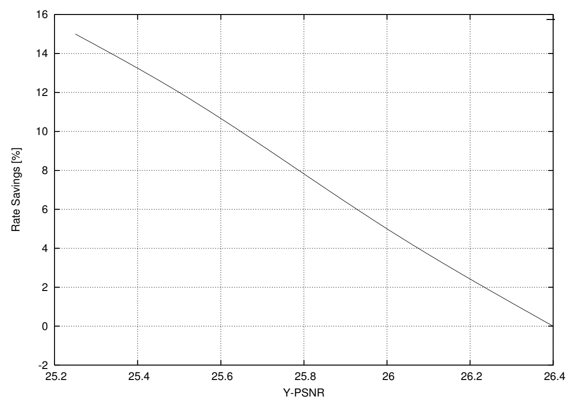


(d) “Entertainment”

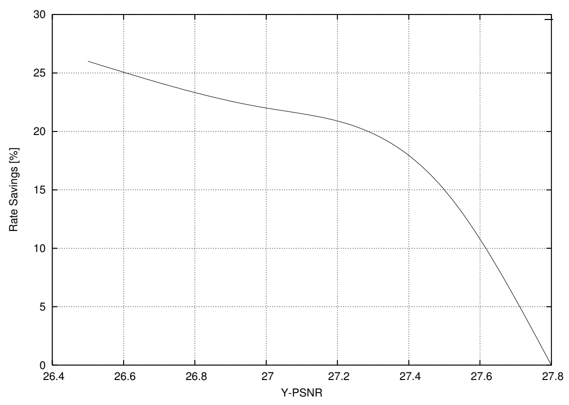
Figure 5.9: Bit rate savings in % of OBVC optimized compared to H.264/AVC (1)

5.4.1 Comparison of optimized OBVC and MPEG-4 Sprite Coding with automatic object segmentation

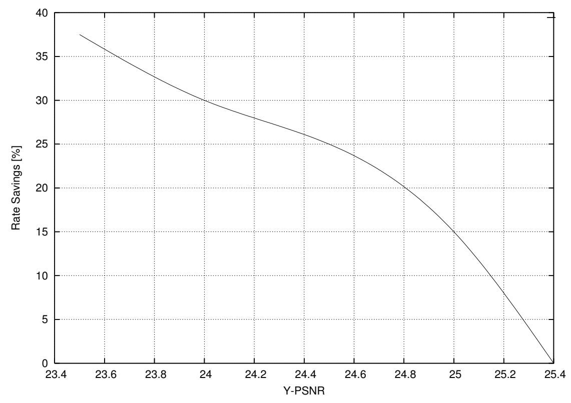
Having the OBVC optimized, an interesting experiment is the comparison with the MPEG-4 Sprite coding technique using the same automatic object segmentation as described in Chapter 4. We have shown that for certain test sequences the OBVC outperforms the common use of H.264/AVC significantly. However, H.264/AVC is not designed for an object-based coding structure. For that, we compare the best configuration of the automatic MPEG-4 Sprite coding setting, i.e. *Algorithm3* for automatic object segmentation (cf. Chapter 3 and 4) and Multiple Sprites for test sequences “Biathlon” and “Stefan”, with common use of H.264/AVC and OBVC. Figures 5.11 (a) - (d) show the rate-distortion curves for four test sequences. It can be seen that the common use of H.264/AVC outperforms the MPEG-4 Sprite coding technique, except for a small bit rate range of the “Stefan”-sequence. This experiment substantiates the advanced coding tools of H.264/AVC. However, using the MPEG-4 object-based coding idea with H.264/AVC brings again a performance gain. To this end, we can summarize that even for further enhanced coding standards, it will be possible to outperform the common hybrid video codec using the object-based representation including



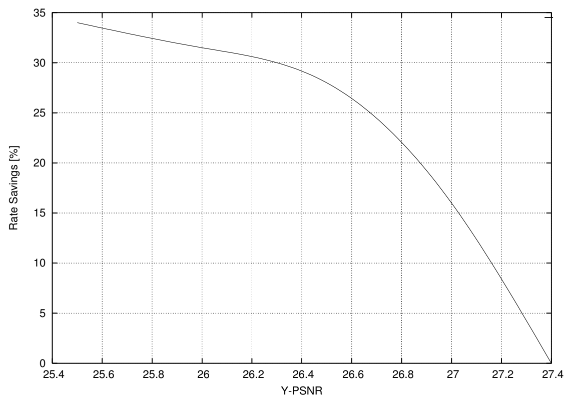
(a) “Mountain”



(b) “Race1”



(c) “Stefan”



(d) “Tempete”

Figure 5.10: Bit rate savings in % of OBVC optimized compared to H.264/AVC (2)

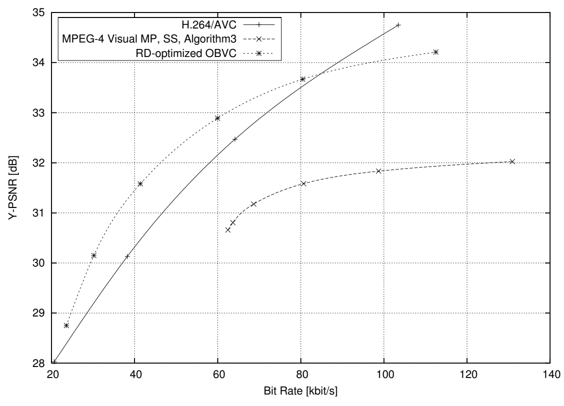
background Sprites.

5.5 Further work

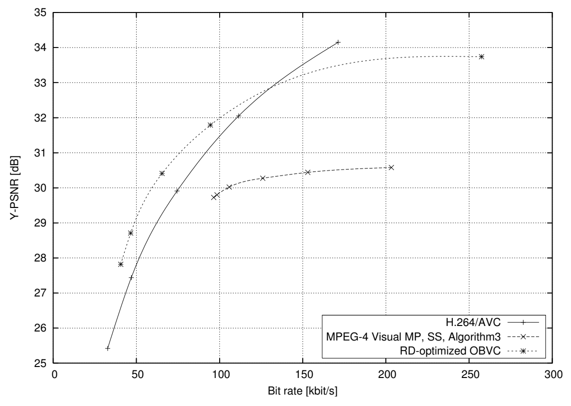
This section draws an overview of other issues, which could be considered regarding the optimization of the OBVC. We mainly point to the very important pre-processing steps including background Sprite generation and in-built object segmentation.

5.5.1 Background Sprite Generation

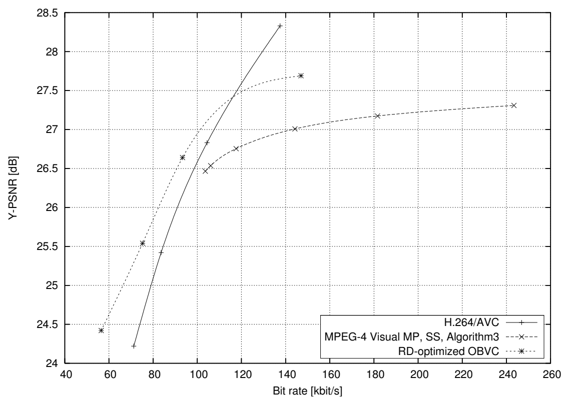
Much research has been done in generating background Sprites from a sequence [70], [12], [11], [93], [42]. The main goal in each of these works is basically to generate a background Sprite as accurate as possible to have the highest possible PSNR-value of the reconstructed scene in comparison to the original. This has led to several approaches ranging from single-, multiple-, and superresolution background Sprites. The latest development introduces a superresolution multiple Sprite, which seems to be optimal in the sense of obtaining the highest PNSR-values for the reconstructed images [94]. Beside the high accuracy of the



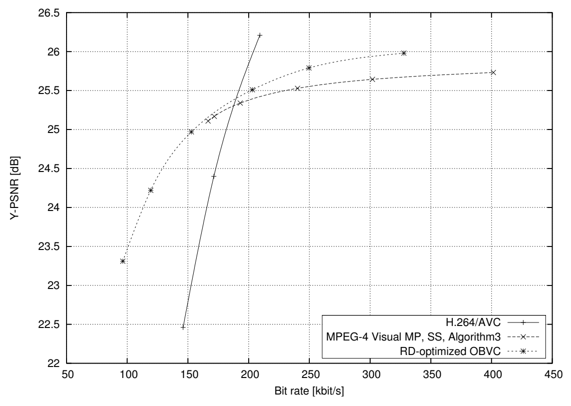
(a) "Allstars"(cif)



(b) "Biathlon"



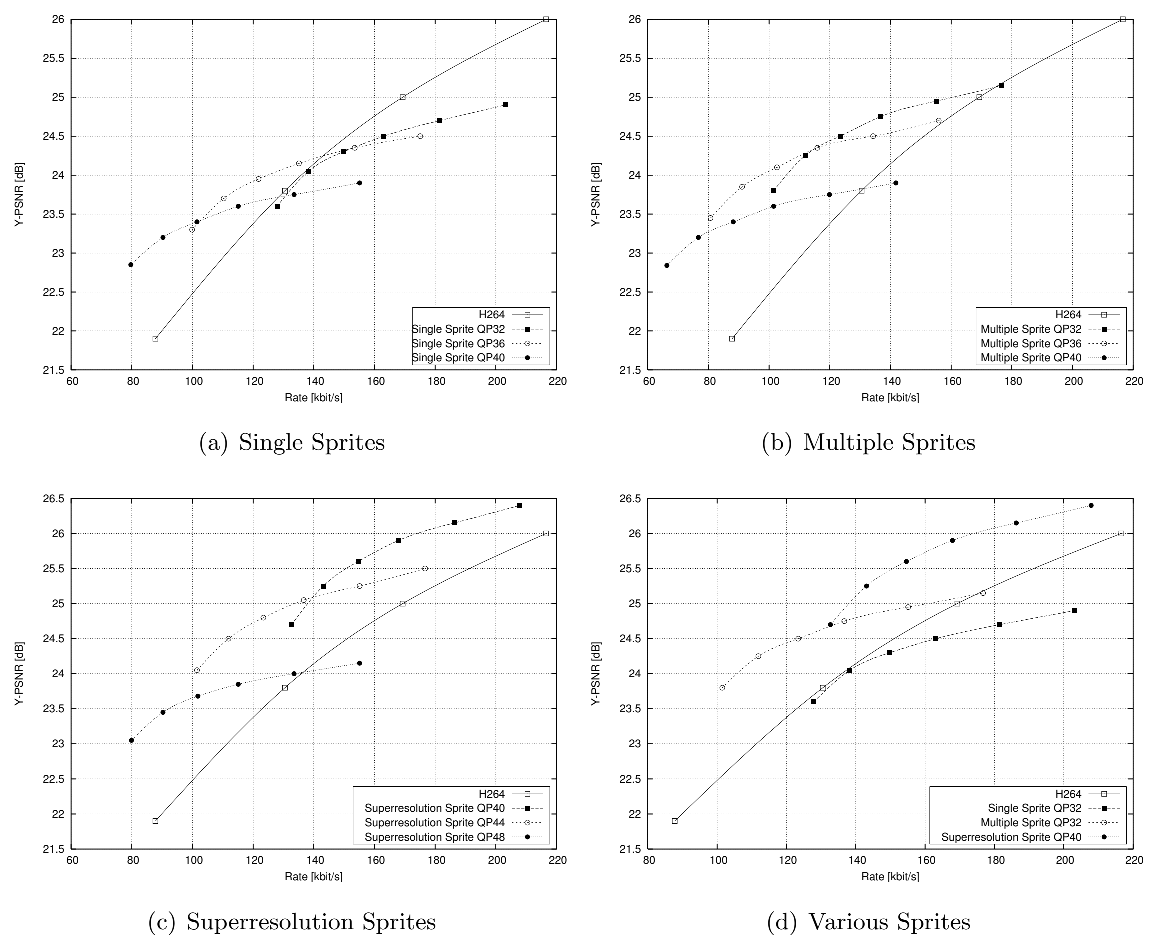
(c) "Mountain"



(d) "Stefan"

Figure 5.11: Comparison of OBVC (optimized), MPEG-4 Sprite Coding, and common H.264/AVC

reconstructed frames, only the coding efficiency by encoding the Sprite images has been considered. The performance of different kinds of background Sprite images is shown in Fig. 5.12. Here, single-, multiple-, and superresolution Sprites are used within the OBVC. It can be seen that using the superresolution Sprite, it is possible to outperform H.264/AVC further to higher bit rate ranges in comparison to the multiple Sprites. However, in lower bit rate ranges the use of multiple Sprites is more efficient. The reason for this is that the cost of the higher accuracy of the reconstructed frames from the superresolution Sprite is the upsampled version of the background Sprite image, i.e. we have more data to encode. Thus, it is the same optimization problem described and evaluated above. It is possible to apply the same strategy including the superresolution background Sprite and we expect that the algorithm will choose the right "mode", i.e. for higher bit rates the superresolution and for lower bit rates multiple background Sprites.



(a) Single Sprites

(b) Multiple Sprites

(c) Superresolution Sprites

(d) Various Sprites

Figure 5.12: Comparison of various kinds of background Sprites within OBVC ("Stefan") [39]

5.5.2 In-built object segmentation

The second issue which can be tackled using the optimization approach described above is the in-built automatic object segmentation. Beside a number of processing steps during the segmentation algorithm, thresholding is applied for converting the error image to a binary version. We use the first statistical momentum of the error image after some pre-processing inlcuding anisotropic filtering. This value can be varied using a tuning constant that weights the foreground objects more or less. This impacts the amount of pixels considered as foreground. If the threshold value is higher, less pixels are taken into account as foreground, which leads to higher coding gain. The drawback is that more artifacts in the real foreground objects appear. We now take the test sequence "Entertainment" and use a number of threshold values and apply the whole encoding process. For that, we left "QP_{bg}" fix and varied only "QP_{fg}" to emphasize the coding behavior of the use of different threshold values. The RD-curves are depicted in Fig. 5.13. It is obvious that there is the same optimization problem as above. The higher the PSNR of the threshold value used the higher the bit rate. This means that we can apply our optimization technique and pick the optimal point of each

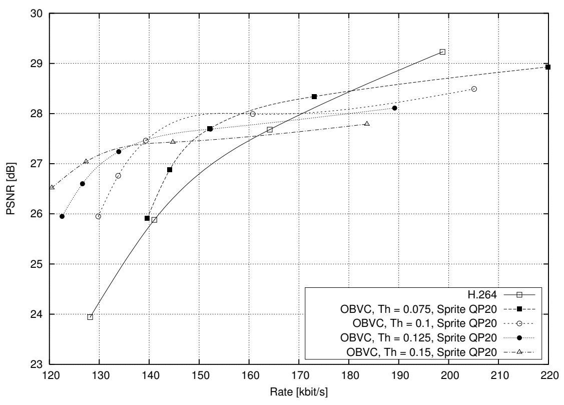


Figure 5.13: Comparison of various threshold values and the impact within the OBVC ("Entertainment")

curve, which means of each threshold value. There are two drawbacks. The first drawback is that we have to encode the entire foreground objects sequence with different "$\mathrm{QP}_{\mathrm{fg}}$'s" and thresholds, which is very time consuming. The second drawback is that towards lower bit rates higher threshold values would be taken because of the better RD-performance. This means that for lower bit rates the artifacts in the foreground objects due to under segmentation would increase. However, in practice it has been found that there are only a few threshold values which would come in consideration. So the real impact of adjusting the threshold in a rate-distortion sense is worth a try.

5.6 Chapter Summary

We considered the rate-constraint encoder optimization problem of our automatic Sprite codec "OBVC". The well-established Lagrangian optimization method is taken into account, which has become the de-facto standard for rate-distortion optimization in hybrid video codecs. This approach was first introduced for common hybrid video codecs. The issue considered more in detail regarding the "OBVC"-encoder was the right choice of the "QP" for the foreground object sequence and the "QP" for the background Sprite. Problems were analyzed first towards the extension of the Lagrangian approach to the "OBVC". The outcome was that the very critical issue of choosing the right Lagrangian multiplier λ cannot be adopted from the derivation of the common hybrid codecs. Due to the complexity of the pre-processing steps during the background Sprite generation, a method for deriving a relationship between the Lagrangian multiplier λ and the quantization parameter of the foreground object "$\mathrm{QP}_{\mathrm{fg}}$" has been found experimentally. This equation is evaluated comprehensively. For that, eight test sequences from different genres with different characteristics and resolutions are used to show on one side the performance of the optimization technique and on the other side the performance of the newly optimized results compared to H.264/AVC. Overall, the optimization technique works very well. For all sequences nearly all optimized point in the rate-distortion sense are chosen by the encoder automatically. In comparison to H.264/AVC, we can say that we achieve for seven test sequences bit rate savings up to 60 % and 20 to 30 % as a mean value. We obtain these improvements in lower bit rate ranges because of the maximum PSNR-limit due to the use of background Sprites.

In the last section, we discussed further issues which can be necessary for the optimization process of the "OBVC"-encoder. The use of different kinds of background Sprites and different choices of a threshold value during the in-built object segmentation were mentioned. We showed that the optimization method described above can be easily extended to these further problems. This would be the next step for further work.

Chapter 6

Video Content Analysis for Automatic Sprite Coding

In this chapter, we tackle the problem of general usability of the Sprite-based coding approach. We have shown that this coding technique outperforms common hybrid video codecs for certain video sequences, i.e. sequences having a fixed camera and containing small foreground objects and almost static background, e.g. sport broadcast or documentary. The goal is to design a content-adaptive video codec including an automatic pre-analysis step to assign the optimal encoding technique depending on the content and camera motion characteristics. First algorithms are presented in this chapter.

6.1 Introduction

The task is to automatically detect material where the Sprite-based codec can be applied. To achieve this, the content of the considered video has to be analyzed first. We apply the global motion estimation algorithm used throughout this thesis to detect the motion of the camera. The first approach is to use the short-term motion parameters calculated already during the Sprite generation step for pre-analysis of the video, including shot-boundary detection. These parameters are also used to build a criterion for choosing the most appropriate video codec, i.e. either the Sprite-based codec or common H.264/AVC. Every shot is then coded using the chosen video codec. At the decoder, all shots are decoded and merged to the original scene. An overview of this content-adaptive video coding system is given in Fig. 6.9.

In a second approach, we will show that the codec selection only relies on the camera motion of the sequence. We consider the object-based video codec (OBVC) in combination with H.264/AVC for camera pans and common H.264/AVC for camera motion types where the OBVC does not work, e.g. camera track. Two algorithms are presented and compared for recognizing the camera motion type. The sequence is then segmented into sub-segments relying on the several motion types. A first technique uses the frame-by-frame global motion compensated error values RMSE (root mean square error) as the input data. Using the eight-parameter perspective motion model [55], camera motion, such as pan, tilt, zoom, rotation, can be estimated very well. This leads to small RMSE-values in the global motion compensated error frames. Having the threshold based on the variance on the RMSE-curve calculated, the video is segmented into two segments for Sprite coding mode (RMSE-values below the threshold) and H.264/AVC mode (RMSE-values above or equal the threshold). The second technique relies on feature tracking and motion model selection. Here, the camera track is recognized using the *Geometric Robust Information Criterion* (GRIC) [82].

Assuming, that the camera is tracked, this sub-sequence is coded with the H.264/AVC mode. All other sub-sequences are coded with the Sprite coding mode. We will show that this codec selection based on the camera motion outperforms the coding performance of using the H.264/AVC for the whole sequence.

6.2 Low-level Content-Analysis using Global Motion Estimation

Part of this section has been developed together with [59].

6.2.1 A hybrid shot boundary detection method using GME

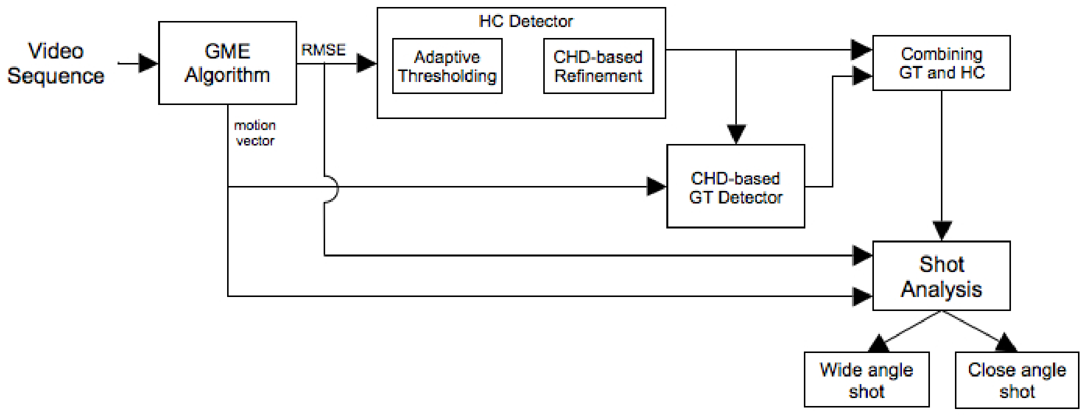


Figure 6.1: Block diagram of the SBD and shot classification stages

The first step in the process is to apply the global motion estimation algorithm over the input video sequence. The outputs required from this algorithm to face the automatic shot boundary detection (SBD) and the shot analysis are the RMSE curve computed for each two frames after the motion estimation and the estimated global motion vector for the whole sequence. Firstly, the SBD process is carried out in order to determine the transitions between shots within the sequence. Concretely, the developed technique for SBD is able to detect both abrupt transitions (or hard cuts, HCs) and gradual transitions (GTs, such as wipes, dissolves and fades). Once the frames, where the shot transitions take place, are found, these shots are segmented from the video sequence and analyzed separately. Specifically, this analysis tries to classify the shots into *wide angle shots*, which are shots captured from a considerable distance to the scene, and *close angle shots*, in which the foreground objects occlude a large part of the background.

The SBD process starts with the detection of the HCs. The motion discontinuity which takes place between two adjacent shots when a HC occurs is reflected in the RMSE curve as a sharp peak. In order to reliably detect those peaks, an adaptive threshold technique, based on the approach presented in [83], is applied over the RMSE time series. Concretely, this thresholding method is based on the use of a sliding symmetric window of size $2w-1$, which progressively covers the whole curve. Therefore, the sliding window is located in each RMSE value, and two conditions are analyzed in order to decide whether the value concerns a HC or not. Firstly, the current RMSE value must be the local maximum within the window. Secondly, the current value must be a number of n times greater than the mean of the remaining values of the window. The mathematical expression is the following, where r is the current RMSE value and c is a constant added both to the mean and the current RMSE value to make the algorithm robust against situations when the mean is approximately zero and the threshold can be too low:

$$r_i + c \geq n \cdot \frac{\sum_{j=i-w+1}^{i+w-1}(r_j + c)}{2w-1}$$

It is worth noting that the parameter w, which determines the size of the window, must be smaller than the minimum distance between two shot transitions. Moreover, when a HC is declared, it is not necessary to check the following discontinuity values which where within the window, because these values are not maximums and do not satisfy the first condition. Thus, the following discontinuity value to analyze should be the next value immediately to the end of the previous window.

After this step, a previous set of frames corresponding to possible HCs is obtained. In order to remove false detections which can be declared as HCs, a refinement process is applied based on the color histogram difference [44]. Concretely, for each possible HC detected after the thresholding, the two previous frames and the following to the frame where the HC has been found are taken. Then, the color histogram difference is computed for the last frame of the existing shot (the immediately previous to the location of the detected HC) and first frame of the entering one (the following frame to the detected location for the HC), obtaining color histogram difference (CHD) 1 CHD_1. The same value is calculated for the two immediately previous frames to the HC, getting CHD_2. The color histogram differences are computed using the following expression, where $p_i(r, g, b)$ is the number of pixels of color

(r, g, b) in the frame I_i of the video sequence. N is the total number of pixels of the frames of the sequence and B is a value (generally 2 or 3) used to discretize all the colors to 2^B different values, thus $r, g, b \in [0, 2^B - 1]$, in order to reduce the influence of noise and light changes in the results.

$$\text{CHD} = \frac{1}{N} \cdot \sum_{r=0}^{2^B-1} \sum_{g=0}^{2^B-1} \sum_{b=0}^{2^B-1} |p_1(r, g, b) - p_2(r, g, b)|$$

If the detected HC is actually an abrupt transition, CHD_1 should be much larger than CHD_2, taking into account the color distribution discontinuity between the entering and the exiting shots when a HC occurs. Thus, the following expression is evaluated in order to declare a HC or a false alarm. After these last step, the HC detection process is finished.

$$\begin{array}{ccl} \text{CHD}_1 \geq n \cdot \text{CHD}_2 & \rightarrow & \textit{Hard Cut} \\ \textit{otherwise} & \rightarrow & \textit{False Alarm} \end{array}$$

Once the HC detection is overcome, the detection of gradual transitions is tackled. The developed technique is able to detect GTs in general, without distinguishing among the different existing types of GTs (such as wipes, fades, dissolves and others special editions). With this aim, between each two HCs detected in the previous stage, the CHD for each two contiguous frames is computed, obtaining a CHD time series for each segment of the video sequence resulting after the HC detection (lets call these segments s_i where $i = 1, 2, ..., N$ and N is the number of detected shots). In this time series, unlike the HCs which produce sharp peaks, the GTs cause “bump” of lower level than those peaks, showing the progressive change between two shots.

Taking this fact into account and extrapolating the idea of the thresholding technique used for the HC detection using a larger sliding window, those “bumps” can be detected (also considering that the same window should not cover two GTs at the same time). Concretely, a set of frames which satisfy the two requirements aforementioned are detected first, called m_j where $j = 1, 2, ..., M$ (where M is the number of sets). Among these frames are those where the CHD value is maximum in a GT, but also frames related to other effects like camera or object movements and luminance changes. In order to remove these possibly false detections, three algorithms are executed.

Firstly, the mean of the estimated global motion values corresponding to the frames within the sliding window are computed. Then, if the estimated global motion value related to the frame m_j is much larger (for example two times), than the computed mean value, the detection is considered a motion effect and is removed from the possible GTs.

Moreover, in order to detect possibly false detections caused by abrupt changes of luminance within a shot, a detector of these kind of effects is implemented. In concrete, a neighborhood of a number of frames around m_j is defined. Then, the mean luminance is calculated for each frame within the neighborhood and then the maximum and the minimum values obtained are taken. If the maximum is much greater than the minimum, then an abrupt change of luminance is considered and not a GT.

Finally, at the same time that the large sliding window is applied over the segment of the video s_i, another sliding window but much smaller (usually with a size of nine frames) is used. With this, isolated peaks of comparable levels to the GTs (and obviously much lower than those related to HCs) are detected and removed from the set of possible GTs too.

After all these processes implemented to eliminate false alarms, a second set of frames corresponding to the maximums (generally intermediate frames in the transition) of the detected GTs is obtained, $m2_k$ where $k = 1, 2, ..., M2$ and $M2$ is the total number of detected GTs after the refinement steps. Then, considering that the remaining frames are related to GTs, it is time to search the first and the last frame of each GT. With this aim, each direction (to the right and left) is followed from the detected maximum until a value is found that is a fixed times lower than the maximum. These are the discontinuity values related to the first and the last frame of the GT, and the process is finally completed.

Then, all the shot transitions detected, (both HCs and GTs) are combined and the shot segmentation is carried out to obtain the inputs to the analysis stage. As said before, the shots are classified into *wide angle shots* and *close angle shots*. In order to distinguish between these two types, statistical and motion features are considered, taking into account the experiments presented in [34]. Therefore, the RMSE curve and the estimated global motion vector, are considered to face the shot classification task. These signals are segmented, taking separately the part corresponding to each video shot, and for each one of these segments the following four features are computed: the mean and the first derivative of the variance of the RMSE segments, and also the mean and the variance of the segments of the global motion vector. Finally, analyzing the obtained values, a threshold for each feature is heuristically selected and the four features are combined in order to achieve a reliable shot classification.

The following tables show the experimental results obtained after applying the whole process to the "Allstars" soccer sequence. Table 6.1 shows the parameters of the test sequence.

Table 6.2 shows the final results of the SBD process using the performance measure recall and precision in percentage. It is worth noting that two of the gradual transitions considered in the groundtruth are not really GTs, since no shot change takes placed. However, the appearance of the score in the bottom part of the frames with a considerable size produces a similar effect to a GT. However, this effect can be removed applying some techniques of text detection avoiding to declare it as a GT.

Sequence	Frames	Rate (fps)	Resolution
Allstars	25083	25	704x576

Table 6.1: Parameters of the test video sequence.

Groundtruth			SBD results					
			HC		GT		Total	
Total	HC	GT	R	P	R	P	R	P
58	33	25	100	100	88	95.65	94.83	98.21

Table 6.2: SBD experimental results.

6.2.2 Shot Analysis

To find a criterion for the decision of the video codec, the differential RMSE-curve is considered, as for shot boundary detection. In the case presented above, the sequence is segmented into three shots. We calculate the variance of each curve segment. It can be seen in Fig. 6.2 (a) that the variance of shot 1 and 3 is less than for shot 2. We know from earlier examination that the object-based video codec achieves a higher coding gain for sequences like shot 1 and 3. For shot 2, it is not possible to build a video mosaic because of this very close camera shot with a large foreground object. In that case, it is very hard to segment. Finally, we know from our recent experiments that the background object has to be much larger than the foreground objects to gain more coding efficiency. So we need a criterion which distiguishes shot 1 and 3 from shot 2. For the example shown in Fig. 6.2, the variance values of the differential RMSE for shot 1 and 3 are 0.4 and 2.4, respectively. The variance for shot 2 is 15.6. This means that the short-term frame-to-frame image registration is very unstable for the second shot. These motion paramters set up the mosaic generation algorithm and if the accuracy of the background estimation varies in that way an accurate mosaic cannot be generated. Considering these variance values, a threshold has to be defined. We calculate the mean of the three variance values in a pre-processing step.

Groundtruth		Analysis results			
WAS	CAS	R	P	WAS % frames	CAS % frames
20	35	95	100	72.27	27.73

Table 6.3: Shot analysis experimental results.

Table 6.3 shows the results of the shot analysis stage. The numbers of the groundtruth represent the number of wide angle shots (WAS) and close angle shots (CAS) in the whole “Allstars” sequence (15 min.). There are two shots that do not belong to any of these types. The results show the recall and precision of the shot classification process. It also shows

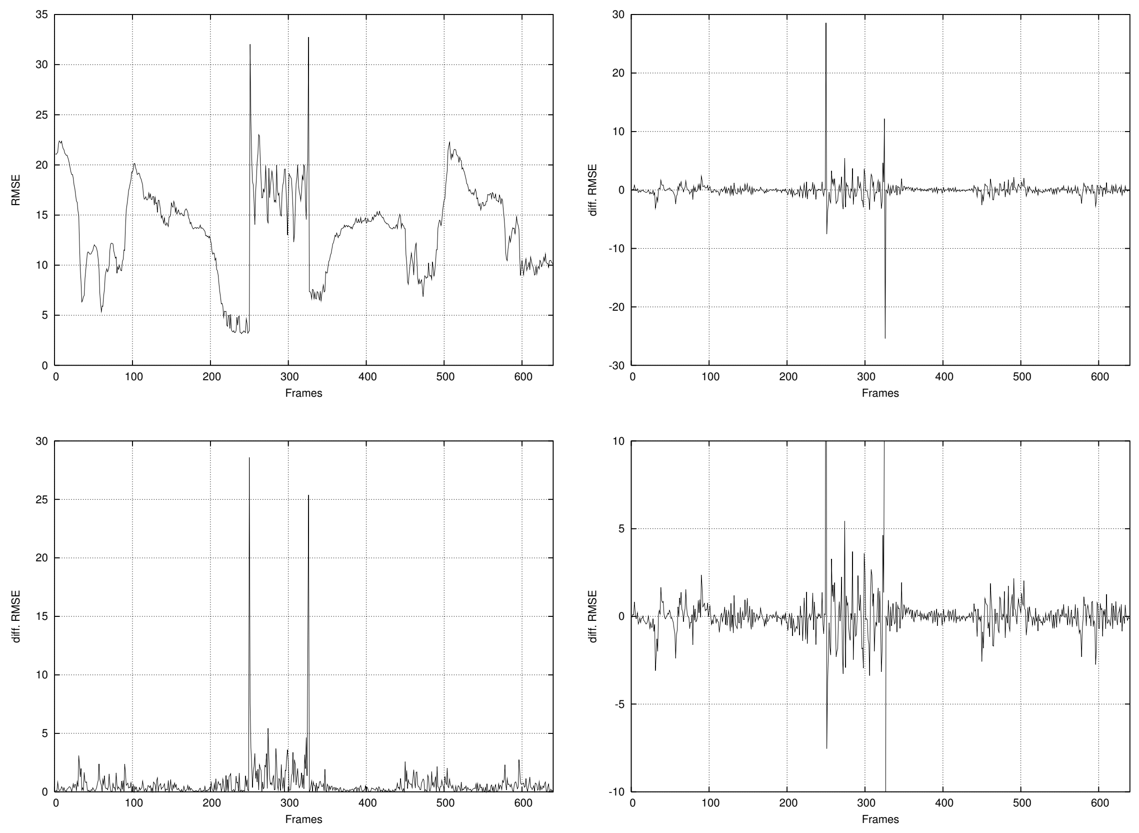


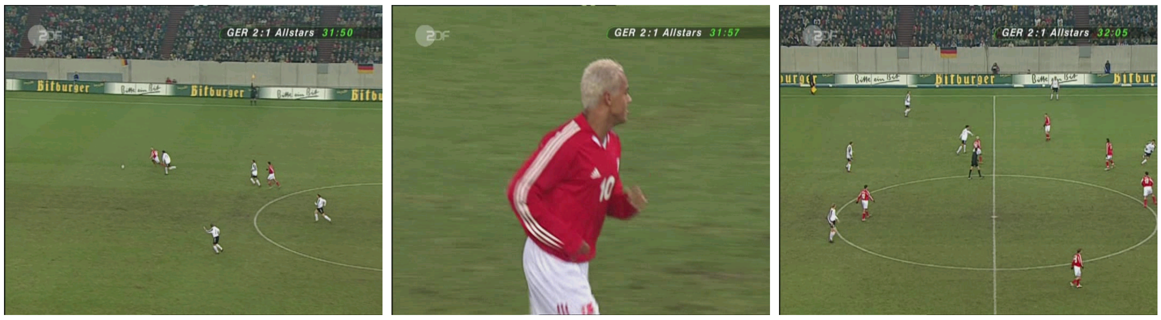
Figure 6.2: Frame-by-Frame RMSE analysis

the percentage of frames belonging to wide angle shots and close angle shots with respect to the global number of frames.

6.3 Video Segmentation based on Camera Motion

6.3.1 Algorithms for Selecting an optimal Video Codec depending on the Camera Motion

The fundamental idea is to segment the input sequence depending on the camera motion of the scene. We consider two types of camera setups, fix camera with pan, tilt, zoom or roll and camera track. It has been shown over the last decade that video seqences with a fix camera (pan, tilt, zoom, roll) can be coded very efficiently with an object-based coding approach, i.e. Sprite coding. If the camera tracks during the sequence, Sprite coding is not possible. Therefore, we will automatically segment the input sequence into sub-sequences and code the sub-sequences with different video codecs, i.e. Sprite coding (e.g. pan) and H.264/AVC (track). The next two sub-sections describe two approaches for analysis and segmentation of a video sequence regarding its camera motion.



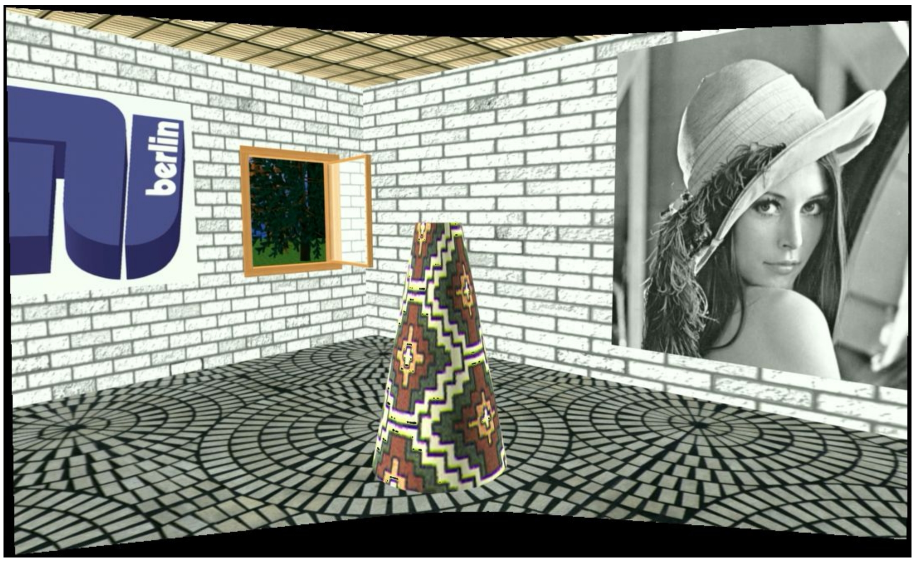
(a) Keyframe (shot1) (b) Keyframe (shot2) (c) Keyframe (shot3)

Figure 6.3: Keyframes of the three shots detected

6.3.1.1 Segmenting the Sequence using Global Motion Estimation

To find a criterion for the decision of the video codec, the RMSE-curve over the whole input sequence is considered. We first apply a frame-by-frame global motion estimation algorithm [27]. Then, for each frame a global motion-compensated error frame is computed. The RMSE-value based on this error frame is taken as quality criterion of the estimation. After applying a temporal median filter on the RMSE-values of the whole sequence for noise reduction, we calculate a threshold which defines the two types of camera motion. If the camera is panning the global motion estimation with the 8-parameter perspective camera model performs quite well for this motion type. The RMSE-values will be low. If the camera tracks it is not possible to have a stable estimation of the global motion because the camera model does not fit anymore. The RMSE-values will increase in this sub-sequence. We calculate the variance of the whole RMSE-curve defining the threshold. We apply this algorithm on one synthetic test sequence "Room3D" with two pans and one camera track. Figure 6.4 shows the result for the first test sequence. It can be seen that the threshold seperated the sequence in three sub-sequences. The first and the third are sub-sequences with camera pans (RMSE below the threshold) and the second sub-sequence has a camera track (RMSE above the threshold). That means, the first and the third sub-sequence can be coded with the Sprite-based approach and the second sub-sequence is coded with common H.264/AVC. Figure 6.4 shows the three sub-sequences. The first and the third sequences are already transformed into background Sprites [40]. For the second sub-sequence, the first, the middle, and the last frame are depicted.

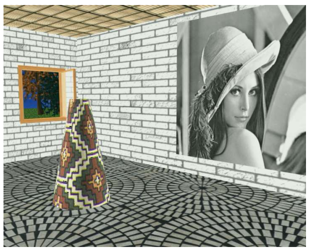
The second test sequence was captured with a hand-held camera and thus it is comparable with common home videos. The first part of the sequence has a camera track and the second part is a camera pan. The way of capturing this sequence is lean on for example home videos. The result for the second video sequence can be seen in Fig. 6.5. For the first sub-sequence the first, middle, and last frame are shown again for the camera track. The second sub-sequence (pan) is also shown in background Sprite-based representation.



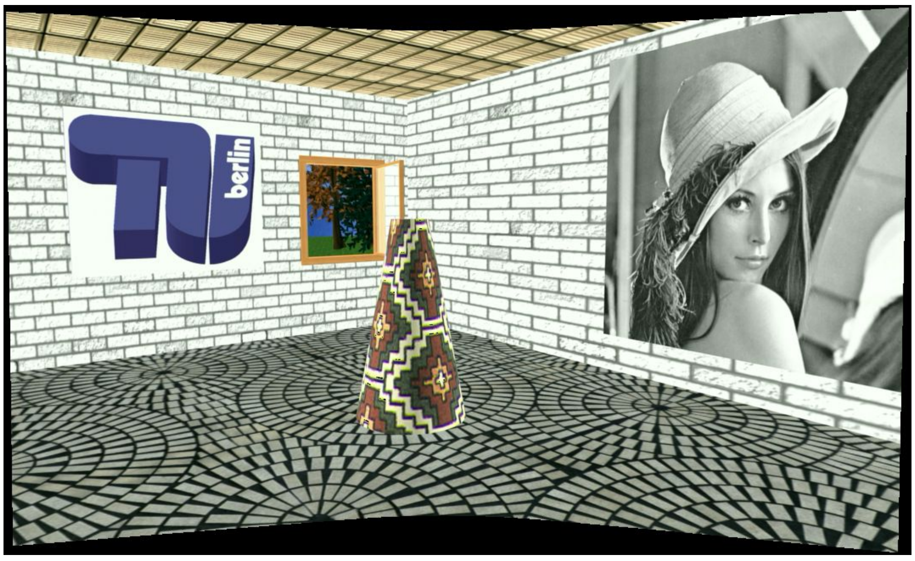
(a) pan, frames 1-67

(b) camera track (first)

(c) camera track (middle)

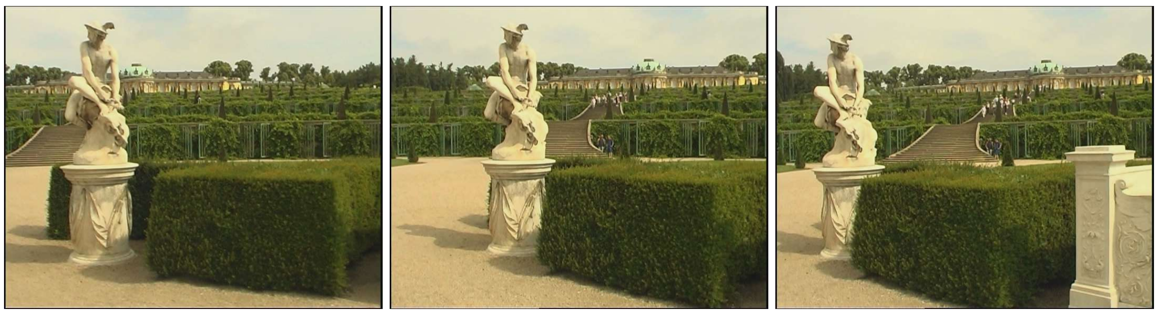


(d) camera track (last)

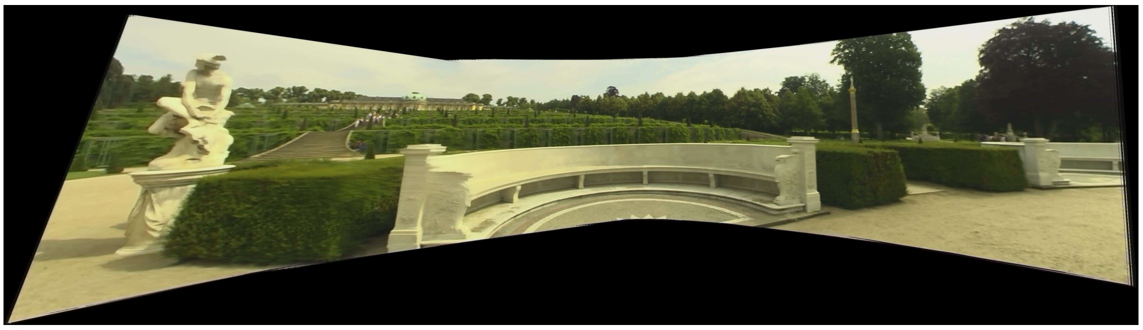


(e) pan, 116-160

Figure 6.4: Video segments using GME of the synthetic sequence “Room3D”



(a) camera track (b) camera track (c) camera track



(d) pan, 199-400

Figure 6.5: Video segments using GME of the real sequence "Castle"

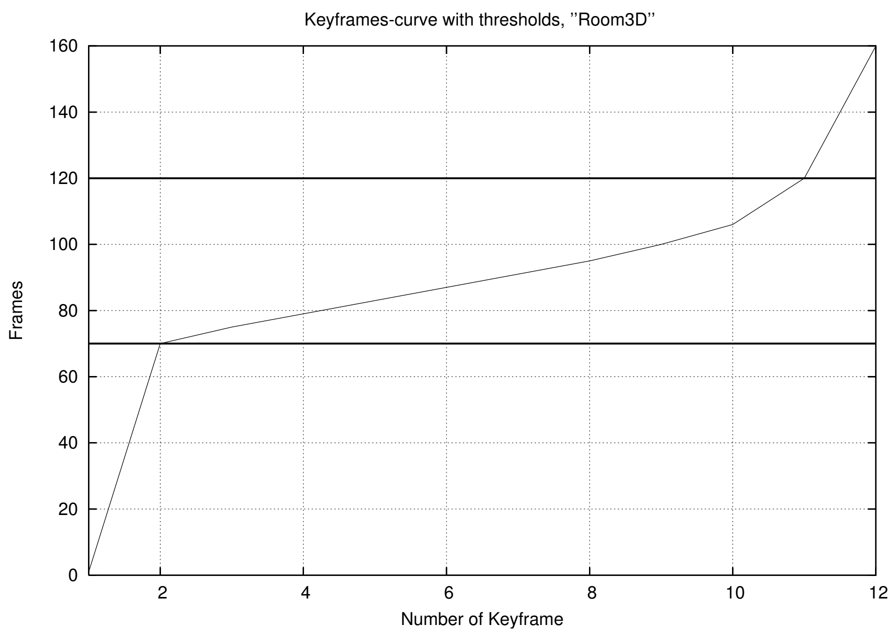
6.3.1.2 Segmenting the Sequence using Feature Tracking and GRIC - Score

The second algorithm for analyzing the camera motion is based on a motion model selection approach. A slightly modified version of the well-known Kanade-Lucas tracker [81] is used to track feature points throughout the video sequence. Since the baseline between consecutive frames is small or the camera rotates about its center, a 2D perspective motion model, H (homography), can be used to transfer features from one frame to their corresponding positions in the second frame [55]. If the baseline, i.e. the estimation between two frames, increases during the tracking process and if the features belong to a 3D scene structure, the transfer error increases as well. Thus, the 2D motion model must be upgraded to a 3D motion model, F (epipolar geometry). Hence, the current frame is the intersection between both motion models and can be selected as a keyframe. The *Geometric Robust Information Criterion* (GRIC) [82] is a robust model selection criterion to extract such keyframes and is defined as:

$$GRIC = \sum \rho(e_i^2) + \lambda_1 dn + \lambda_2 k \tag{6.1}$$

where $\rho(e_i^2)$ is a function of residuals:

$$\rho(e_i^2) = min\left(\frac{e_i^2}{\sigma^2}, \lambda_3(r-d)\right) \tag{6.2}$$



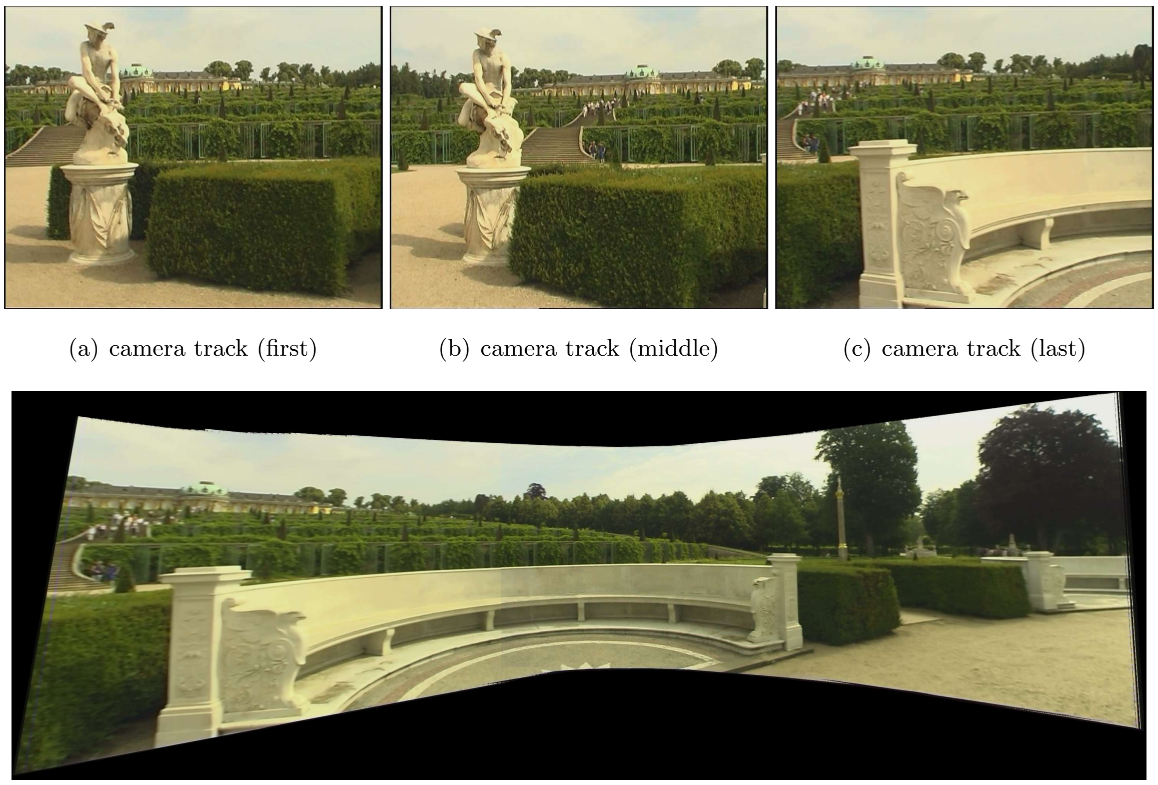
(a) Key frame curve “Room3D”

Figure 6.6: Keyframe curve of feature tracker using GRIC-score with turning points

The parameters are defined as follows: d is the dimension of the selected motion model (H has the dimension 2, whereas F has 3 dimensions), r is the dimension of the data (i.e. equal to 4 for two views), k is the number of the estimated model parameters (7 for F and 8 for H), n is the number of tracked features, σ is the standard deviation of the error on each coordinate and e_i is the distance between a feature point transfered through H and the corresponding point in the target image, or the Euclidian distance between the epipolar line of a feature point and its corresponding point in the target image, dependent on the selected model. λ_1, λ_2, and λ_3 are tuning parameters.

Initializing the first frame of the sequence as keyframe and proceeding frame by frame, the next keyframe is selected, if the GRIC value of the motion model F is below the GRIC value of H, i.e. a 2D motion model is no longer an accurate representation of the camera motion with respect to the 3D structure.

Having the set of keyframes of the considered sequence, we have to find a decision criterion for segmenting the sequence. For this, we draw the keyframe-curve over all frames of the sequence. Figure 6.6 shows the diagram for test sequence “Room3D”. Defining the criterion for the sequence segmentation, we need to take a look at the characteristic of this curve. We can see that in the first part of the sequence, where the camera pans, no keyframes are selected by the GRIC-score algorithm except the initial first frame of the sequence. When the camera movement changes to a track, keyframes are selected along this sub-sequence. In the third part, the camera motion turns back to pan and only the last frame is selected for completion. We now have to find the turning points of the keyframe-curve to define borders of the camera motion types by calculating the maximum of the second derivative



(a) camera track (first) (b) camera track (middle) (c) camera track (last)

(d) pan, frames 249-400

Figure 6.7: Video segments using GRIC, “Castle”

of this curve. It can be seen in Fig. 6.6 that the detected turning points segment the video sequence regarding to the camera motion types. For sequence “Room3D” the sub-sequences are slightly different to the GME-based sequence segmentation as described above (first part frame 1-70, second 71-120, third 121-160). The segmentation of the second test sequence “Castle” using the GRIC-score is more different to the first GME-based approach. Figure 6.7 shows the results. The performance of both algorithms is evaluated within a camera-motion constraint video codec. We will see the impact of using different kinds of video codecs and the role of the sub-sequence segmentation.

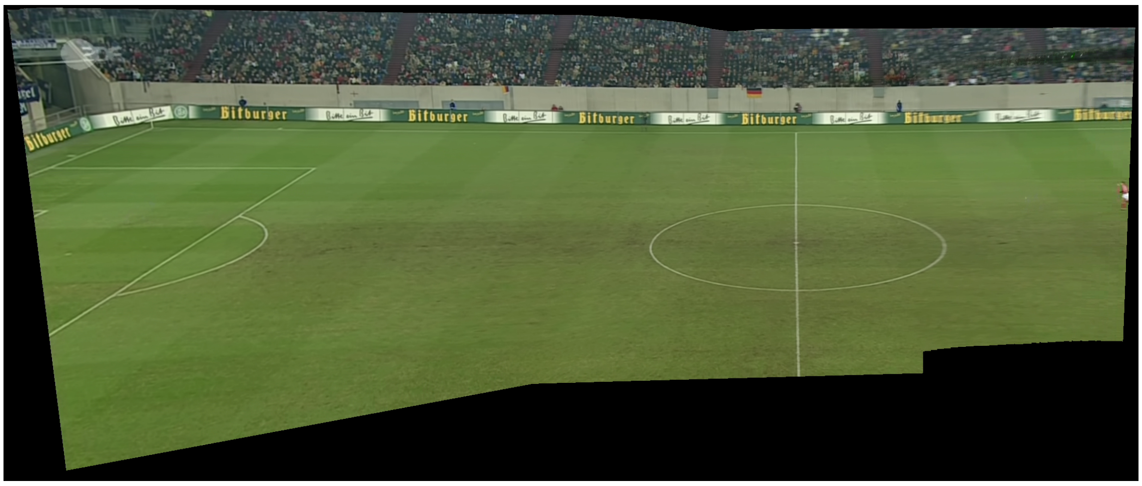
6.4 The content-adaptive (CAVC) and camera-constraint video coding system (CMCVC) using OBVC and H.264/AVC

This section introduces the object-based coding scheme used and summarizes the complete content-based video coding system.

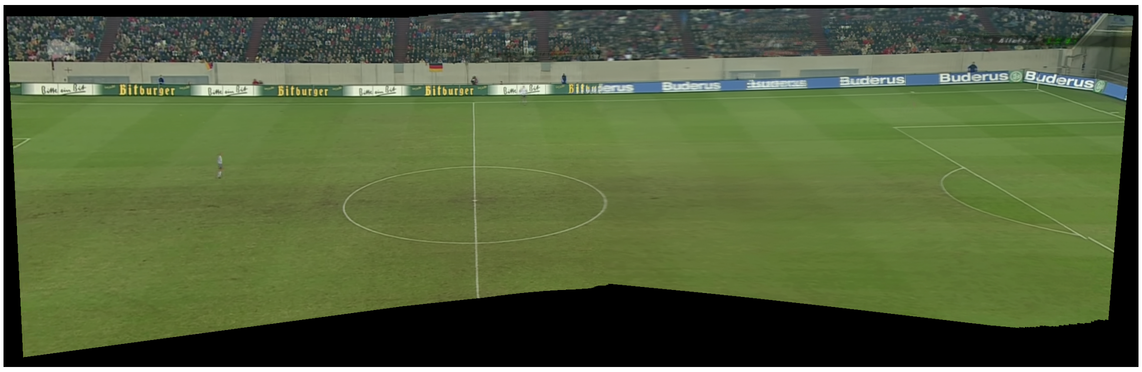
6.4.1 Content-adaptive Video Codec

6.4.1.1 Object-based Coding Approach

The object-based video codec (OBVC), which has been presented in previous chapters and in [26] and [41], combines the advantages of the object-based coding idea using background mosaics and the excellent coding performance of the H.264/AVC. As pre-processing, a background Sprite is generated which contains all the background information of the sequence. By applying a blending technique, nearly all foreground objects can be removed from the background mosaic image. Figure 6.8 shows the background Sprites for shot 1 and 3 of our considered test scene. The video sequence is then reconstructed from the Sprite and all the frames contain only background information. This background video sequence is used for an in-built foreground/background segmentation algorithm, which relies on a background subtraction technique (see Chapter 3) and some further algorithms.



(a) Mosaic (shot1)



(b) Mosaic (shot3)

Figure 6.8: Background Sprites of shot 1 and 3

Having the segmented video data, the background mosaic image, the foreground objects sequence, the foreground/background binary mask (which is needed at the decoder) and the motion parameters (which are not coded), the H.264/AVC is used to code the video

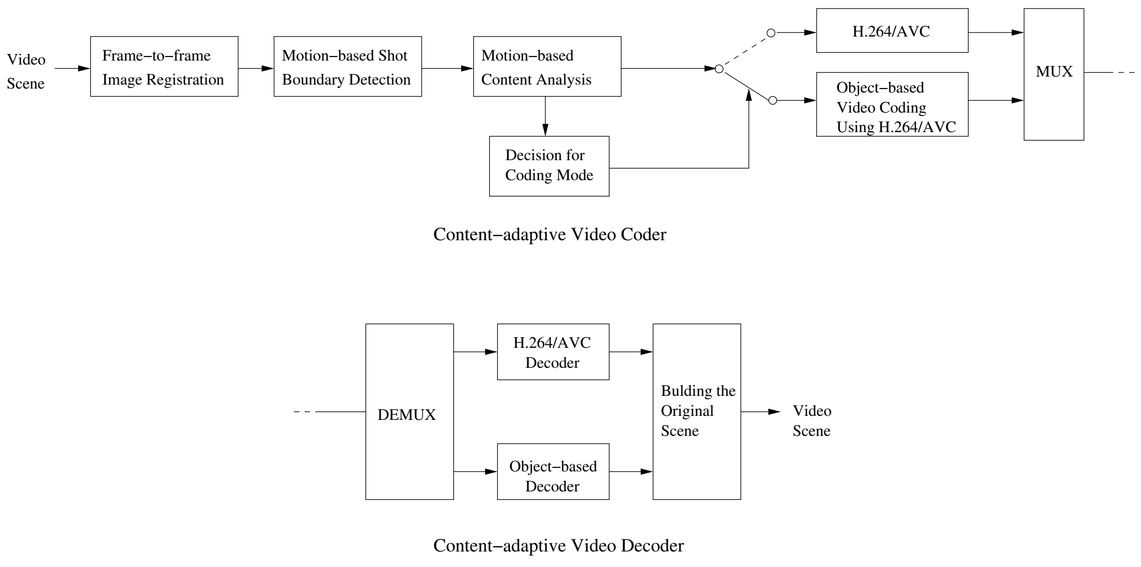

Video Scene
Frame−to−frame Image Registration
Motion−based Shot Boundary Detection
Motion−based Content Analysis
Decision for Coding Mode
H.264/AVC
Object−based Video Coding Using H.264/AVC
MUX
Content−adaptive Video Coder
DEMUX
H.264/AVC Decoder
Object−based Decoder
Bulding the Original Scene
Video Scene
Content−adaptive Video Decoder

Figure 6.9: Content-adaptive Video Codec

segments. At the decoder, the segments are merged together to the reconstructed video sequence.

6.4.1.2 The Content-based Video Coding System

All the techniques described are combined in the coding system shown in Fig. 6.9. The coding-mode decision relies only on the camera motion estimation. Two video codecs, object-based video coding (OBVC) and H.264/AVC, are considered for coding shots of a scene separately. The video data is then transmitted and decoded with the related video decoder. Afterwards, the scene is set together from the separated shots. In the next section, the first experimental results are presented. The results show the suitability of content-adaptive coding.

6.4.2 Camera motion-constraint Video Coding using Sprite Coding and H.264/AVC (CMCVC)

The described analysis algorithms are now embedded in a second video coding system. A simplified block chart of the system is depicted in Fig. 6.10. For the analysis part, both algorithms can be used as the pre-processing step. Having the borders calculated, the input sequence is segmented into a sub-sequence where the camera is on a fixed point and the movements are pan, tilt, zoom, rotation and a sub-sequence where the camera is tracked. Both labels go into the decider, which switches between both possible video codecs. The sub-sequences with a fixed camera (motion types: pan, tilt, zoom, rotation) are coded with a Sprite codec using H.264/AVC. The two test sequences considered contain no moving foreground objects. Therefore, no object segmentation is needed. We use a simplified version of the object-based video codec proposed in [41]. If the GRIC-based analysis is used

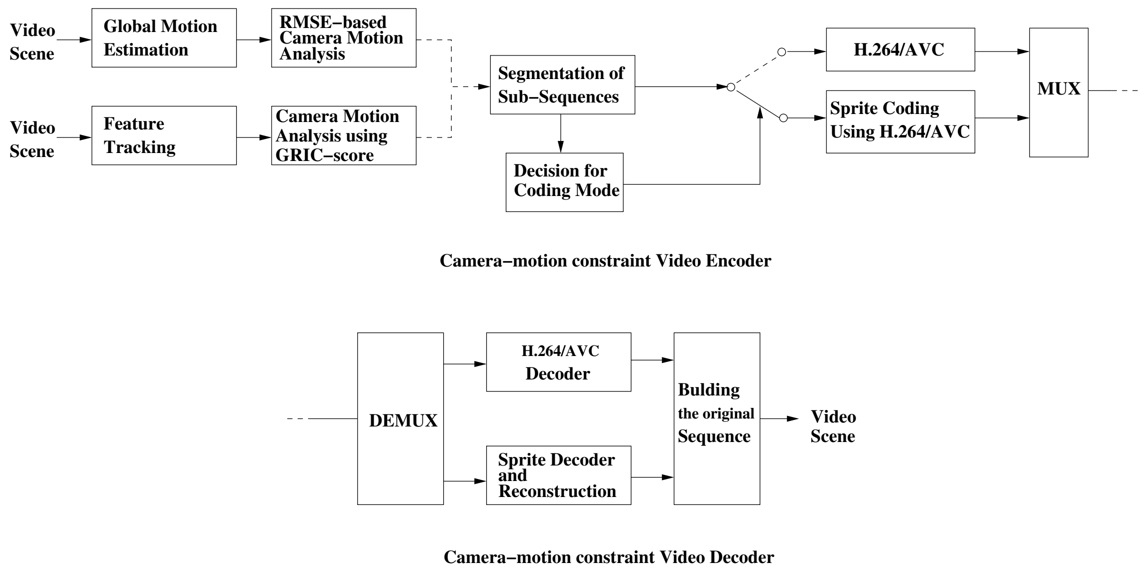

Video Scene
Global Motion Estimation
RMSE–based Camera Motion Analysis
Video Scene
Feature Tracking
Camera Motion Analysis using GRIC–score
Segmentation of Sub–Sequences
Decision for Coding Mode
H.264/AVC
Sprite Coding Using H.264/AVC
MUX
Camera–motion constraint Video Encoder
DEMUX
H.264/AVC Decoder
Sprite Decoder and Reconstruction
Bulding the original Sequence
Video Scene
Camera–motion constraint Video Decoder

Figure 6.10: Camera-motion constraint Video Codec (CMCVC)

an extra short-term global motion estimation is applied to set up the Sprite generation step. Otherwise, the GME-part can be left behind and the global motion parameters already calculated during the analysis part are used. This is the main advantage of the GME-based analysis algorithm. Then, a Sprite generation algorithm is used to store all the frames into one image. In this work, we use a single background Sprite. The background Sprite image is coded as an Intra-frame using the H.264/AVC. The long-term motion parameters which control the background Sprite generation and reconstruction are not coded. We assume that providing three byte per parameter is accurate enough for transmission. The 8-parameter perspective motion model is used. That means, we have 24 bytes per frame as side information. Figure 6.11 illustrates this simplified Sprite codec using H.264/AVC.

If the camera motion of the sub-sequence is track it is treated like a general video sequence, because is not possible to generate a background Sprite for this type of camera motion. For this, we use the common H.264/AVC video codec.

6.5 Experimental results

6.5.1 CAVC

The experiments are examined with the football test sequence “Allstars” (704x576 pixels, 641 frames, 25 fps). The sequence is coded using the proposed content-adaptive OBVC and only with H.264/AVC. For H.264/AVC, we use the latest examined prediction scheme, hierarchical B-frames, with a GOP of 15 frames. These settings are fixed for the content-adaptive codec and the use of H.264/AVC. Shot 1 (250 frames) and 3 (316 frames) can be coded using the OBVC. Figure 6.12 and 6.10 show rate-distortion curves for these two sub-sequences. It can be seen that especially for shot 1, the OBVC achieves a much higher

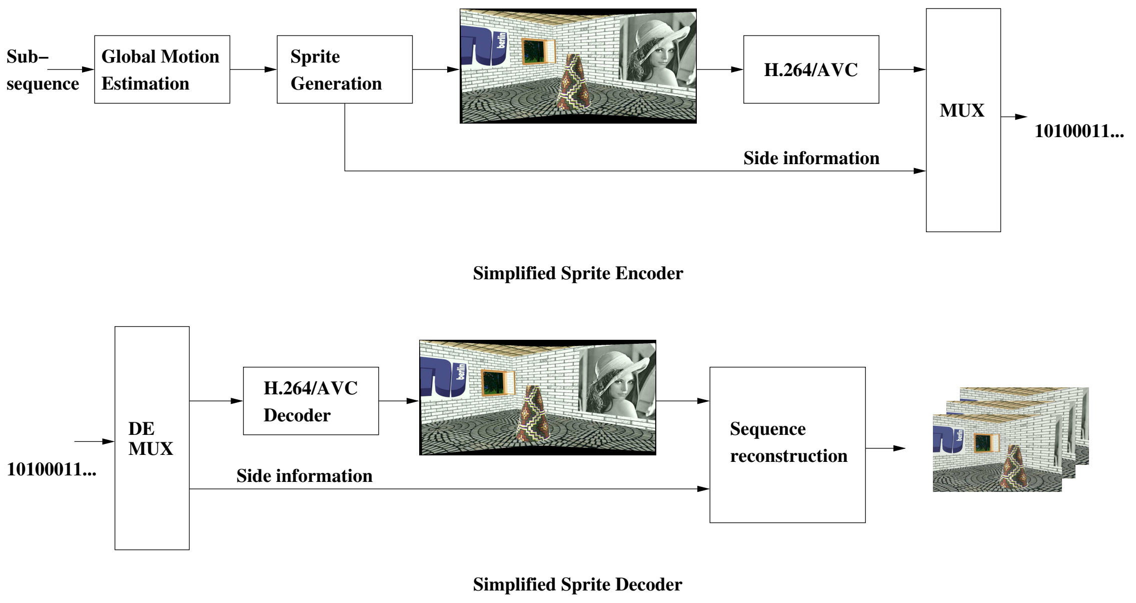
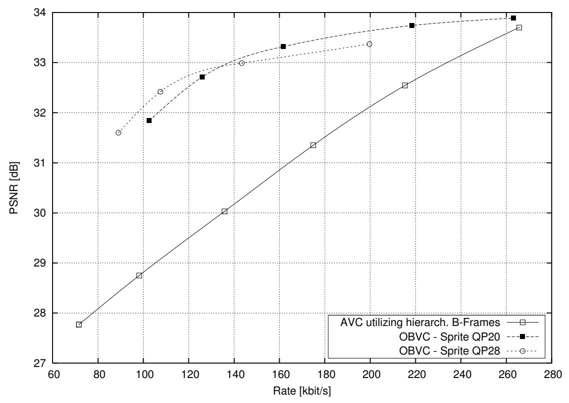


Figure 6.11: Simplified Sprite codec using H.264/AVC

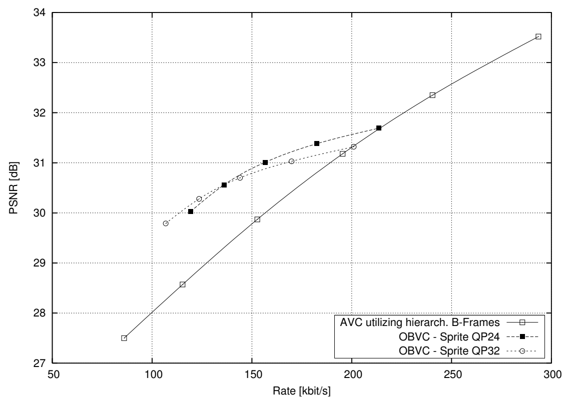
coding performance in comparison to H.264/AVC. The difference of the PSNR-values is up to 3 dB and higher. For shot 3, there is also an imrovement of the coding performance (up to 2 dB). However, the coding limit is reached here earlier because of the presence of more foreground objects in the scene. The shot 2 is coded with H.264/AVC for both cases, so there is no benefit using our approach compared to H.264/AVC. Figure 6.12 shows the rate-distortion curve for the whole scene. Due to the coding gain of shot 1 and 3 of the OBVC, the content-based video coding system outperforms H.264/AVC over a bit rate range of up to 250 kbits/s. We achieve gains of up to 2 dB in quality for the same bit rates, or save more than 30% of the bit rate for the same quality. This can be stretched by providing more bits for shot 2 (last point of the curve). It can be seen that despite the limit of shot 3, a coding gain can be held in that range for the whole scene. Figure 6.13 shows parts of frames taken from the decoded videos from shot 1. It can be seen that the subjective quality as well as the objective quality of the OBVC-coded video is higher than for that coded with H.264/AVC.

6.5.2 CMCVC

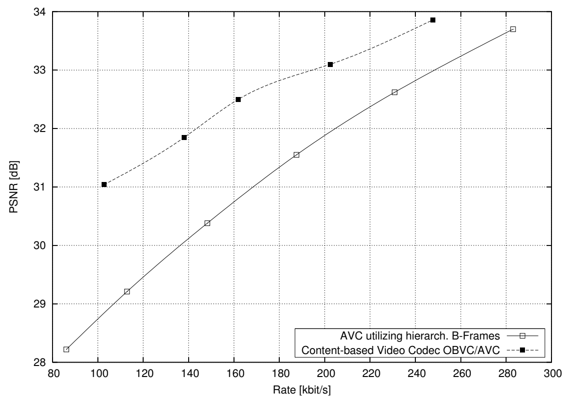
We consider two test sequences for the experimental evaluation. The first sequence is a synthetic video called “Room3D” (720x576 pixel, 25 fps, 160 frames, progressive) with camera pan, track and pan again. The second test sequence is a real video captured with a hand camera. It is called “Castle” (720x576 pixel, 25 fps, 400 frames, progressive). Here we have a typical “home-made” scenario where the camera is tracked in the first part and panned in the second part of the sequence. Of course, this camera movement can occur on a large number of types of video content. The application for home videos is only an example. We code the sequences using our camera motion-constraint coding approach with the GME-based



(a) OBVC (shot 1)



(b) OBVC (shot 3)

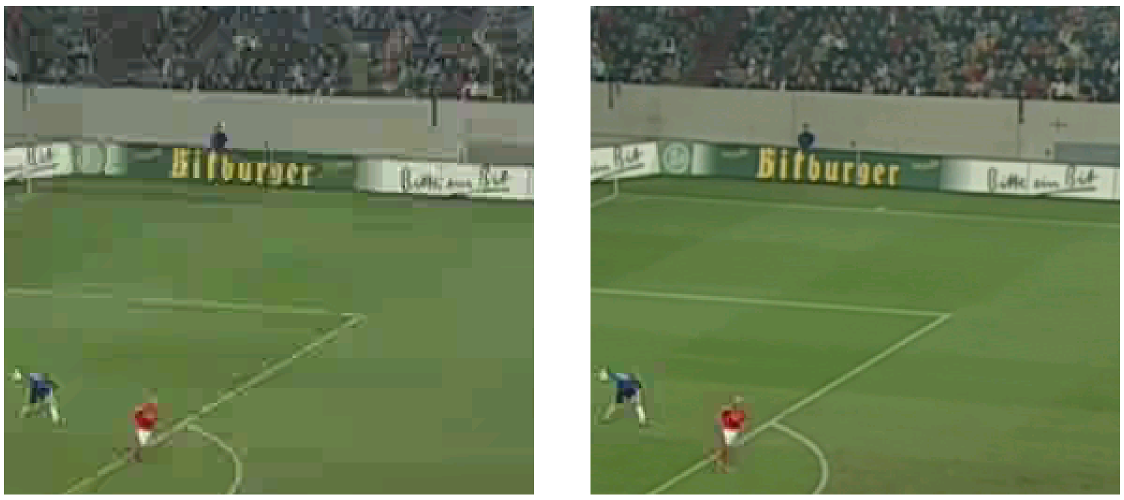


(c) CAVC (complete scene)

Figure 6.12: Coding results OBVC of the selected shots of the scene and CAVC for the complete scene, sequence “Allstars”

analysis (CMCVC - GME) and feature tracking and the GRIC-score (CMCVC - GRIC). We compare both approaches to each other and against common H.264/AVC objectively using the PSNR-value for the objective quality. A GOP-size of 15 frames is used for common H.264/AVC. For the prediction structure, we use the latest approved hierarchical B-frames. Additionally, we apply all new features inlcuding CABAC to have best coding performance for both compared codecs. Rate-distortion curves are shown in Fig. 6.14 (a) (“Room3D”) and (b) (“Castle”). It can be seen that we achieve a higher coding gain over a large bit rate range for the first test sequence. We achieve bit rate savings up to 50% against common H.264/AVC with both approaches. However, for this sequence, the GME-based algorithm slightly outperforms the GRIC-based approach. For the second test sequence, we increase the coding performance in comparison to common H.264/AVC in the lower bit rate range. The last point of the H.264/AVC-curve is the lowest coding limit for this sequence. We achieve up to 20% bit rate savings. Furthermore, we can extend the bit rate range in the lower direction using the CMCVC. The GRIC-based approach is here much better than the GME-based CMCVC.

Overall we can say that it is possible to increase the coding performance of common



(a) PSNR=31.35 dB, R=174.88 kbits/s (H.264) (b) PSNR=33.32 dB, R=161.64 kbits/s (OBVC)

Figure 6.13: Comparison of decoded frames (parts)

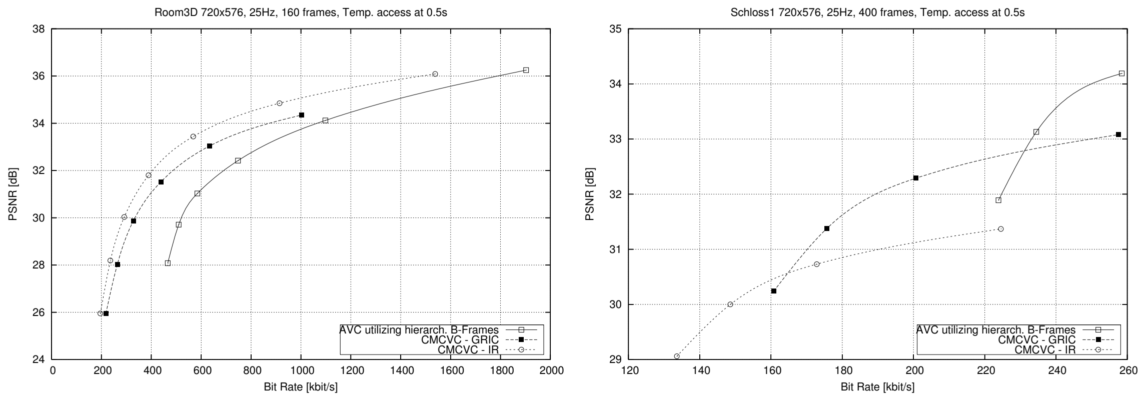
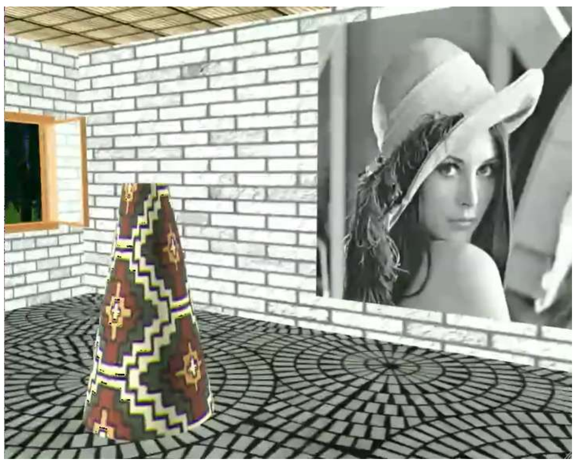
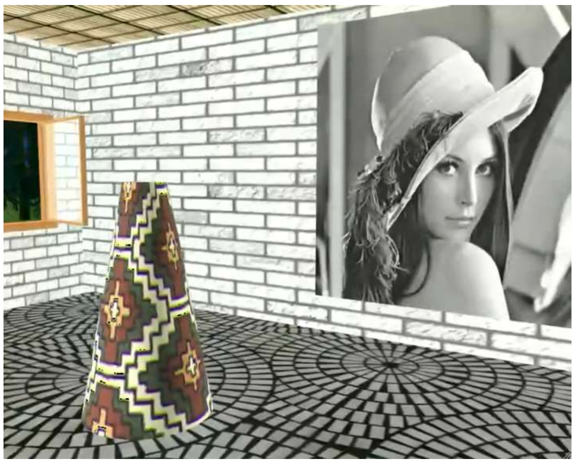


Figure 6.14: Rate-distortion curves comparing the CMCVC and H.264/AVC, “Room3D”, “Castle”

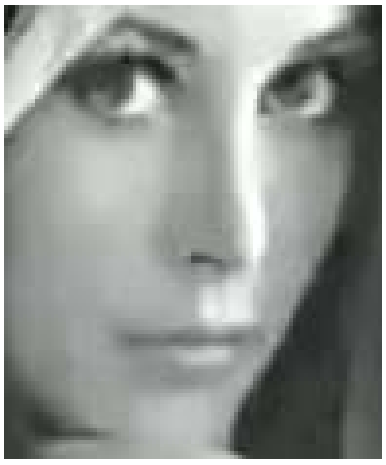
hybrid video codecs by applying a pre-processing step where the camera motion type is automatically recognized. Based on this, we add a Sprite coding mode for sequences where the camera is fix and the movements are pan, tilt, zoom, rotation, etc. We can interpret these first experimental results regarding two issues.

The first one is the ratio of the frames of the whole input sequence which are coded using common H.264/AVC mode and the frames which are coded using the Sprite coding mode (coding mode frame ratio (CMFR)). For the first sequence, the CMFR is 60:100 (groundtruth). That means, the sub-sequence where the camera tracks contains 60 frames (coded with H.264/AVC mode) and the two other sub-sequences where the camera pans contain 100 frames (coded with the Sprite mode). The automatically segmented sub-sequences lead to a CMFR of 48:112 = 0.43 for the GME-based algorithm and 50:110 = 0.45 for the GRIC-based approach. These results are very similar, however, the GME-based algorithm achieves a slightly higher coding gain, which means that less frames of the second


(a) H.264/AVC (common use), 465.98 kbits/s, frame 73


(b) CMCVC (H.264/AVC mode), 235.93 kbits/s (GME), frame 73

(c) Original (cut)


(d) H.264/AVC (cut)

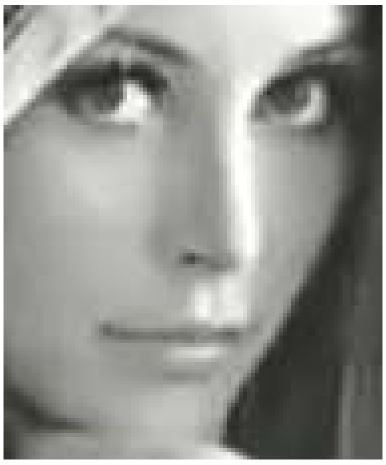
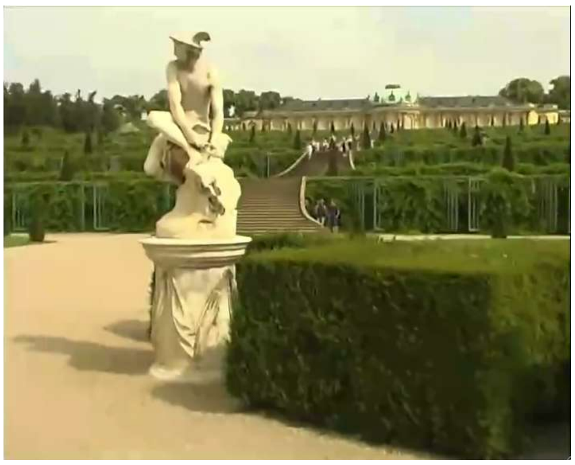
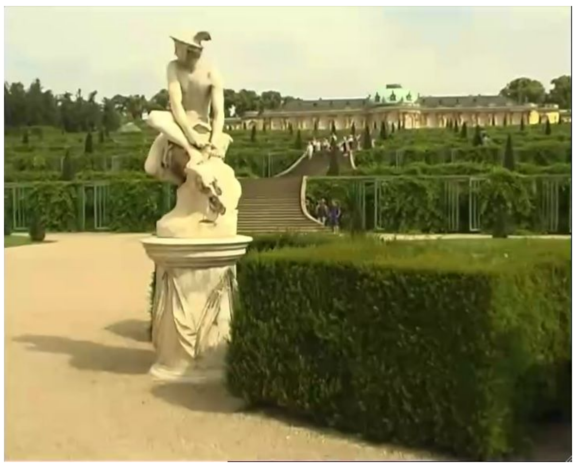

(e) CMCVC (cut)

Figure 6.15: Example for subjective comparison, CMCVC and H.264/AVC only, “Room3D” frame 73

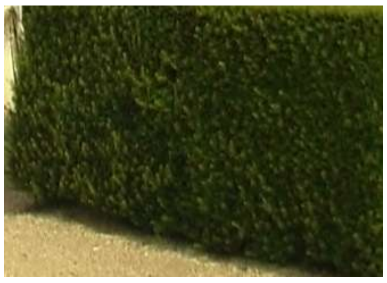
sub-sequence (track) are coded with the Sprite mode. For the test sequence “Castle”, no exact groundtruth is available because of the smooth crossover of the camera track and pan. It can be seen that here the GRIC-based approach segments the sequence better than the GME-based approach. Compared to sequence one, a coding gain can only be achieved for some parts of the bit rate range. The CMFR is 248:152 = 1.63 for GRIC-based and 198:202 = 0.98 for GME-based analysis. The GME-based algorithm includes too many frames from the camera track for the Sprite coding mode. This leads to distortions during the Sprite generation process because for the frames taken from the camera track sub-sequence, the assumed camera model is not valid anymore. The CMCVC using GRIC-based analysis produces a better result. If we compare the best CMCVC result for sequence one and two we can see that the coding gain against common H.264/AVC is not that high like for the first sequence for bit rate savings and coding gain over bit rate range.



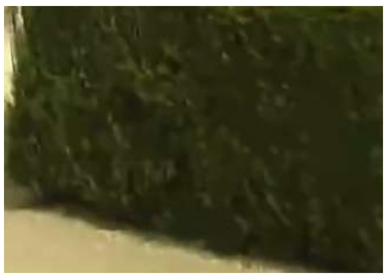
(a) H.264 (common use), 223.85 kbits/s, frame 100



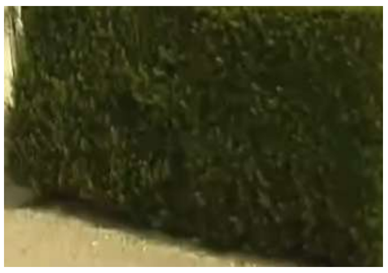
(b) CMCVC (H.264 mode), 172.84 kbits/s (GME), frame 100



(c) Original



(d) H.264/AVC



(e) CMCVC

Figure 6.16: Example for subjective comparison, CMCVC and H.264/AVC only, "Castle" frame 100

The reason is the much higher CMFR (1.63) for the second sequence.

The second issue for interpretation of the results is the video content itself. Test sequence "Room3D" has high-frequency textures, which is really hard to predict for a DCT-based motion-compensated codec. The use of a Sprite codec improves the objective as well as the subjective quality. Figure 6.15 and 6.16 show examples for the two test sequences. Example frames and close-ups are shown for H.264/AVC and CMCVC with H.264/AVC mode. It can be seen that due to the bit rate savings from the Sprite coding mode the sub-sequence part with the camera track can also be coded with higher subjective quality in comparison to H.264/AVC only. Thus, the sequences coded with CMCVC come along with bit rate savings together with increasing the subjective quality of the video.

6.6 Potential of the content-adaptive approach

We have seen that it is possible to achieve significant bit rate savings by the use of a pre-processing step analyzing the features of the input video, e.g. segmenting into shots, classifying the shots regarding its content and/or camera motion and selecting a video codec which fits to the video segment in an optimal way. In a first approach, we have used two video coding approaches. First, a content- and camera motion depending approach using background Sprites. It has been shown earlier in this work that it is possible to outperform the well-known hybrid video coding approach with bit rate savings of more than 50% and quality improvements of up to 2-3 dB (subjective quality is also increased). So it is obvious to take the next step and try to find out if the input video can be coded with the background Sprite-based approach or not. For that, a pre-analysis step is necessary to recognize the features of the input video. Two main aspects have been considered in this chapter which are necessary for a very good performance of the Sprite-based video coding approach, the kind of camera motion and the ratio of the foreground pixels and the background pixels. All aspects of the pre-analysis should be calculated with low-level algorithms and most of them global motion-based because global motion parameters have to be calculated anyway. Therefore, we have shown an improved shot boundary detection algorithm, a global motion-based content analysis step, and a first approach for segmenting a video sequence based on the camera motion. Based on this segmentation, we have selected either the Sprite-based approach if the current video sequence is valid for Sprite-based coding or use of the common hybrid approach in the opposite case. We have achieved high bit rate savings over a longer bit rate range. The relation between the bit rate saving by the use of a Sprite-based approach and the percentage of the frames of a sequence which are appropriate for coding with the Sprite-based approach is linear. This means that if 50% of an input sequence can be coded Sprite-based and a bit rate saving of 50% can be achieved and the remaining frames are coded with the hybrid codec, the overall bit rate saving is 25%. If the input sequence only contains 25%, it would be still 12.5% bit rate saving. This description gives a more general view of the potential of the proposed approach to use two (or more) different video codecs depending on the video content. We are aware that this issue was discussed earlier, however, not in this specific way, using a new Sprite-based approach with in-built object segmentation and H.264/AVC (OBVC) and use of common mode of H.264/AVC. Having this knowledge, it is possible to estimate the coding performance of sequences with about 70% of valid frames coded by the OBVC, like e.g. 15 min of a football broadcast evaluated in this chapter. This means that the overall bit rate saving can be calculated with the following equation:

$$rs_{all} = rs_{obvc} \cdot f_{obvc}, \tag{6.3}$$

where rs_{all} is the overall bit rate saving, rs_{obvc} is the bit rate saving of the part coded with the OBVC, and f_{obvc} is the percentage of frames of the input sequences coded with OBVC. If we assume that for our football broadcast example, we can achieve a mean bit rate saving of 30%, which is a realistic estimate, with the OBVC-coded parts, the overall bit rate savings is 30% $\cdot$ 0.7 = 21%. It is a very simple calculation, but it shows that it is really worth to proceed working this approach. At the moment, it is very time-consuming and the analysis algorithms still have problems in stability. However, with increasing computational power in the near future, this type of encoding of video content could become very attractive not only for offline applications.

6.7 Chapter Summary

In this chapter, we have proceeded the content-adaptive video coding idea considering the input video content in time. We have seen that the OBVC described in the previous chapters can signifcantly outperform the common hybrid video coding method, but only for an input video with certain features regarding camera motion and relation between foreground objects and background objects. To make this approach applicable, pre-analysis steps are needed to define whether a considered sequences of frames of the input video is valid for coding with the OBVC or not. The validity was the biggest challenge in the past when people worked on this type of coding approach and until now, this technique has not come to the market because of the limitation mentionend above and throughout this thesis. Compared to earlier work, a first step making this approach more applicable is the approach of setting up an in-built fully-automatic object segmentation for separating the video content before coding as described in Chapters 4 and 5. The next step is to analyze the input video time-based and content-based whether a certain amount of frames is valid for coding with OBVC or not. We have shown two approaches for segmenting the video temporally into shots which can be coded either with OBVC or H.264/AVC. We tried to stay at low-level analysis methods to keep the pre-processing as simple as possible. The first experimental evaluation has shown very promising results.

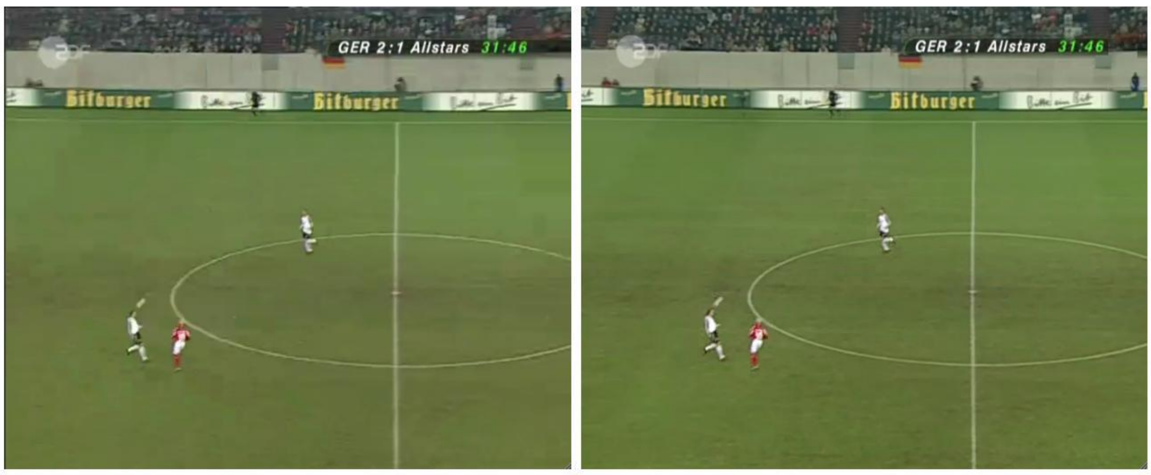
We conclude though that analyzing the video for selecting different video segments and coding these segments depending on their features with different coding approaches is worth a look in the future.

Chapter 7

Video Quality Metrics

7.1 Introduction

In all previous chapters, we used the PSNR for a quality measurement because it is widely used and it fits well to the rate-distortion theory. However, we have also seen that limitations of these coding approaches appear due to the limit in the objective quality measured with the PSNR of the decoded frames caused by the background Sprite generation and reconstruction. Random subjective tests have shown that despite lower PSNR values, the quality of the OBVC encoded video sequences are even better in comparison to common H.264/AVC encoded ones. Therefore, we briefly introduce alternative video quality metrics which try to fit more to the human perception and are probably more useful for the future.



(a) Frame 24, (H.264/AVC, PSNR = 36.01 dB, SSIM = 0.93)

(b) Frame 24, (OBVC, PSNR = 34.54 dB, SSIM = 0.94)

Figure 7.1: Comparison of a decoded frame between H.264/AVC and OBVC, sequence "Allstars"

7.2 Video Quality Metrics and OBVC

Until now, all experimental evaluation was examined using the PSNR for measuring the quality of the decoded sequences. As mentioned above, the reason for that is that this metric is widely used and accepted in the video coding community. Although there are studies from the early 90's until now about the PSNR (or MSE) and its lack of perceptual sensivity for the human visual system, e.g. [15], [9], [91]. Therefore, the research field "image quality assessment" tries to find an object metric which fits more to the human perception than the PSNR. Over the last decade, several alternative metrics have been developed, such as Video Quality Metric (VQM) [54], Sarnoff's Just Noticable Difference (JND), and SSIM (structural similarity). The latter has been published as the latest metric fitting more to the human visual system [88]. It has been shown that SSIM outperforms PSNR, UQI ([87]), and JND in terms of correlating with subjective quality evaluation. For our purpose, the SSIM is attractive because of the sensivity of blocking artifacts of common hybrid video codecs. Due to our PSNR limit using background Sprites, it is not possible to show objectively that we provide a subjectively high quality decoded frame while blocking artifacts still remain in the common hybrid encoded frames. Figure 7.1 shows an example.

7.3 Experimental results

Now, the performance of the OBVC described in the previous chapters is compared to common H.264/AVC using alternative visual quality metrics. For that, we consider the SSIM, which has been shown as the best quality metric in comparison to previous approaches. Because the SSIM is an image-related quality metric (as PSNR and MSE), we use a second metric, VQM (video quality metric), which also takes into account video features such as motion, etc. Four sequences are selected from previous experiments and new rate-distortion curves are generated using the two different quality metrcs. Figure 7.2 shows the results using the SSIM. Bit rate savings achieved are depicted in Fig. 7.2 (e) - (h). It can be seen that the OBVC outperforms common H.264/AVC in lower band width as resulted using the PSNR metric. Bit rate reductions up to 70% are possible at very low rates and around 20% at higher bit rates. The evaluation of the use of VQM as the quality metric is shown in Fig. 7.3.

To interpret these results, we measured the mean bit rate savings using the three quality metrics considered, i.e. PSNR, SSIM, and VQM. Table 7.1 shows the results. By calculating the mean rate savings over all four test sequences, it can be seen that the bit rate savings increase significantly using the SSIM and VQM metric. Furthermore, the second interesting value is the bit rate range where the OBVC outperforms H.264/AVC. Due to the background Sprite generation and reconstruction process there is the limitation in the objective quality measurement of the reconstructed frames. This means that we can reach only a fix value of

Table 7.1: Comparison of mean bit rate savings in % using various video quality metrics

Test videos	**PSNR**	**SSIM**	**VQM**
“Allstars”	41.5	47.1	46.7
“Biathlon”	14.5	25.3	25.1
“Entertainment”	18.2	32.2	9
“Race1”	15.37	7.7	31.25
mean of all sequ.	22.39	28.1	28.01

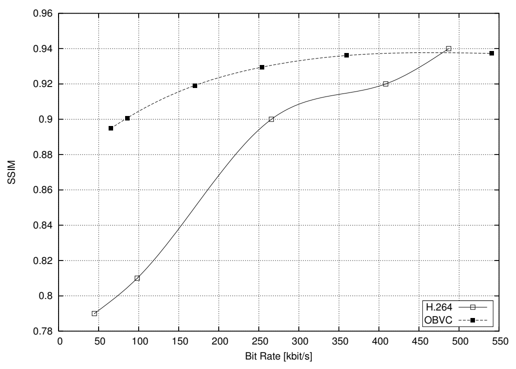
Table 7.2: bit rate ranges in kbits/s outperforming H.264/AVC starting from 0 kbits/s

Test videos	**PSNR**	**SSIM**	**VQM**
“Allstars”	260	460	400
“Biathlon”	125	240	220
“Entertainment”	170	260	210
“Race1”	125	300	460
mean of all sequ.	170	315	322.5

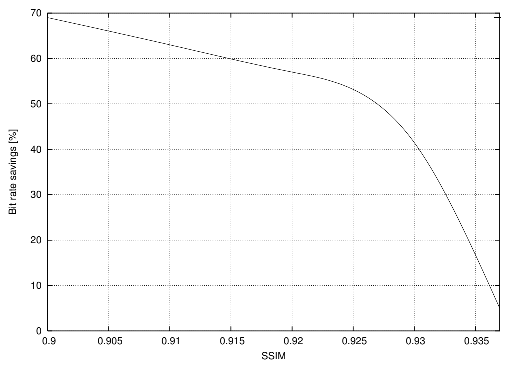
objective quality. By considering the outperforming bit rate range we can evaluate this fix value using SSIM and VQM as the quality metrics instead of PSNR. Table 7.2 shows the mean bit rate ranges of all considered test sequences. Using SSIM and VQM for objective quality it is possible to show that the OBVC is able to outperform common H.264/AVC for a higher bit rate range than using PSNR. Due to these improvements, the benefit of the OBVC can be more highlighted using the metrics that are designed to model the human visual system.

7.4 Chapter Summary

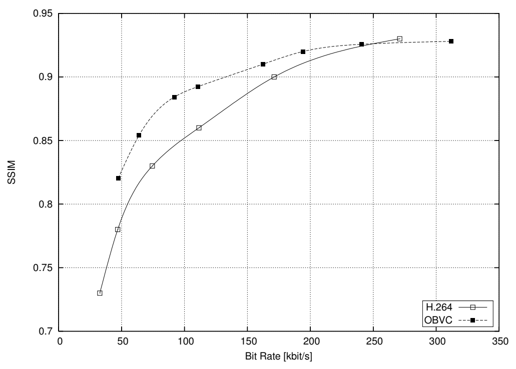
We shortly introduced alternative quality metrics to the widely used PSNR. The goal for these metrics is to come closer to the human perception to measure the visual quality of an image or video as seen by humans. Because of our modeling technique, the measurement using PSNR brings a limit in the PSNR quality. However, these limits due to the warping process while generating the background model do not cause distortions recognized by the human eye. Therefore, we have shown experiments comparing the OBVC developed in this thesis with common use of H.264/AVC with different kinds of quality metrics. It turns out that the OBVC is able to outperform common H.264/AVC in a wider bit rate range when perceptual-based quality metrics are used, such as SSIM. Introducing these alternative metrics sets up the motivation for a design of a quality assessed video codec, which will be optimized regarding to the human perception.



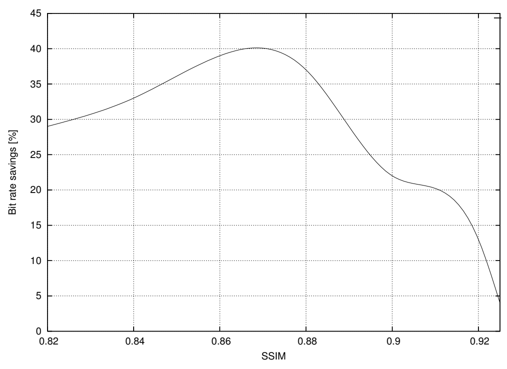
(a) “Allstars (4cif)”



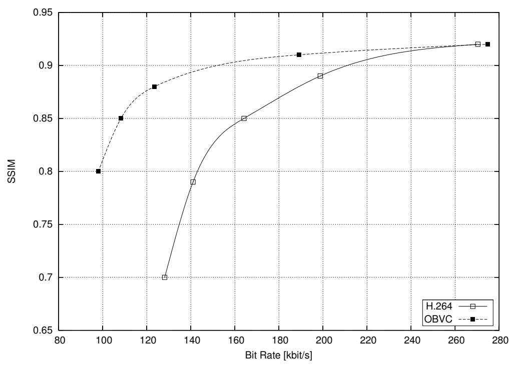
(b) Bit rate savings



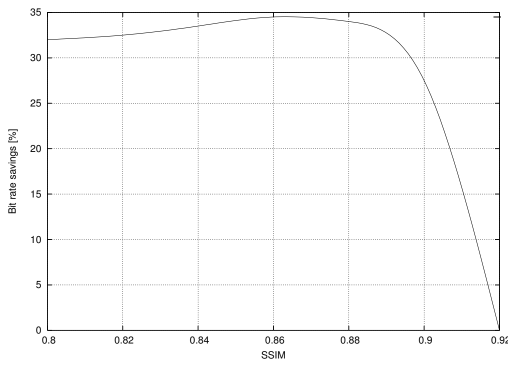
(c) “Biathlon”



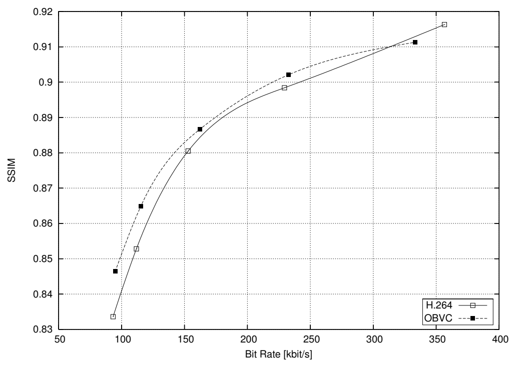
(d) Bit rate savings



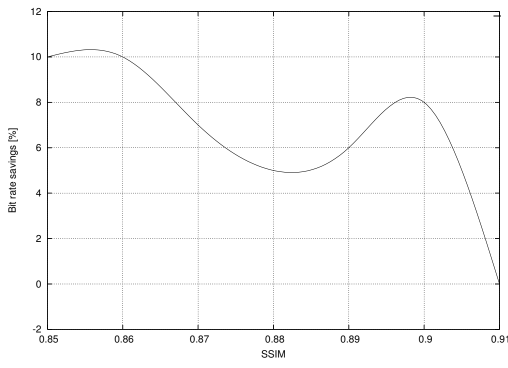
(e) “Entertainment”



(f) Bit rate savings

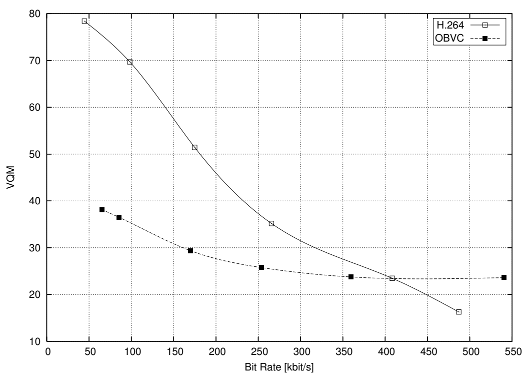


(g) “Race1”

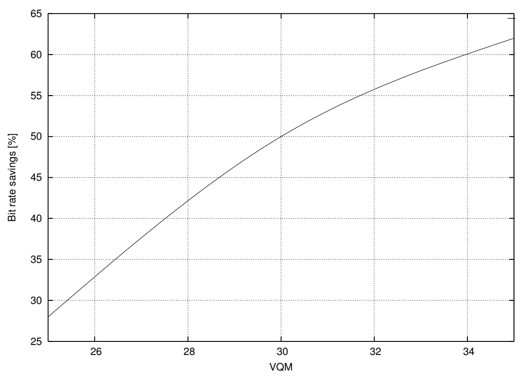


(h) Bit rate savings

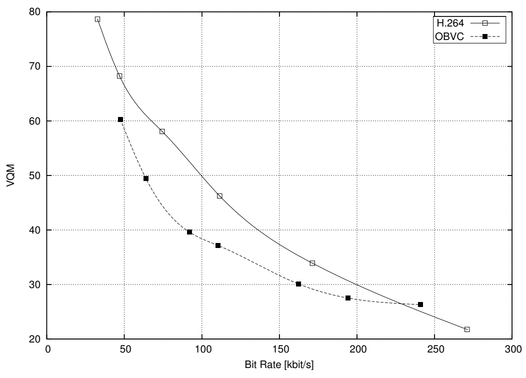
Figure 7.2: Rate-Distortion curves and bit rate savings in % of OBVC against H.264/AVC using SSIM metric



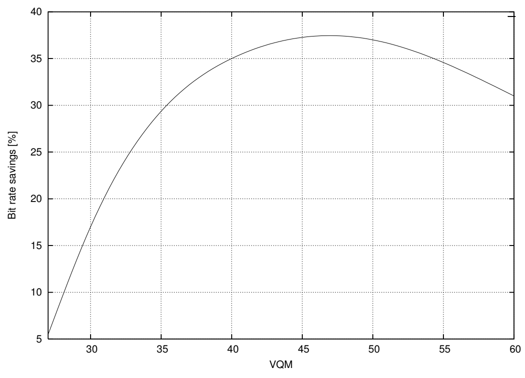
(a) “Allstars (4cif)”



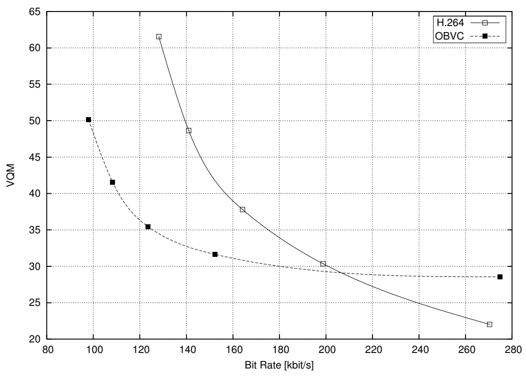
(b) Bit rate savings



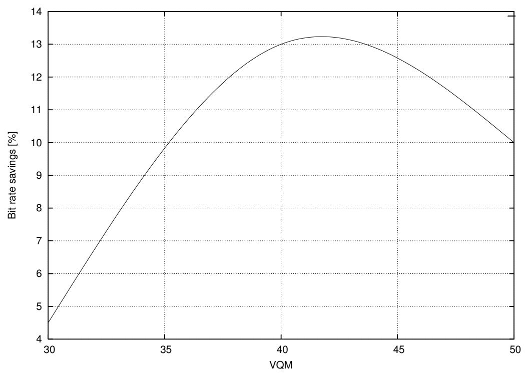
(c) “Biathlon”



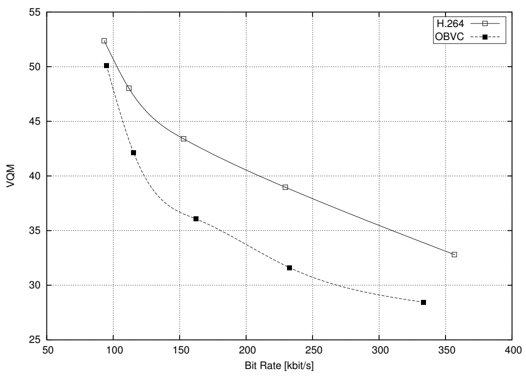
(d) Bit rate savings



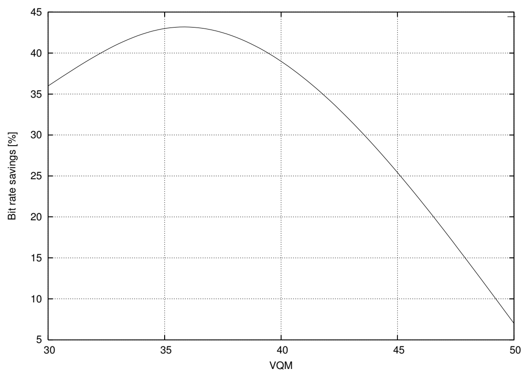
(e) “Entertainment”



(f) Bit rate savings



(g) “Race1”



(h) Bit rate savings

Figure 7.3: Rate-Distortion curves and bit rate savings in % of OBVC against H.264/AVC using VQM metric

Chapter 8

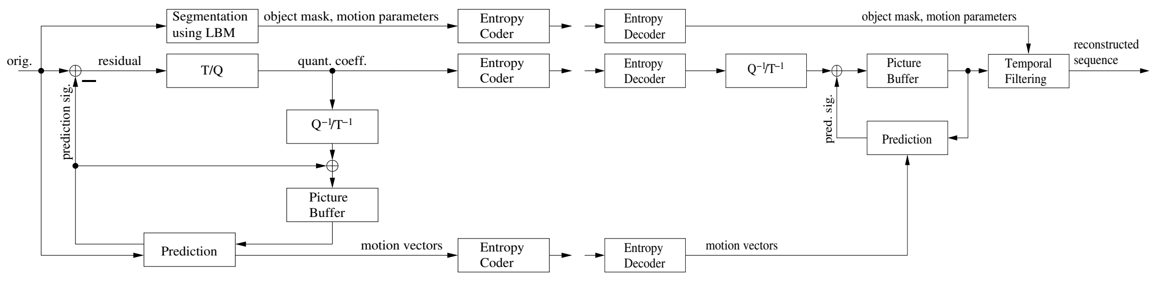
Video Coding using Global Motion Temporal Filtering

8.1 Introduction

In the previous chapters, the classical object-based video coding approach was determined. New and improved algorithms have been shown in every single part of the processing chain like global motion estimation, automatic object segmentation, and the way of encoding. Furthermore, due to the limitations of the object-based video coding using background Sprites, an approach has been developed and evaluated for a more general usability combining the OBVC and a common use of H.264/AVC. Having shown the potential of the techniques used in the previous codecs, we now would like to develop tools which can be used in common hybrid video codecs. We've got inspiration concering this issue by earlier work from Smolic et al [73], [71], [72]. Our algorithm described in Chapter 2 3 called Local Background Sprite is the key technique for the method described next. We apply this technique for global motion temporal filtering (GMTF). First, GMTF is used as a post-processing step to enhance the quality of coded video data. Second, considering perceptual-based quality metrics, a video coding scheme is designed using GMTF, which will be optimized regarding the human perception.

8.2 Global Motion Temporal Filtering for Post-processing

In our proposed Local Background Sprite algorithm, a mapping of content from many frames in a scene is performed for each individual frame for background construction. In other words, global motion estimation (registration) is performed from many adjacent frames into the frame where the background needs to be reconstructed. No backward mapping is required. Thus, our background Sprites are local and there are as many individual Sprites generated as frames exist in a sequence. This will result in a more precise background reconstruction compared to conventional global Sprites.



(a) Encoder (b) Decoder

Figure 8.1: Coding scheme using GMTF post-processing

Beside others, one application comes inside the method. When a Local Background Sprite is generated, a number of frames are aligned to the reference frame and for each pixel position of the reference frame, a row of pixel candidates is evaluated. Appying a filter, here the median, to the pixel row a final value of each pixel position of the reference frame is obtained. In other words, we apply a temporal filtering on the reference frame. That means the new background model frame is also free of noise due to this temporal filtering. We now tackle the blocking artifacts problem in common hybrid video codecs. The idea is to apply the Local Background Sprite generation algorithm on each frame at the decoder. If the video was transmitted at a low bit rate, e.g. in internet video portals, the quality of the decoded video is improved due to the post-processing step described.

8.2.1 Theoretical Consideration of GMTF deblocking

Blocking artifacts after encoding and decoding are coding noise. We can use the temporal mean filtering idea for noise reduction.

It is assumed that a number of distorted versions Y from an original image X are available after registration using global motion estimation. The local Sprite approach essentially identifies and registers these noisy versions. Consider the pixel registered value $y_k(m,n)$ of the kth frame as the sum of the original pixel $x(m,n)$ and a value from the noise signal $n_k(m,n)$:

$$y_k(m,n) = x(m,n) + n_k(m,n) \tag{8.1}$$

The mean value over all noisy versions $y_k(m,n)$ is:

$$y(m,n) = \frac{1}{N}\sum_{k=1}^{N} y_k(m,n) = x(m,n) + \underbrace{\frac{1}{N}\sum_{k=1}^{N} n_k(m,n)}_{r(m,n)}\,. \tag{8.2}$$

Uncorrelated white coding noise is assumed with the variance σ_n^2 and the autocorrelation matrix:

$$R_{nn} = \begin{pmatrix} \sigma_n^2 & 0 & \dots \\ 0 & \sigma_n^2 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{8.3}$$

We now show that the variance of the noise is reduced by the factor N (number of registered frames). The mean noise signal is $r(m, n)$. The variance can be calculated as:

$$\sigma_r^2 = E[R^2(m,n)] = \frac{1}{N^2}\sum_{i=1}^{N}\sigma_n^2 = \frac{\sigma_n^2}{N} \tag{8.4}$$

Thus, the variance of the coding noise has been reduced by the factor N by averaging pixel values of N noisy versions. Having this result we can turn to our deblocking problem in a codec environment. If we are able to apply a noise reduction using the GMTF approach we are able to increase coding efficiency.

One of the major problems in a common hybrid video codec are the blocking artifacts. For our theoretical estimation we treat these blocking artifacts as above as temporally independent white noise. For independent Gaussian sources, the rate-distortion function can be formulated as follows:

$$D_{\text{nf}}(R_{\text{nf}}) = 2^{-2R_{\text{nf}}}\sigma_x^2, \tag{8.5}$$

where $D_{\text{nf}}(R_{\text{nf}})$ is the distortion, R_{nf} is the bit rate, and σ_x^2 is the variance of the coded pixel amplitude for a single frame. We now apply a decoder noise reduction using N versions. With Equ. 8.4 the new rate-distortion function is:

$$D_{\text{f}}(R_{\text{f}}) = 2^{-2R_{\text{f}}}\frac{\sigma_x^2}{N}. \tag{8.6}$$

With $D_{\text{nf}}(R_{\text{nf}}) = D_{\text{f}}(R_{\text{f}}) = D$:

$$\begin{aligned} 2^{-2R_{\text{nf}}} &= 2^{-2R_{\text{f}}}\frac{1}{N} && (8.7) \\ \Rightarrow R_{\text{f}}(D,N) &= R_{\text{nf}}(D) - \frac{1}{2}log_2(N) && (8.8) \end{aligned}$$

We thus obtain a bit rate saving of $\frac{1}{2}log_2(N)$ per pixel by applying a noise reduction using averaging of frames. Equ. 8.8 is valid for $N \cdot D \leq \sigma_x^2$.

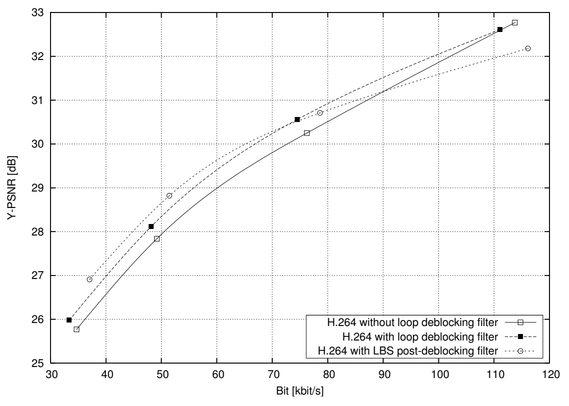
8.2.2 Coding environment using GMTF deblocking as post-processing step

The deblocking filter using GMTF is used as a post-processing step. Two aspects regarding the foreground objects have to be handled. First, the foreground objects information has to be added after the filtering process and second, the deblocking issue in foreground objects regions. For that, the following hybrid approach is considered. At the encoder, the input video signal is processed by an automatic object segmentation method using Local Background Modeling (LBM) ([29]). The output of this segmentation step is a binary mask for each frame, which defines foreground objects and background region along the video sequence. This binary mask sequence is encoded using the same binary mask encoder used in [41] and transmitted as side information. At the decoder, the binary mask is used to extract the foreground object regions from the decoded video sequence where the common deblocking filter used in H.264/AVC is applied. Then, GMTF is performed on the decoded video sequence. In the final reconstruction step, the filtered forground objects are mapped on the GMTF processed frames. As a result, pixels of the background regions are temporally filtered and the foreground object regions are filtered spatially using the common H.264/AVC deblocking filter. The block diagram of the described method is given in Fig. 8.1.

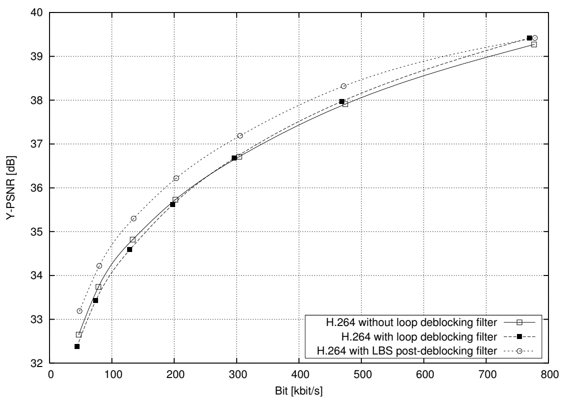
8.2.3 Experimental evaluation

Having this new approach, experiments are conducted to show the performance in comparison to the state-of-the-art deblocking techniques used in H.264/AVC. For that, we first have to define an objective metric for evaluation. In previous work in the field, PSNR was used to evaluate the performance of a deblocking filter, e.g. in the standard H.264/AVC in-loop filter [45].

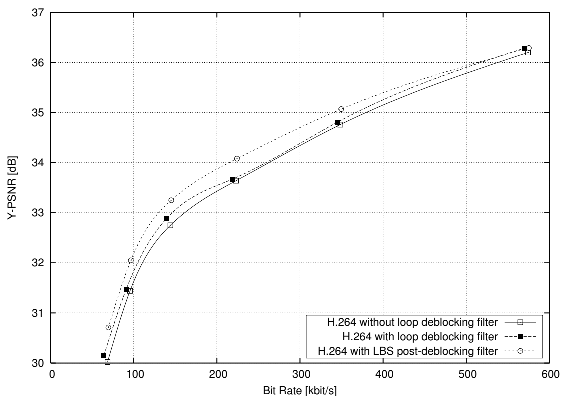
So we use PSNR to measure the amount of blocking artifacts in a decoded video frame. We choose four test sequences to show the performance of the deblocking using Local Background Sprites. The sequences “Biathlon” (352x288, 200 frames) and “Race1” (544x336, 100 frames) are used before and represent sport videos with large camera pans and zooms including single and multiple objects. Additionally, we take into account two sequences recorded from the BBC documentary “Planet Earth” called “Birds” (720x576, 100 frames) and “Desert” (720x400, 240 frames). Three different coding environments are considered for the experiments. Except the deblocking filter, all encoder settings are fixed to ensure a fair comparison. The JSVM v.9.1 is used as the reference H.264/AVC encoder with hierarchical B-frames prediction structure and GOP-size 15. The first setting is to encode the test videos at different QP’s without in-loop deblocking filter. The in-loop filter is turned on in the second term. The third coding environment is the scheme described above (Fig. 8.1)



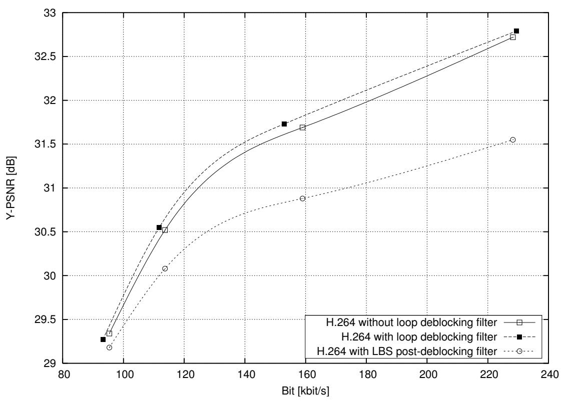
(a) “Biathlon”



(b) “Birds”



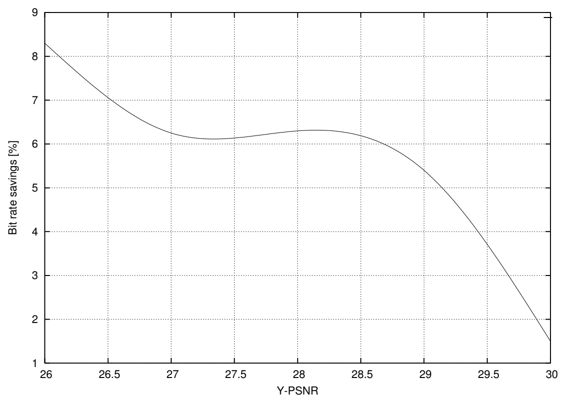
(c) “Desert”



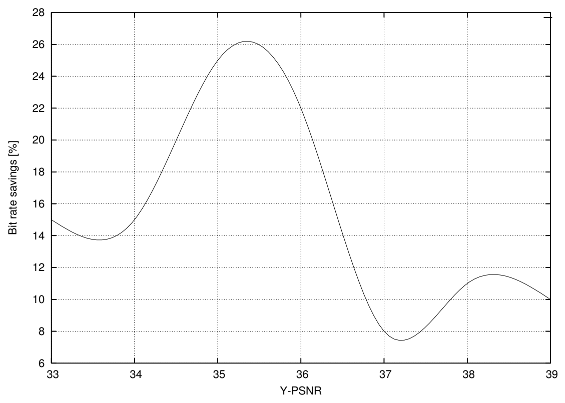
(d) “Race1”

Figure 8.2: Comparison of H.264/AVC in-loop deblocking filter and a post-processing deblocking filter using GMTF

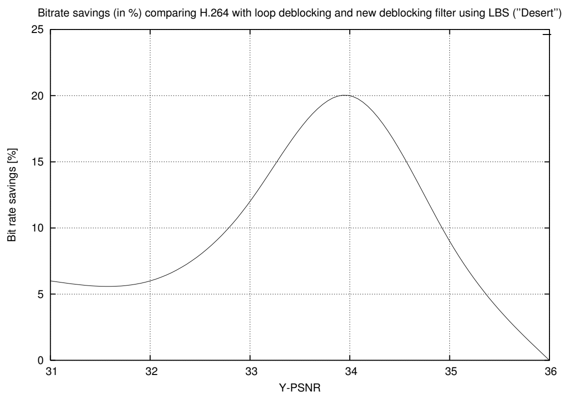
where encoding and decoding is done without in-loop deblocking filter. In theory, the post-processing deblocking using GMTF improves the quality of decoded video. To evaluate this, we have a look at the rate-distortion performance comparing the three coding settings mentioned. Figure 8.2 shows the rate-distortion curves for the four test sequences considered. It can be easily seen that the post-processing GMTF deblocking outperforms the common in-loop filter used in H.264/AVC significantly, except for test sequence “Race1” (we consider this problem later). It is obvious that the performance of the deblocking approach increases in the lower bit rate ranges when blocking artifacts appear. It starts at about 1Mbit/s in the TV-resolution and 500 kbit/s and 200 kbits/s at the test sequences with lower resolution, respectively. To emphasize this, bit rate saving curves are drawn comparing the use of common H.264/AVC in-loop deblocking filter and the GMTF deblocking filter in comparison to the use of no deblocking in-loop or as post-processing. The curves are depicted in Fig. 8.2. The savings achieved using the GMTF deblocking filter are immense at all sequences considered. Figure 8.3 shows the bit rate savings of the coding scheme described in Fig. 8.1 in comparison to the common use of H.264/AVC with in-loop deblocking filter. It is



(a) "Biathlon"



(b) "Birds"



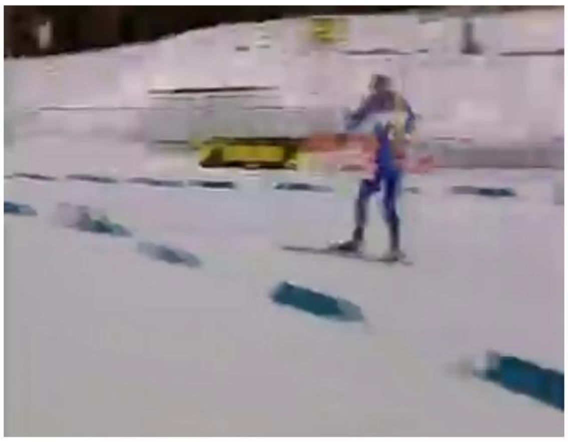
(c) "Desert"

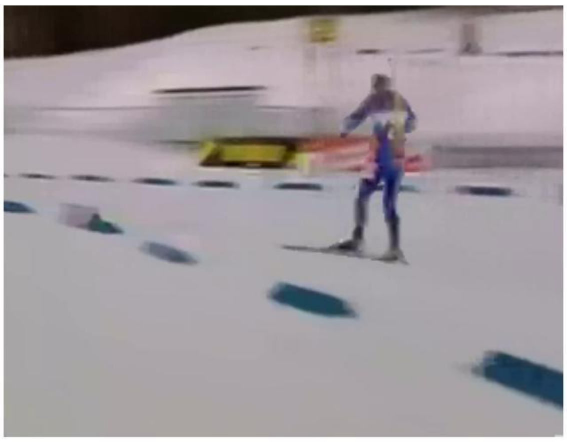
Figure 8.3: Bit rate savings of post-processing deblocking filter using GMTF compared to H.264/AVC in-loop deblocking filter

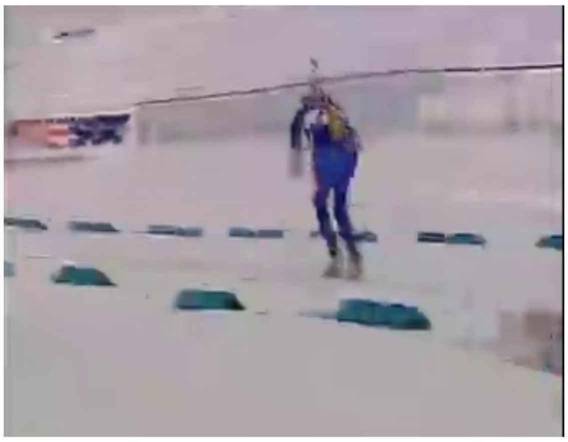
noticable that especially at the test sequences with TV-resolution, bit rate savings up to 37% can be achieved. Again, the performance gain of the new method increases with the resolution of the test video. For a subjective impression, Fig. 8.4 - 8.6 depict samples of the test sequences where the very good performance of the new deblocking scheme is shown. The images are taken from sequences coded with the same quatization parameter. The only difference is the use of the in-loop deblocking filter on one side and the GMTF deblocking on the other side.

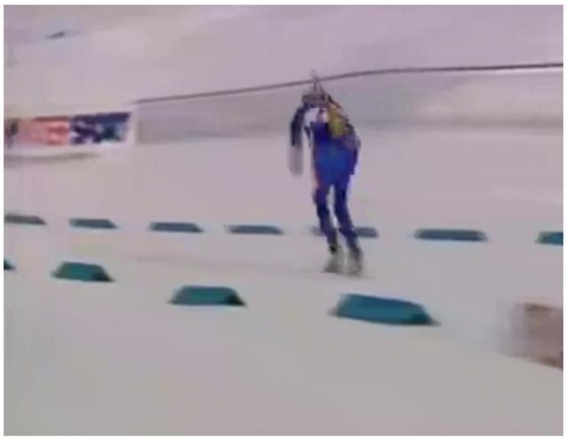
We have shown that our new deblocking method outperforms the state-of-the-art approach. It is possible to set up a coding scheme to transmit the input video with a lower bit rate and enhance the quality at the decoder using the post-processing step proposed. For three test sequences, the objective and subjective results are highly correlated. For "Race1", the GMTF brings no improvement according to the quality metric PSNR. We will show later that GMTF also improves the test sequences "Race", when a quality metric more reliable to the human visual system is used (SSIM). Another open issue is to find out how many frames are taken into account to build the Local Background Sprite for deblocking.

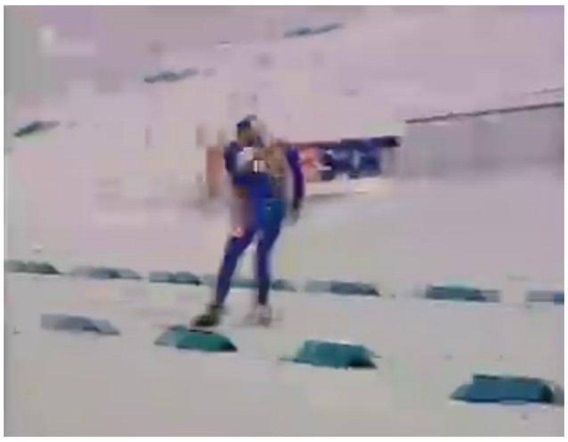
Until now, we have used a fixed number of frames (20) chosen empirically. This problem is tackled in the next section.


(a) Frame 11, (H.264/AVC)


(b) Frame 11, (H.264/AVC + GMTF)


(c) Frame 101, (H.264/AVC)


(d) Frame 101, (H.264/AVC + GMTF)


(e) Frame 130, (H.264/AVC)

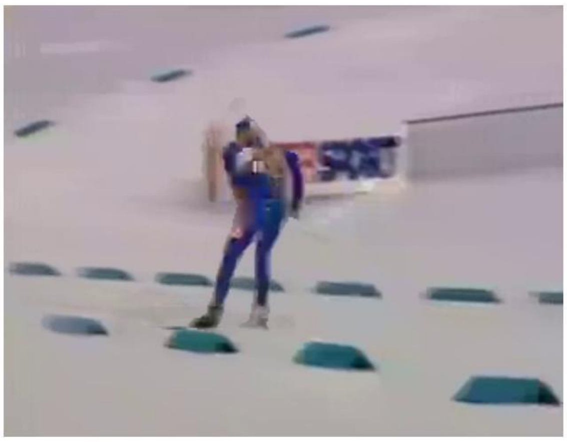
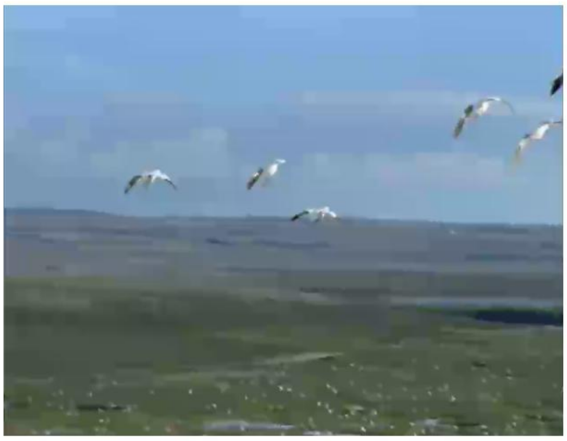
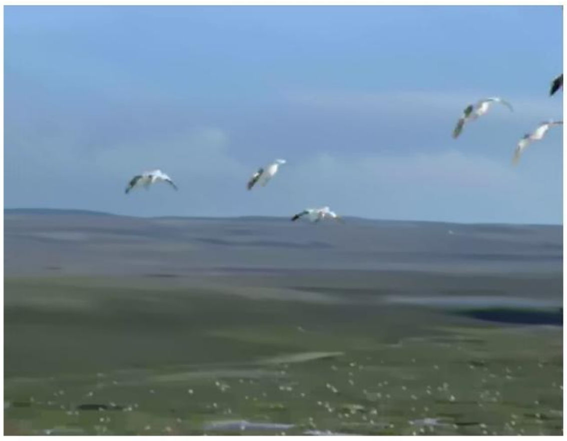
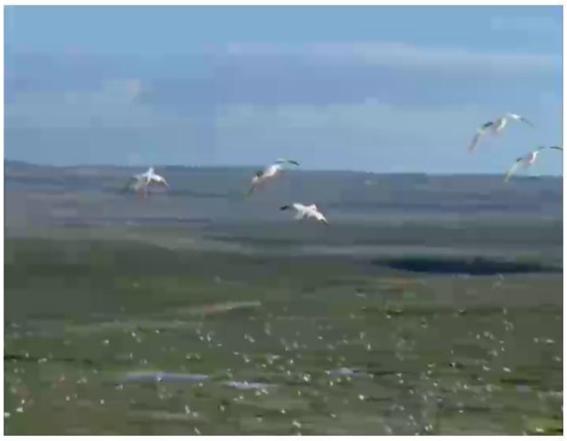
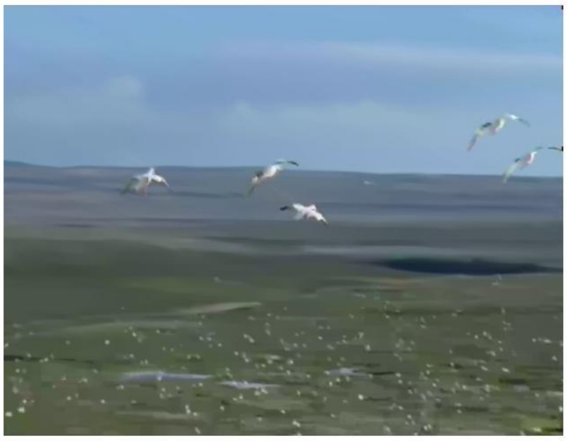

(f) Frame 130, (H.264/AVC + GMTF)

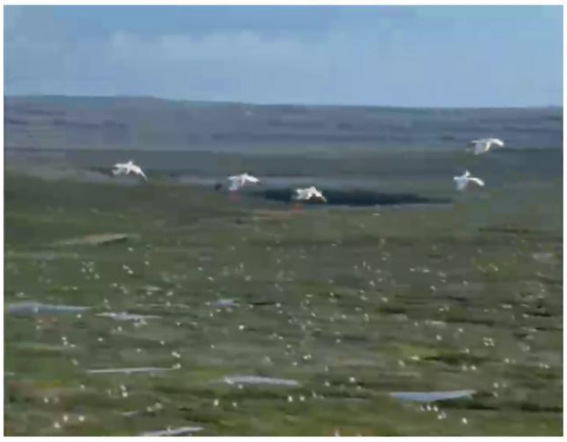
Figure 8.4: Comparison of commonly decoded frames and the deblocking version using Local Background Sprites, sequence "Biathlon"


(a) Frame 17, (H.264/AVC)


(b) Frame 17, (H.264/AVC + GMTF)


(c) Frame 33, (H.264/AVC)


(d) Frame 33, (H.264/AVC + GMTF)


(e) Frame 57, (H.264/AVC)

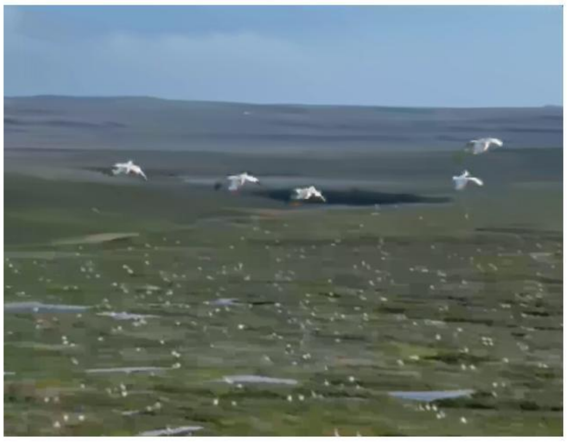
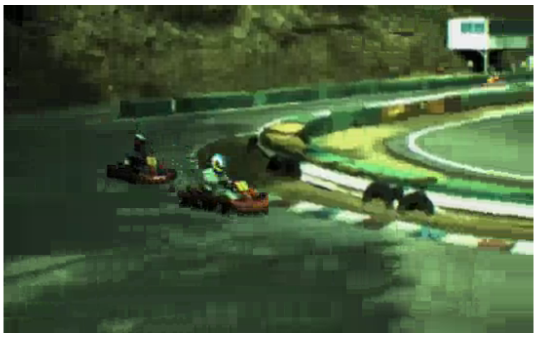
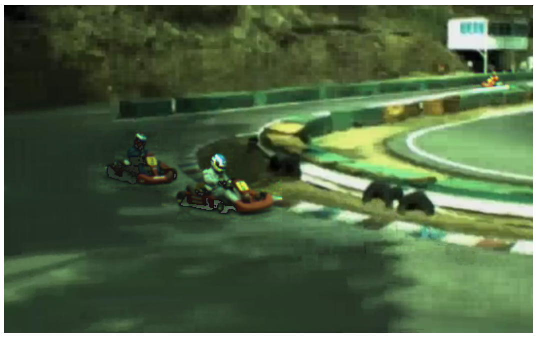
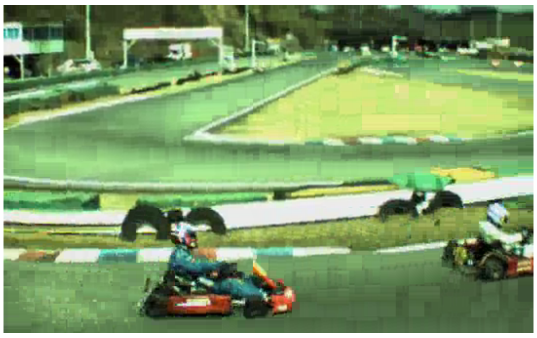
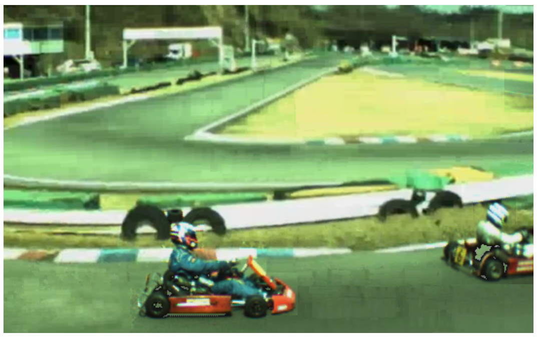

(f) Frame 57, (H.264/AVC + GMTF)

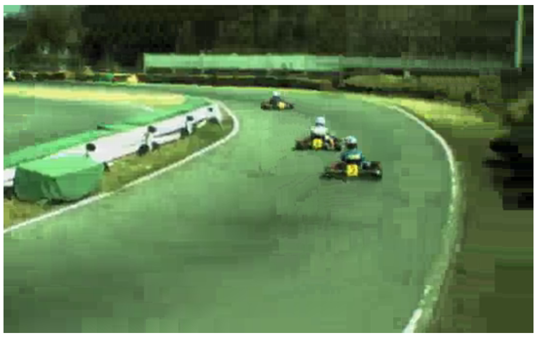
Figure 8.5: Comparison of commonly decoded frames and the deblocking version using Local Background Sprites, sequence “Birds”

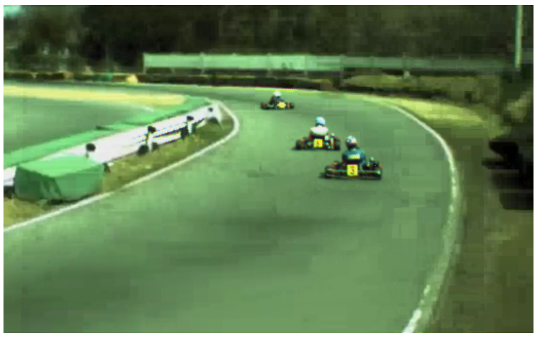

(a) Frame 11, (H.264/AVC)

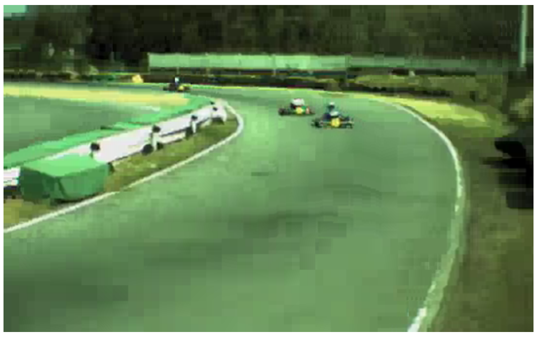

(b) Frame 11, (H.264/AVC + GMTF)


(c) Frame 36, (H.264/AVC)


(d) Frame 36, (H.264/AVC + GMTF)


(e) Frame 69, (H.264/AVC)


(f) Frame 69, (H.264/AVC + GMTF)


(g) Frame 92, (H.264/AVC)

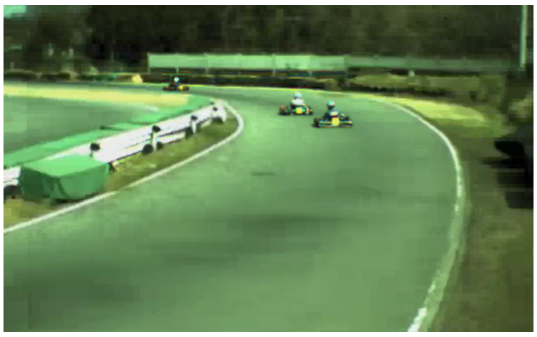

(h) Frame 92, (H.264/AVC + GMTF)

Figure 8.6: Comparison of commonly decoded frames and the deblocking version using Local Background Sprites, sequence "Race1"

8.3 Visual Quality assessed Video Coding using GMTF

It has been shown above that it is possible to save a lot of bits to transmit an encoded video applying a quality enhancement, that means removing blocking artifacts almost completely using the temporal filtering during a Local Background Sprite generation. The question now is how many frames used in the generation process of each reference frame achieves the optimal deblocking result. For that, a full codec approach has been developed to solve this problem. This method is described and evaluated in the next subsections.

8.3.1 Finding the optimal frame number for deblocking

In the previous section, we have used GMTF deblocking at the decoder. This means that there is only the decoded version of the video available. To find the optimal number of frames for deblocking, we have to design a new encoding scheme including a quality assessment loop. The block chart of the new encoder is shown in Fig. 8.9. The decoder is the same as in Fig. 8.1 except the aspect that additionally the number of frames used for generating the Local Background Sprite for each frame is transmitted as side information and delivered to the post-processing step. We start with an object segmentation algorithm to separate the foreground objects from the background. For that, we use the method described in chapter 3 ([29]). The next step is to find the optimal number of frames for the temporal deblocking using the Local Background Sprite. For that, we use the decoder including the deblocking filter to reconstruct the video as done at the receiver. Having the reconstructed video including temporal deblocking filtering in the background regions and merging the spatially deblocked foreground objects the quality is measured compared to the original after each frame used for generating the background Sprite. We use SSIM for the quality measure as reasonably outlined above. The reconstruction process is determined until a fix maximum frame number, here 40, is reached. The frame number which produces the best filtering quality for the reference frame considered is chosen and is encoded and transmitted as side information to the receiver. This means that each frame of the video sequence has an associated number of frames for generating the corresponding Local Background Sprite. We expect that the numbers of frames are highly correlated and slowly increase with a higher QP value. At the receiver, each frame at the decoded video can be temporally filtered and deblocked using the optimal amount of frames taken into account. This visual quality assessed codec (VQVC) is evaluated using the same test sequences used above. The results are outlined in the next section.

8.3.2 Theoretical Consideration

We repeat the consideration of the previous subsection to get the same starting point as above. Additionally, we provide an approach to model the global motion estimation process.

It is assumed that a number of distorted versions Y from an original image X are available. We consider the kth pixel value $y_k(m,n)$ of the kth version which is the sum of the original pixel $x(m,n)$ and a value from the white noise signal $n_k(m,n)$:

$$y_k(m,n) = x(m,n) + n_k(m,n) \tag{8.9}$$

We calculate the mean value using each candidate of pixel $y_k(m,n)$:

$$y(m,n) = \frac{1}{N}\sum_{k=1}^{N} y_k(m,n) = x(m,n) + \underbrace{\frac{1}{N}\sum_{k=1}^{N} n_k(m,n)}_{r(m,n)}\,. \tag{8.10}$$

White noise is assumed with the variance σ_n^2 and the autocorrelation matrix:

$$R_{nn} = \begin{pmatrix} \sigma_n^2 & 0 & \cdots \\ 0 & \sigma_n^2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{8.11}$$

We now show that the variance of the noise is reduced by the factor N (number of overlapping signals). The mean noise signal is $r(m,n)$. The variance can be calculated as:

$$\sigma_r^2 = E[R^2(m,n)] = \frac{1}{N^2}\sum_{i=1}^{N}\sigma_n^2 = \frac{\sigma_n^2}{N} \tag{8.12}$$

Thus, the variance of the noise has been reduced by the factor N. We now turn to our deblocking problem in a video. Assume that it is possible to observe N representations of y, i.e. corresponding quantized pixels from N frames of a video sequence. Averaging these quantized pixels y in temporal direction reduces the error variance by

$$E[e_f^2] = \frac{E[e^2]}{N}, \tag{8.13}$$

with N being the number of noisy versions x. This assumes that the versions of e in temporal direction are also not correlated. Our goal is to see whether it is possible to code all versions of x along the temporal direction with reduced bits/sample when afterwards temporal filtering is applied. Before showing that, we have to consider the generation of version of e in temporal direction. That means we have to align a number of frames for filtering one reference image. This is conducted using short-term and long-term motion estimation. We now find a way to model this. It is still assumed that we have the 2-dimensional Gaussian distributed memoryless signal $x = x_n$. We now try to estimate the current sample using the previous one. This is done with an additive operation, that is in an optimal estimation:

$$x_n(x,y) = x_{n-1}(x+t_x, y+t_y), \tag{8.14}$$

where t is the optimal estimation parameter. In practice, this operation does not match exactly due to the way of the estimation is calculated, right motion model, interpolation operation for sub-pixel case, etc. Therefore, an estimation error appears, which can be written for our theoretical case:

$$x_{n-1}(x+t_x+\Delta_x, x+t_y+\Delta_y) = x_n(x+\Delta_x, y+\Delta_y), \tag{8.15}$$

where Δ_x, Δ_y is the estimation error. Thus, the resulting error signal is:

$$e_n(x,y) = x_n(x,y) - x_n(x+\Delta_x, y+\Delta_y). \tag{8.16}$$

The term $x_n(x+\Delta_x, y+\Delta_y)$ can be approximated using the first Taylor expansion:

$$x_n(x+\Delta_x, y+\Delta_y) \approx x_n(x) + \nabla x_n^T \cdot \begin{pmatrix} \Delta_x \\ \Delta_y \end{pmatrix}. \tag{8.17}$$

With this, the error signal is:

$$\begin{aligned} e_n(x,y) &= x_n(x,y) - \left(x_n(x,y) + \frac{\partial x_n(x,y)}{\partial x} \cdot \Delta_x + \frac{\partial x_n(x,y)}{\partial y} \cdot \Delta_y \right) \\ e_n(x,y) &= -\frac{\partial x_n(x,y)}{\partial x} \cdot \Delta_x - \frac{\partial x_n(x,y)}{\partial y} \cdot \Delta_y \end{aligned} \tag{8.18}$$

We can now calculate the error variance using the expected value

$$\begin{aligned} \sigma_{e_n}^2 &= E[e_n^2] \\ &= E\left[\left(-\frac{\partial x_n(x,y)}{\partial x} \Delta_x - \frac{\partial x_n(x,y)}{\partial y} \Delta_y \right)^2 \right], \end{aligned} \tag{8.19}$$

with the assumption that $\frac{\partial x_n(x,y)}{\partial x}$ and Δ_x and $\frac{\partial x_n(x,y)}{\partial y}$ and Δ_y are uncorrelated and statistically indepented:

$$\begin{aligned} \sigma_{e_n}^2 &= E\left[\left(-\frac{\partial x_n(x,y)}{\partial x} \cdot \Delta_x \right)^2 \right] + \underbrace{2 \cdot E\left[\frac{\partial x_n(x,y)}{\partial x} \frac{\partial x_n(x,y)}{\partial y} \Delta_x \Delta_y \right]}_{=0} + E\left[\left(-\frac{\partial x_n(x,y)}{\partial y} \cdot \Delta_y \right)^2 \right] \\ \sigma_{e_n}^2 &= E[\Delta_x^2] \cdot E\left[\left(\frac{\partial x_n(x,y)}{\partial x} \right)^2 \right] + E[\Delta_y^2] \cdot E\left[\left(\frac{\partial x_n(x,y)}{\partial y} \right)^2 \right]. \end{aligned} \tag{8.20}$$

Now we approximate the derivative of x_n with the first numerical derivative in both directions:

$$\begin{aligned}\frac{\partial x_n(x,y)}{\partial x} &\approx x_n(x,y) - x_n(x-1,y)\\ \frac{\partial x_n(x,y)}{\partial y} &\approx x_n(x,y) - x_n(x,y-1),\end{aligned} \tag{8.21}$$

using this and the assumption $E[\Delta_x^2] = E[\Delta_y^2] = E[\Delta^2]$ we can plug the approximation in Equ. 8.20:

$$\begin{aligned}\sigma_{e_n}^2 &= \sigma_\Delta^2 \cdot \left\{ E\left[\left(\frac{\partial x_n(x,y)}{\partial x}\right)^2\right] + E\left[\left(\frac{\partial x_n(x,y)}{\partial y}\right)^2\right]\right\}\\ &= \sigma_\Delta^2 \cdot \{E[(x_n(x,y) - x_n(x-1,y))^2] + E[(x_n(x,y) - x_n(x,y-1))^2]\}\\ &= \sigma_\Delta^2 \cdot \{E[x_n^2 - 2x_n x_n(x-1,y) + x_n^2(x-1,y)] + E[x_n^2 - 2x_n x_n(x,y-1) + x_n^2(x,y-1)]\}\\ &= \sigma_\Delta^2 \cdot \{\sigma_x^2 - 2\underbrace{E[x_n(x,y)x_n(x-1,y)]}_{ACF(AR(1))=\sigma_x^2\cdot\alpha_1^{|1|}} + \sigma_x^2 + \sigma_x^2 - 2\underbrace{E[x_n(x,y)x_n(x,y-1)]}_{ACF(AR(1))=\sigma_x^2\cdot\alpha_2^{|1|}} + \sigma_x^2\}\\ &= \sigma_\Delta^2\sigma_x^2 \cdot (4 - 2(\alpha_1 + \alpha_2)\\ &= \underline{2\sigma_\Delta^2\sigma_x^2(2 - \alpha_1 - \alpha_2)}\end{aligned} \tag{8.22}$$

Equ. 8.22 is the prediction error variance due to the estimation of one signal from a previous version.

Knowing this we can derive a rate-distortion equation of reducing the noise using temporal filtering with the constraint of the estimation error variance. For our Gaussian distributed memoryless signal x_n, the D-R-function is:

$$\sigma_{e_{x_q}}^2 = 2^{-2R} \cdot \sigma_x^2. \tag{8.23}$$

In the aligning process for our temporal filtering, we calculate short-term parameters for the estimation between consecutive signals. Every aligned signal which represent a "new" version of the signal to be filtered is generated by applying long-term motion parameters according to the reference signal. The long-term motion parameters are calculated using accumulative multiplication of the short-term parameters. We assume that the model for the short-term estimation errors between two consecutive frames derived in Equ. 8.22 can serve for every estimation step. If these errors now are accumulated because of building the long-term motion parameters, the overall error caused by the motion estimation process increases. To model that, the motion estimation error increases with increasing number of frames taken into account for temporal filtering process, we basically sum the errors which occur due to short-term motion estimation. It is highly emphasized that this assumption is made to simplify the the theoretical modeling at this stage, because calculating the long-term motion parameters and the blending process to build the filtered version of the current image is a very sophistictated process. To design a more accurate model for that process is an issue for further work. However, it will be shown later that this assumption approximates the real behavior of the video codec using global temporal filtering very well.

Thus we consider two error components of our model for temporal noise reduction. The temporally overlapped quantization error represented by its variance $\sigma^2_{e_q}$ and the prediction error variance due to the motion estimation $\sigma^2_{e_m}$:

$$\begin{aligned}\sigma^2_{e_q} &= 2^{-2R}\frac{\sigma_x^2}{N}\\ \sigma^2_{e_m} &= N \cdot 2\sigma^2_\Delta\sigma^2_x(2-\alpha_1-\alpha_2)\end{aligned} \tag{8.24}$$

We assume that the final error variance is built by the sum of the two components shown above. Thus the D-R-function of our model for the temporal noise reduction with Equ. 8.24 is:

$$\underline{\sigma^2_{e_{tf}} = 2^{-2R}\frac{\sigma_x^2}{N} + N \cdot 2\sigma^2_\Delta\sigma^2_x(2-\alpha_1-\alpha_2).} \tag{8.25}$$

Now it is of interest how possible bit rate savings are carried out from this theoretical D-R-function. For that, the distortion values of Equ.'s 8.23 and 8.25 are set equal. The bit rate of the general quantization error shall be R_1 and the bit rate using temporal noise reduction shall be R_2. An equation of the bit rate R_2 can now be derived as:

$$\begin{aligned}\sigma^2_{e_{xq}} &\overset{!}{=} \sigma^2_{e_{tf}}\\ 2^{-2R_1}\sigma_x^2 &= 2^{-2R_2}\frac{\sigma_x^2}{N} + N \cdot 2\sigma^2_\Delta\sigma^2_x(2-\alpha_1-\alpha_2)\\ 2^{-2R_2}\frac{1}{N} &= 2^{-2R_1} - N\cdot 2\sigma^2_\Delta(2-\alpha_1-\alpha_2)\\ -2R_2 - ld(N) &= ld\{2^{-2R_1} - N2\sigma^2_\Delta(2-\alpha_1-\alpha_2)\}\\ -2R_2 &= ld\{2^{-2R_1} - N2\sigma^2_\Delta(2-\alpha_1-\alpha_2)\} + ld(N)\\ R_2 &= -\frac{1}{2}\Big\{ld\{2^{-2R_1} - N2\sigma^2_\Delta(2-\alpha_1-\alpha_2)\} + ld(N)\Big\}\end{aligned} \tag{8.26}$$

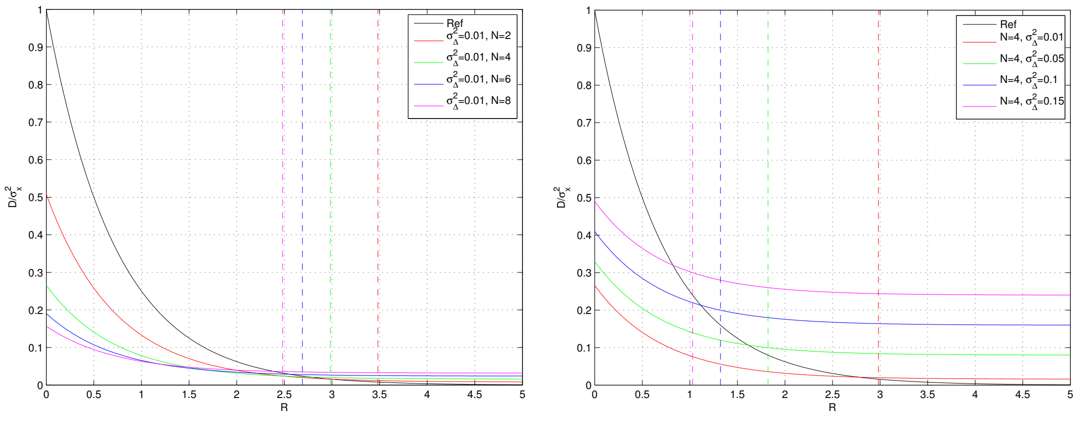
For a meaningful interpretation of the equation derived in Equ. 8.26 for the bit rate of the temporal filtered video signal, limits are calculated where Equ. 8.26 is valid considering the real coding and filtering algorithm. First, a lower limit for R_1 is derived. Equ. 8.26 makes only sense if the term $2^{-2R_1} - N2\sigma^2_\Delta(2-\alpha_1-\alpha_2)$ is greater than zero. This leads to:

$$\begin{aligned}
0 &< 2^{-2R_1} - N2\sigma^2_\Delta(2-\alpha_1-\alpha_2) \\
2^{-2R_1} &> N2\sigma^2_\Delta(2-\alpha_1-\alpha_2) \\
-2R_1 &> ld(N2\sigma^2_\Delta(2-\alpha_1-\alpha_2)) \\
R_1 &> -\frac{1}{2}ld(N2\sigma^2_\Delta(2-\alpha_1-\alpha_2)) \qquad (8.27)
\end{aligned}$$

The next consideration is that the number of frames for temporal filtering N is greater or equal to 1. This means that $ld(N) \geqslant 0$. The higher limit of N can be derived from Equ. 8.26. The higher limit for N and the error variance of the pixel difference due to motion estimation σ^2_Δ, respectively:

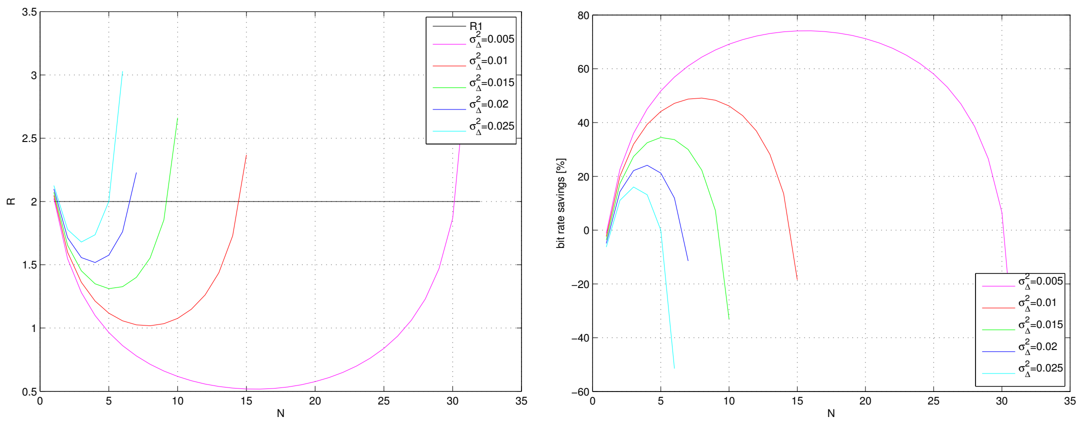
$$\begin{aligned}
0 &\leqslant 2^{-2R_1} - N2\sigma^2_\Delta(2-\alpha_1-\alpha_2) \\
2^{-2R_1} &\leqslant N2\sigma^2_\Delta(2-\alpha_1-\alpha_2) \\
N &\geqslant \frac{2^{-2R_1}}{2\sigma^2_\Delta(2-\alpha_1-\alpha_2)} \qquad (8.28) \\
\sigma^2_\Delta &\geqslant \frac{2^{-2R_1}}{2N(2-\alpha_1-\alpha_2)} \qquad (8.29)
\end{aligned}$$

Defining equations for ranges of N and σ^2_Δ is important for drawing rate-distortion curves of the theoretical R-D-functions. For that, reasonable values have to be set to show curves which can be interpreted. Figure 8.7 shows curves for the theoretical rate-distortion function developed above. The behavior with a variable frame number N to be used for filtering and a fix motion estimation error variance can be seen in Fig. 8.7(a). It can be seen that with increasing the number of frames to be used for filtering, the distortion decreases at the lower bit rate ranges. However, the motion estimation error has a higher impact at higher bit rate ranges, which causes in an increase of the distortion in that range. This is illustrated by the dashed vertical lines, where the model of our temporal filtering method crosses the reference model without filtering. This matches exactly with the real PSNR curves shown above for GMTF post-processing. Figure 8.7(b) shows theoretical curves with variable motion estimation error variance and a fix number of frames N. This case illustrates that with an increase of motion estimation errors, the gain of the filtering approach decreases rapidly. Another very interesting issue is shown in Fig. 8.8. Here, rate-"number of frames"-curves are shown to examine the behavior of the possible bit rate savings. In both cases, i.e. the rate (of the filtered version) vs. number of frames and bit rate savings (difference of $R_{filtered}$ and $R_{reference}$), it can be seen that there is one optimal number of frames to be filtered, which brings the best improvement. This outcoming aspect brings us to a design of a visual quality assessed video codec, where an optimal number of frames can be found at the encoder to get the best visual quality at the decoder after GMTF post-processing. Finally, we can say that our first theoretical model approximates the real behaviour of the GMFT approach very well. However, further investigations can be done in this area, especially for modeling the motion estimation error, which is approached very simply in our first approach.



(a) Variable frame number N (b) Variable motion estimation error variance

Figure 8.7: Curves of theoretical rate-distortion function Equ. 8.25



(a) R-N-curve, variable motion estimation error variance (b) Bit rate savings ΔR-N-curve, variable motion estimation error variance

Figure 8.8: Theoretical rate(R)-number of frames(N) function Equ. 8.26

8.3.3 Visual quality assessed Video Coding using Global Motion Temporal Filtering (VQVC)

Figure 8.9 shows a block diagram of the temporal deblocking approach included in an H.264/AVC coding environment. Herein, frames of a given video sequence are normally coded as one would do using H.264/AVC without using its standardized deblocking filter [45].

At the encoder side, frames are decoded and stored in a buffer that is used for the temporal deblocking filter. As described above, for a given reference frame I_{ref} the temporal

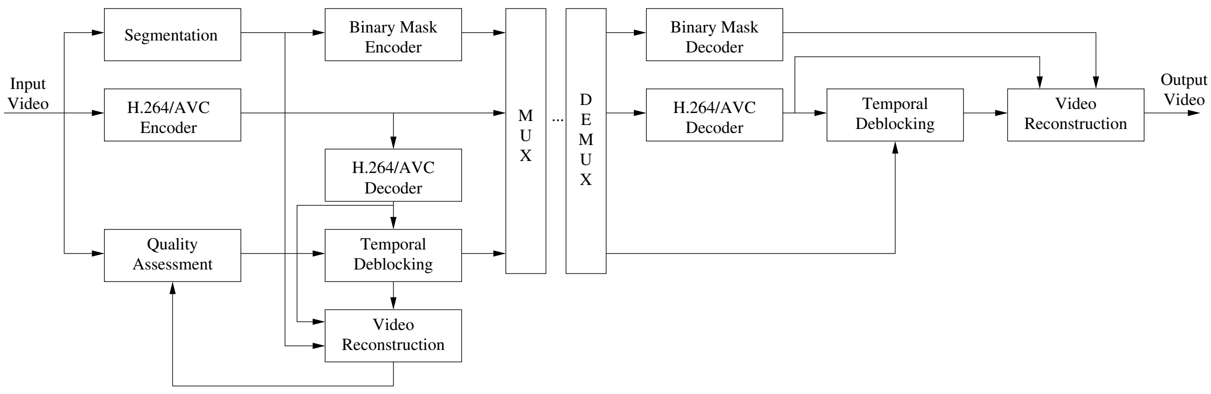


Figure 8.9: Global motion temporal filtering within a video coding environment

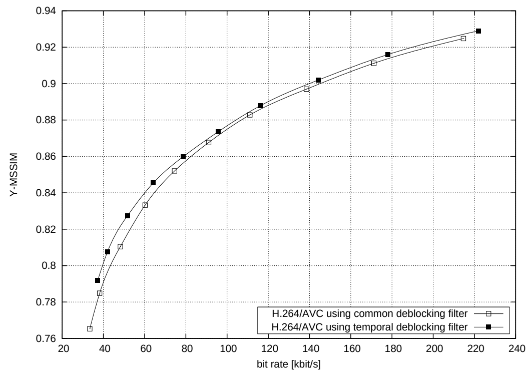
neighborhood, consisting of a set of distorted frames, is taken into account for deblocking. The algorithm successively warps temporally neighboring frames into the coordinate system of the reference frame. To this end, the global motion – e.g. an 8-parameter perspective motion model – has to be estimated, which is done using a hierarchical gradient descent approach based on the Gauss-Newton method. Thus, a growing image stack of spatially aligned frames is generated, which is blended together to build a single denoised representation I'_{ref} of the reference frame. In every blending step the quality of the current representation I'_{ref} is compared to the original frame $I_{orig,ref}$ using SSIM. The representation with the highest SSIM value is taken as the final deblocked frame. We have chosen the SSIM metric because, as already mentioned, it has been shown recently that the SSIM metric fits best to the human perception compared to other objective metrics for images. Therefore, we have presented a coding approach which is optimized regarding the human perception. Of course, it is possible to use any other metric in the "quality assessment" step, like PSNR. However, it is then optimized regarding the respective behavior of the metric used. For example, if the PSNR is used, the quality of the frames is not really optimized regarding the human perception which is well-known.

The number of frames N used for deblocking is differentially coded using signed mapping Exp-Golomb codes and sent as side information to the receiver. Temporal deblocking at the encoder only takes place to assess best quality at the receiver and therefore to measure the ideal amount of frames to be used.

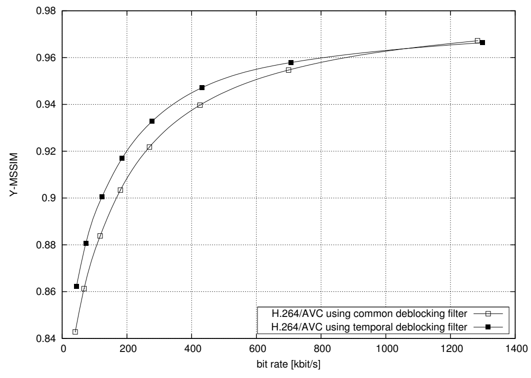
Foreground segments are defined as segments in the scene that move other than the global motion. These segments vanish successively from the deblocked frames the more temporal neighbors are used for generating them. The segments are coded using H.264/AVC with standardized deblocking filter. The segment mask has to transmitted as side information to the receiver. Automatic segmentation takes place in a preprocessing step. Since it is only necessary to ensure a correct binary mask, we will not further define the way it is generated. In our work we used the algorithm developed in Chapter 3 which is an anisotropic diffusion-based background subtraction technique using Local Background Sprites.

At the receiver the common H.264/AVC bit stream, the binary object mask, and the number of frames used in the temporal deblocking filter are decoded. The deblocking filter computes the denoised frames as defined above. Finally, the frames are reconstructed using the binary foreground object mask and presented to the viewer. In our approach, global motion parameters are not transmitted. The decoder estimates the parameters based on the noisy decoded images.

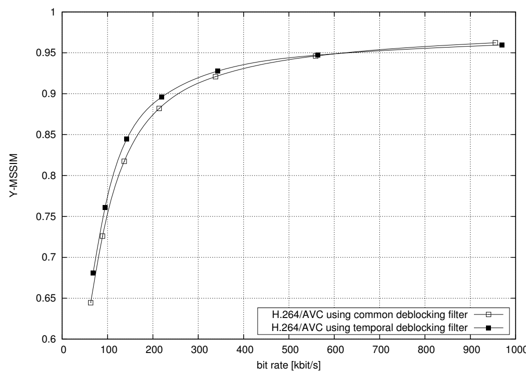
8.3.4 Experimental Evaluation



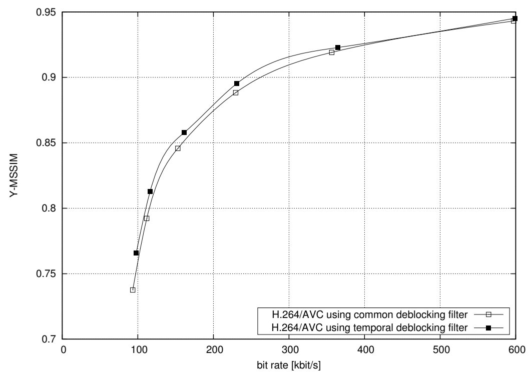
(a) Sequence "Biathlon"



(b) Sequence "Birds"



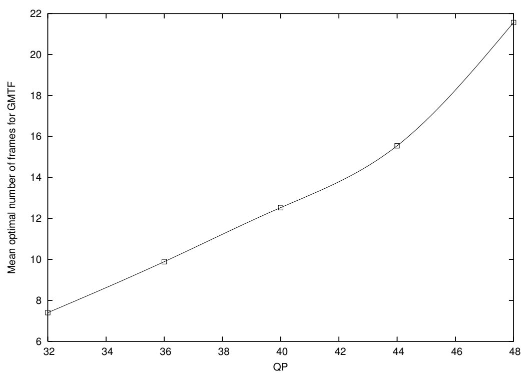
(c) Sequence "Desert"



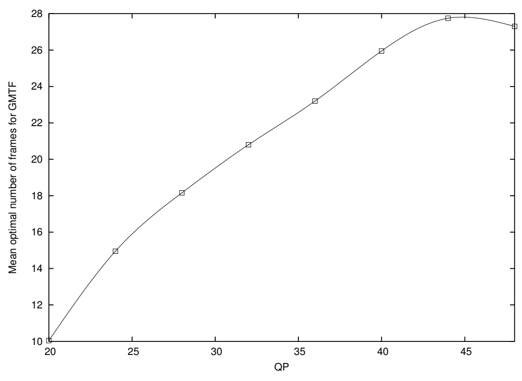
(d) Sequence "Race1"

Figure 8.10: Comparison of H.264/AVC deblocking with VQVC in terms of rate-distortion performance (SSIM)

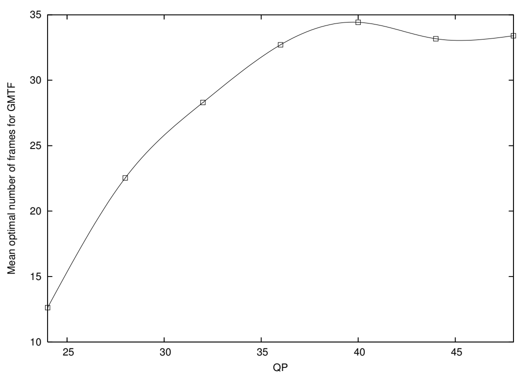
For experimental evaluation we compared the common H.264/AVC deblocking filter to the new temporal deblocking approach. For the H.264/AVC encoder we used hierarchical B-frames and CABAC entropy coding. The in-loop deblocking filter is turned off for our new approach. We used four test sequences: "Biathlon" (352×288, 200 frames) taken from a German televison broadcast, "Birds" (720×576, 110 frames) from the BBC documentary "Planet Earth", "Desert" (720×400, 240 frames) from "Planet Earth" as well, and "Race1" (554×336, 100 frames) from an MPEG multi-view test sequence. Figure 8.10 shows compression efficiency in terms of rate-distortion performance. For distortion measurement we



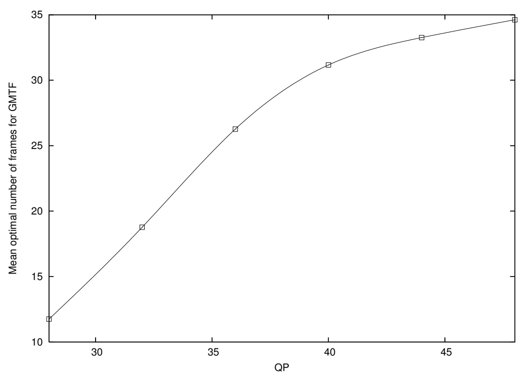
(a) Sequence “Biathlon”



(b) Sequence “Birds”



(c) Sequence “Desert”

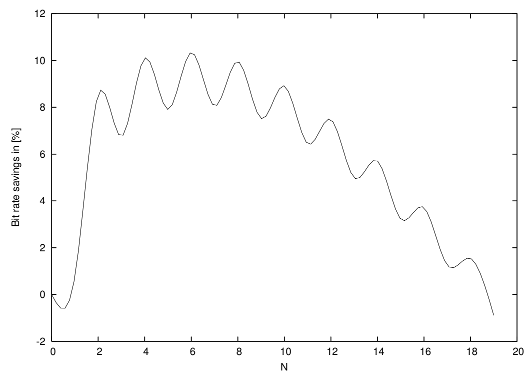


(d) Sequence “Race1”

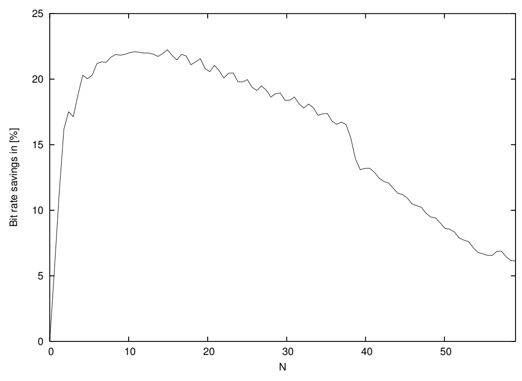
Figure 8.11: Mean optimal frame number for GMTF vs. quantization parameter QP

used the mean SSIM on the luminance channel (Y-MSSIM) instead of PSNR because the encoding scheme is optimized on that. We reach bit rate savings of up to 18% (“Birds” at 0.93 Y-MSSIM) mostly in lower bit rates. This is understandable because with higher bit rate the amount of blocking artifacts drops rapidly meaning there should be no difference between the various deblocking approaches. Figures 8.13 - 8.16 show a subjective comparison of a representative frame from the four test sequences considered (when using Adobe Reader, the complete videos can be watched and the reader can decide which one is subjectively better by considering the bit rate). Here, one can clearly see the significant deblocking capabilities of the approach presented. The proposed approach performs excellent deblocking while preserving edges in the images efficiently. Furthermore, Fig. 8.11 shows the mean optimal frame number for GMTF assigned in the encoder loop of the VQVC. It can be seen that the number of frames to be used for GMTF increases with an increase of the quantization parameter QP. This effect also complies with the theoretical consideration. A higher QP results in a higher quantization noise, i.e. block noise, and the distortion increases. For that more frames are needed to perform GMTF for removing the blocking artifacts. With lower QP’s, the bit rate increases and the blocking artifacts decrease. Therefore, less frames are needed for GMTF.

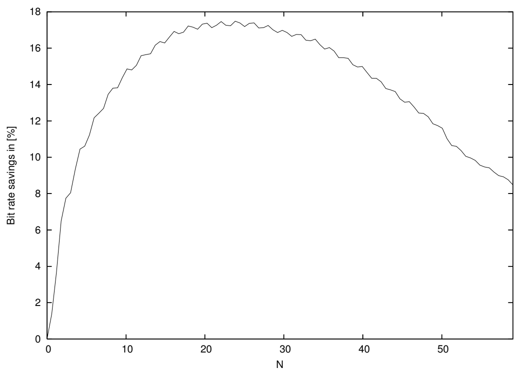
Finally, we would like to evaluate the theoretical model found in Equ.’s 8.25 and 8.26.



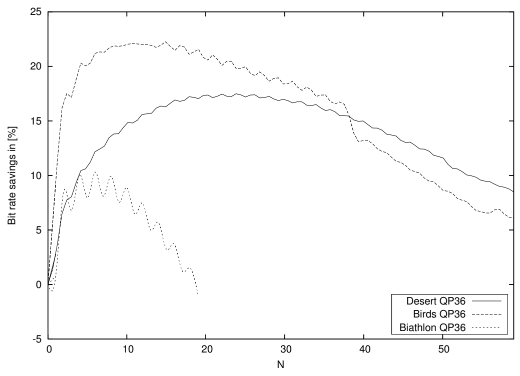
(a) "Biathlon", QP36



(b) "Birds", QP36



(c) "Desert", QP36



(d) All sequences

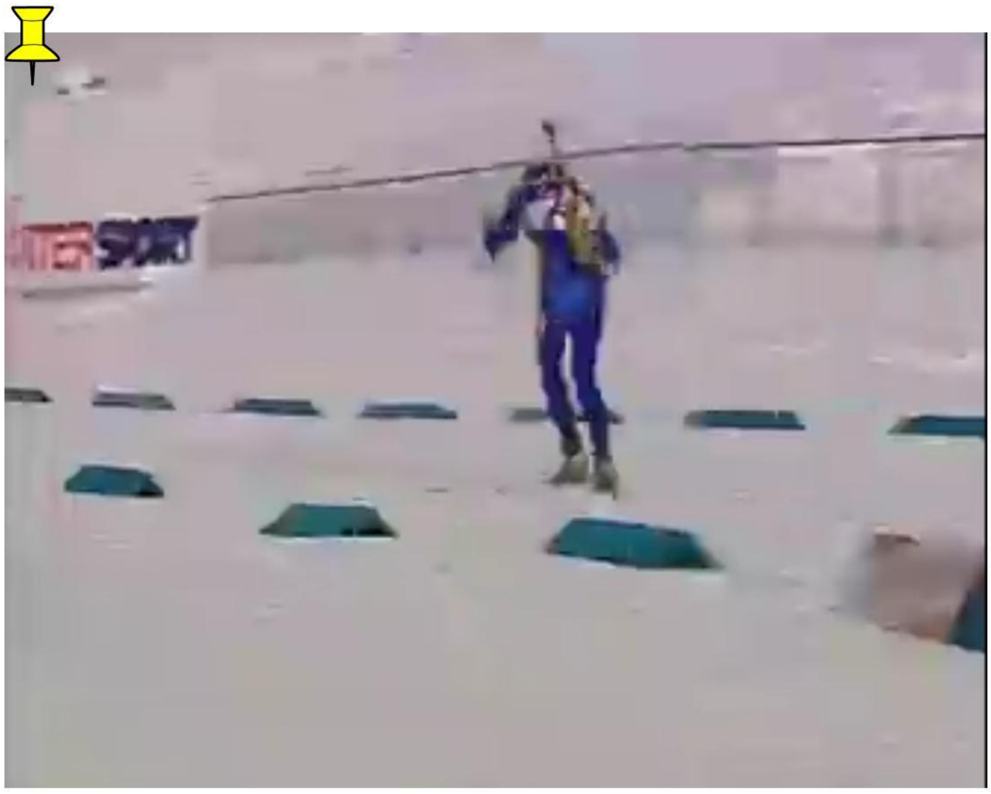
Figure 8.12: Bit rate savings vs. number frames for filtering

We also can do this because our theoretical model also applies for the SSIM (see Appendix). Therefore, we consider theoretical rate(R)-number of frames for filtering (N) function Equ. 8.26 shown in Fig. 8.8. We would like to show how good our theoretical function models the behavior of the VQVC. For that, we conducted experiments to draw R-N-curves with the experimental data. Figure 8.12 shows the results for three test sequences ("Biathlon", "Birds", "Desert") at one QP. It can be seen that the experimental curves approximately follow the theoretical curves. The curves have different shapes depending on the possible bit rate saving that can be achieved. As the theoretical function demonstrate, the bit rate savings increase with a decreasing error variance, motion esitmation error. The motion estimation performs very good with the test sequences "Birds" and "Desert", but is more difficult with "Biathlon". Therefore, less bit rate savings are possible with "Biathlon" than with "Birds" and "Desert". However, it can be observed that the gain of the temporal filter and the VQVC increases with an increasing resolution of the test video. This is also a reason why the amount of bit rate savings is much higher with the test sequences "Birds" and "Desert". Besides, the "zig-zag" inside the R-N-curve comes from the different performance of forward and backward motion estimation. Forward motion estimation is performed first and for backward motion estimation, the inverse motion parameters are used, which have a slightly lower quality that the direct estimated foreward motion paramters. This results in such a “zig-zag” form of the curve. Thus, we have proven with our theoretical model that the global motion temporal filter inside a video codec brings a very good performance and is able to outperform the common state-of-the-art techniques.

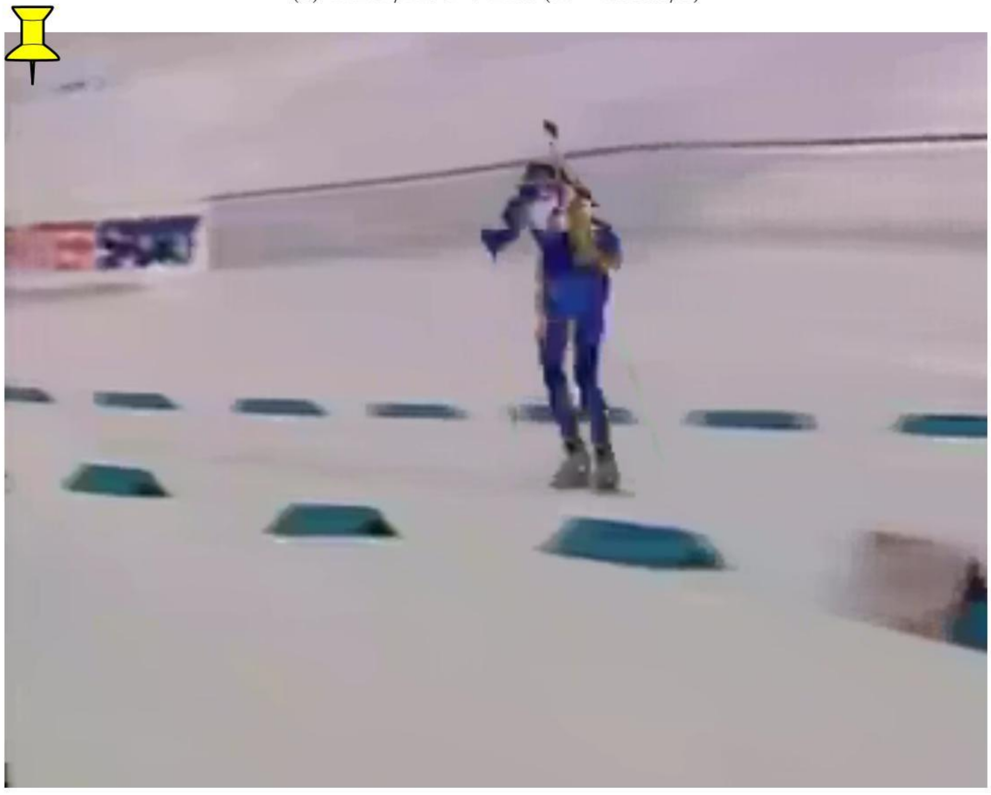
8.4 Summary and outlook

In this chapter, a new filtering approach was presented to tackle the blocking artifact problem of highly compressed video. We have shown in theory and practice that our global motion temporal filter produces very good results even with highly distorted material at very low bit rate ranges. With the theoretical consideration, we have proven the concept of our approach. In practice, motion estimation is needed to provide the aligned versions of the frame to be filtered. Therefore, we developed a theoretical model including the motion estimation step to show the behavior of our approach in a real coding environment. The theoretical rate-distortion function achieved approximates real RD-curves very good. The more frames N are used for filtering the better is the performance in the lower bit rate ranges, but the lower is the performance in higher bit rate ranges due to the motion estimation errors. This means that we have proven our method theoretically and practically. Based on that, a visual quality assessed video codec has been designed which uses the GMTF technique. Furthermore, the coding design includes an optimization step, which is adapted to the human perception.

Having this technique available, further steps can be examined in both directions. First, the model-based approach can be pursued, where e.g. further optimization can be done regarding region-of-interests etc. Second, we have also shown that this GMT filter can produce very good results in a post-processing step when a common hybrid video coder is used. It would be very interesting to see if the GMTF brings also improvements when it is used for in-loop filtering.

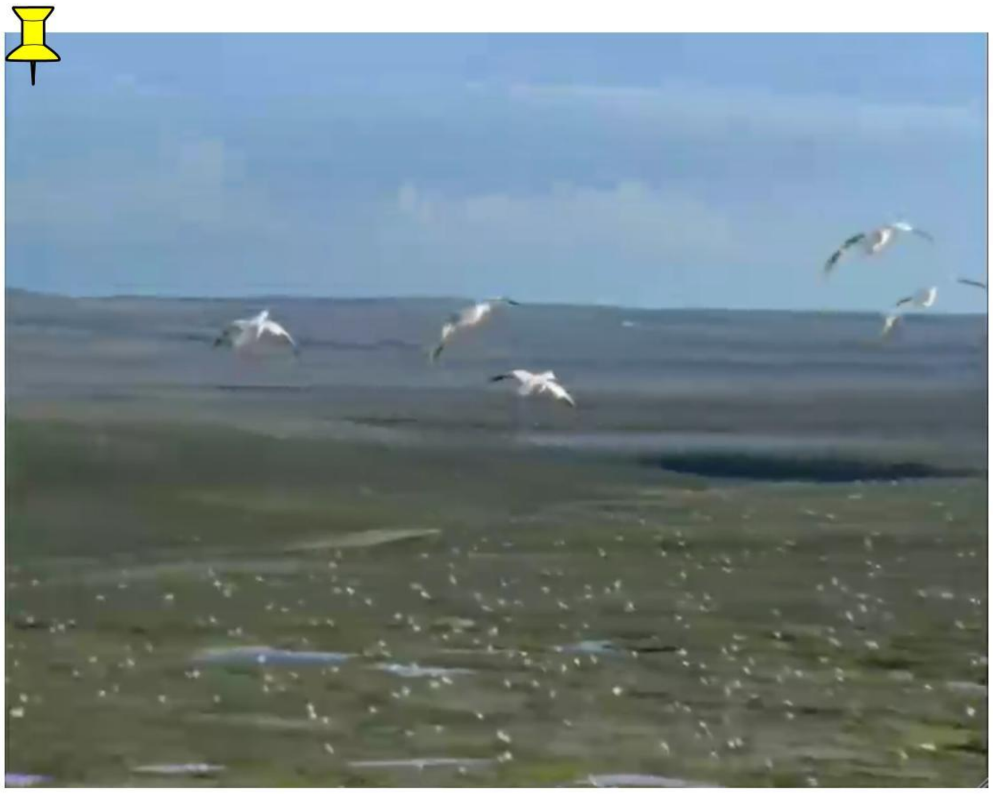


(a) H.264/AVC + List ($R = 60kbit/s$)

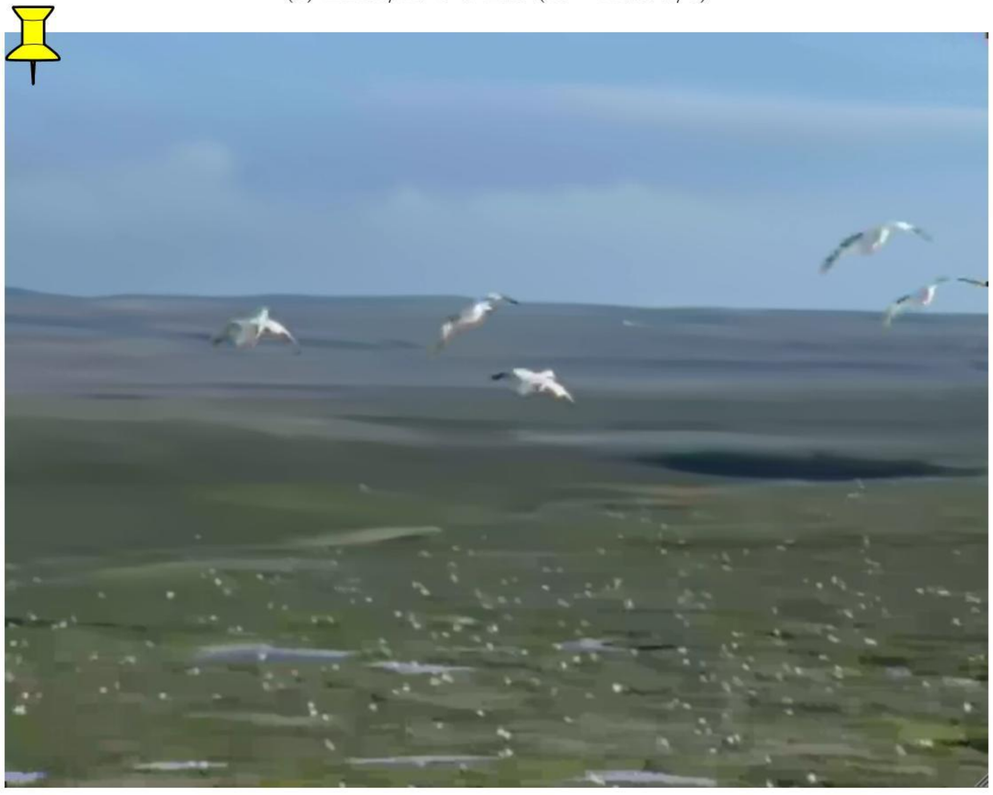


(b) H.264/AVC + GMTF ($R = 52kbit/s$)

Figure 8.13: Subjective comparison of H.264/AVC with in-loop deblocking and GMTF for post-processing "Biathlon" (Click on the images while using *Adobe Reader*)

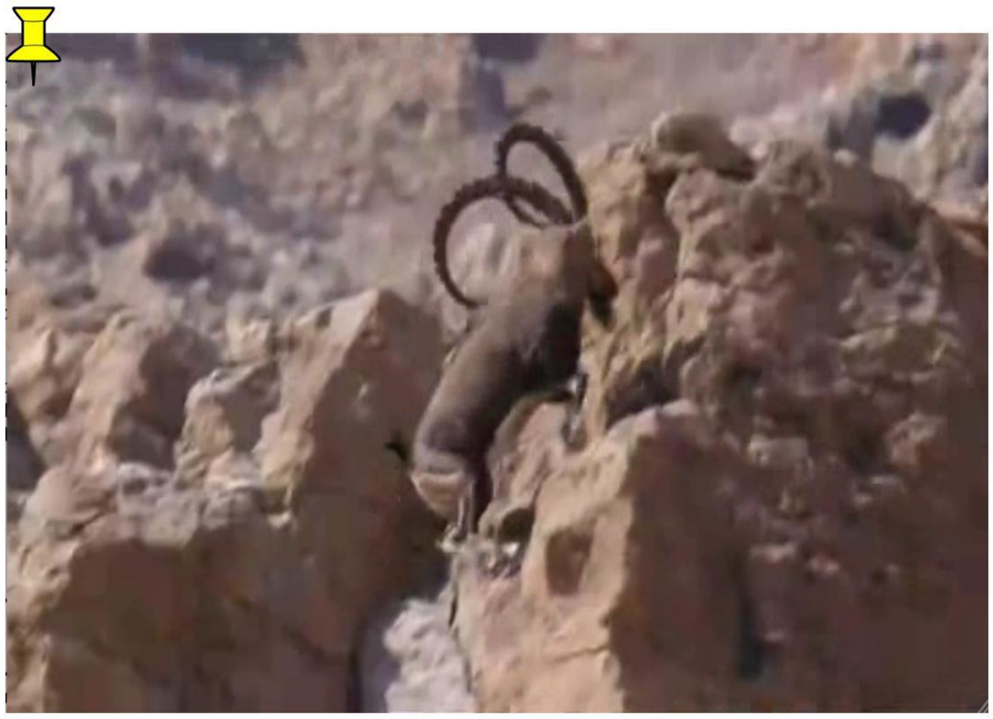


(a) H.264/AVC + List ($R = 117 kbit/s$)

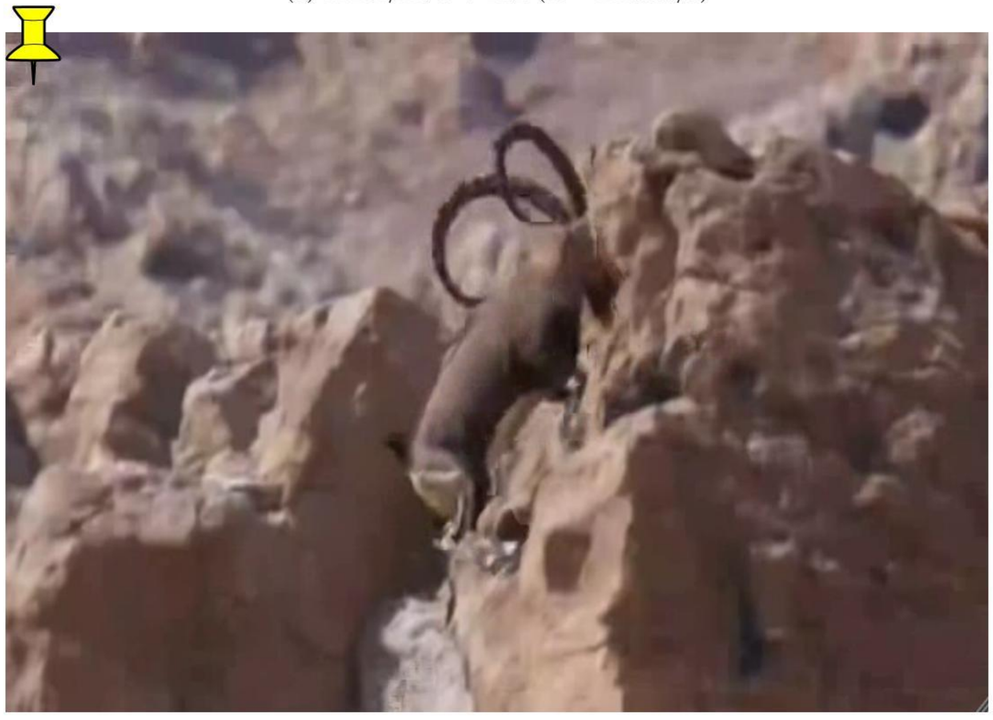


(b) H.264/AVC + GMTF ($R = 73 kbit/s$)

Figure 8.14: Subjective comparison of H.264/AVC with in-loop deblocking and GMTF for post-processing "Birds" (Click on the images while using *Adobe Reader*)

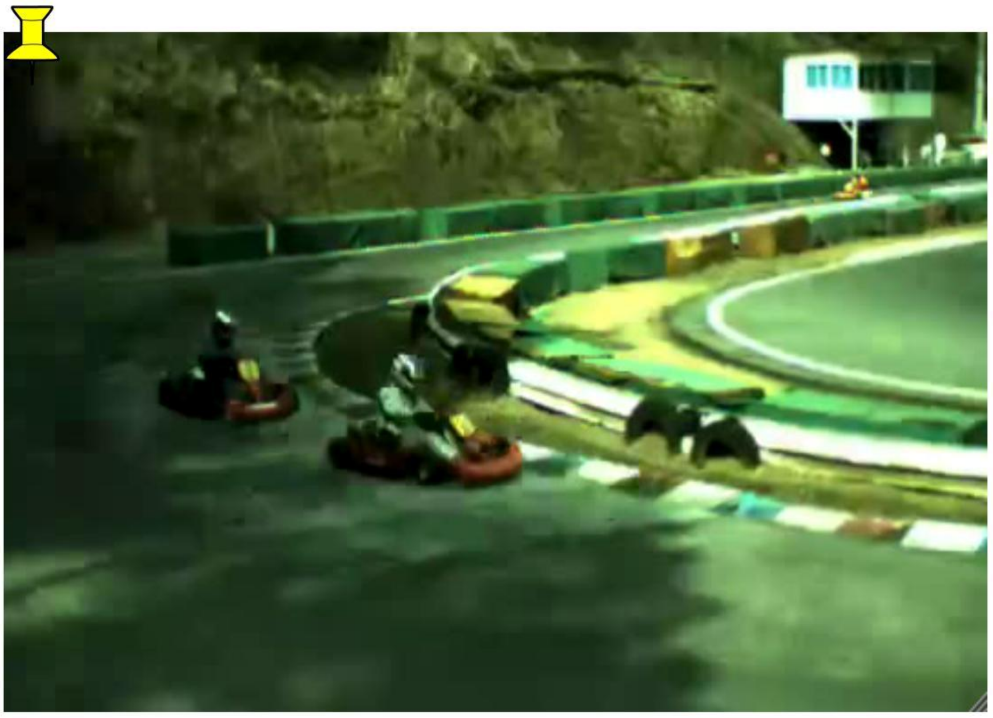


(a) H.264/AVC + List ($R = 214kbit/s$)



(b) H.264/AVC + GMTF ($R = 142kbit/s$)

Figure 8.15: Subjective comparison of H.264/AVC with in-loop deblocking and GMTF for post-processing "Desert" (Click on the images while using *Adobe Reader*)


(a) H.264/AVC + List ($R = 229kbit/s$)

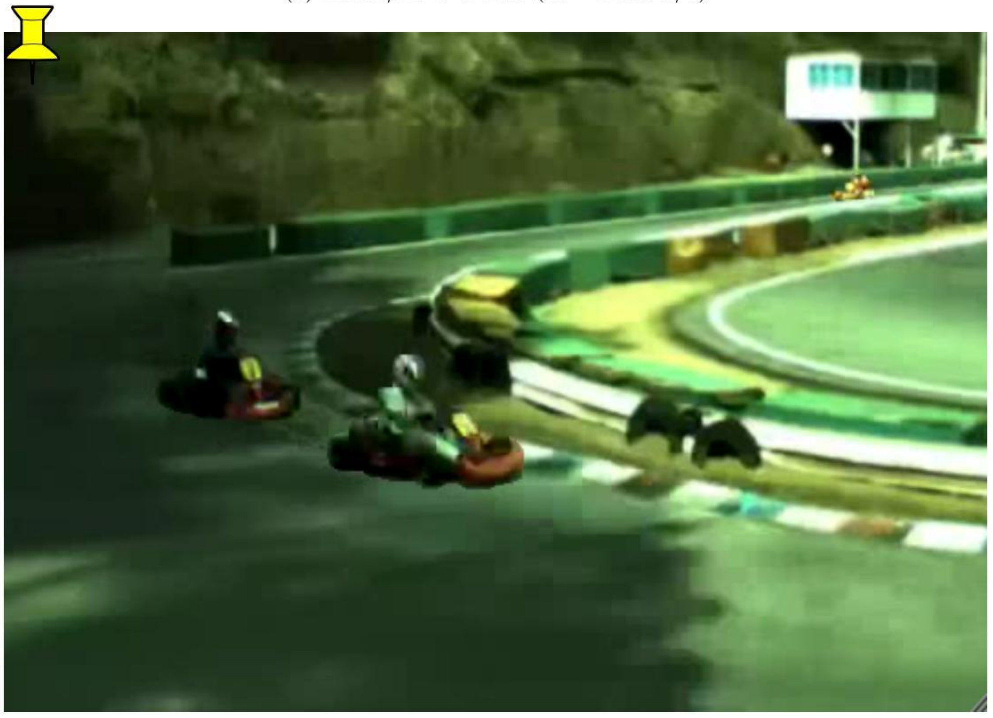

(b) H.264/AVC + GMTF ($R = 161kbit/s$)

Figure 8.16: Subjective comparison of H.264/AVC with in-loop deblocking and GMTF for post-processing "Desert" (Click on the images while using *Adobe Reader*)

Chapter 9

Summary and Conclusion

In this thesis, we have determined aspects related to advanced video coding techniques which could be integrated into common video transmission systems. We have considered the Sprite coding approach developed almost 15 years ago. Because of its complexity and alleged problem in terms of its applicability, this technique has not come into the market until now. However, in a large number of studies the high potential of this technique at all as well as partial algorithms has been shown. So the main goal of this thesis was first to pursue the traditional Sprite coding technique, improve its processing chain, further developing a Sprite coding scheme with an in-built object segmentation including the design of a rate-distortion algorithm. Additionally, methods have been developed to make the Sprite coding approach more applicable. Since this approach outperforms the common hybrid video codecs for certain kinds of input video sequences, a pre-processing analysis step is necessary to define whether an input sequence can be coded with the Sprite codec or not.

The second goal was to find out where else the partial algorithms within the processing chain of the automatic Sprite codec developed can be used. A temporal deblocking filter has been developed by use of the new Sprite generation technique developed in this thesis, that is called local background Sprites. It has been found that using this algorithm it is possible to remove blocking artifacts appearing in common hybrid video codecs significantly. Different scenarios for such a deblocking method have been evaluated including the use of the deblocking approach at the decoder. A complete video codec has been developed where temporal deblocking is performed in an optimal way.

Addtional to these practical developments and different designs of video codecs, a theoretical model has been developed. A theoretical rate-distortion function has been found, which shows that the codec developed is able to outperform the hybrid approach. Theoretial bit rate vs. number of frames for filtering curves underline the expected performance in practice.

In both practical matters, i.e. pursuing the traditional Sprite coding approach and developing new video codecs with use of partial algorithms related to the Sprite coding, very good results have been achieved in comparison to earlier work and especially to hybrid video coding methods. It can be stated that the techniques developed and evaluated in this thesis have come much closer to be useful for future video codecs. However, there is still a lot of space for future work both areas. Five possible further developments are presented in the following.

- Regarding the approach to bring the traditional Sprite coding towards general usability, the following encoding scheme could be developed. A two stage pre-analysis component selects the input video into several parts and defines which coding method is most appropriate for each video segment depending on their feature. A third coding issue can be added where the traditional Sprites are used as a prediction signal. A rate-distortion algorithm optimizes the scheme regarding all coding modes and all related features. For that, a rate-distortion optimization algorithm has to be developed dealing with the automatic Sprite coding approach and the common hybrid encoder in combination. The rate-distortion optimization approach developed in this thesis can be further developed for that purpose. The method proposed in this work decides only between two codecs. However, theoretically it is possible to have one codec for each different kind of input video classified in genres and/or features. For example, it has been shown that sports and documentary are appropriate for global motion-based prediction codecs like conventional Sprites. On the other side, there are codec-designs for news and video conferencing-like content, which can also be added in such a system. The main goal here would be to find a low-complex content analysis method that is used as a pre-processing step.
- The visual quality assessed video coding (VQVC) approach opens a wide new range of research. It is actually well-known that existing video coding approaches are not sufficiently designed towards the human perception, especially in the rate-distortion sense. The rate-distortion optimization algorithm developed in this thesis and the VQVC lead to the idea to develop a visual quality assessed rate-distortion optimized video coding system. Because there are different demands on a rate-distortion optimizing algorithm, e.g. frames have to be decoded completely when the quality is analyzed by such an algorithm, the RD-optimization approach developed in this thesis can also be applied. The theoretical consideration derived for this new coding approach can also be further developed, including refining the existing theoretical rate-distortion function to model the behavior of this VQVC-coding system.
- We have shown the performance of temporal filtering for deblocking as a post-processing step. The results were very promising with the considered test sequences. Now, the next step for future work is to implement this filter inside the encoding loop of a hybrid video encoder. For that, two ways are possible. We have shown that temporal filtering results in very good deblocking. Therefore, temporal filtering can be used

as an in-loop deblocking filter. If the global motion temporal filter is used, a combination with a spatial deblocking filter is appropriate. The spatial deblocking filter is necessary to perform deblocking of objects that do not follow the global motion. Having this design, further developments in finding an optimal combination of the spatial and temporal filter or enhance the temporal filter can be determined. One enhancement of the temporal filter could be that the filter does not only work for one motion but also in different motion directions regarding several moving objects appearing in the video. The main improvement using the temporal filter for deblocking will be the enhancement of the motion estimation due to its improved deblocking performance. The second way to use the temporal filter developed is to apply it as an additional prediction mode. Here, higher-order motion estimation can be used for generating the prediction signal. It is possible to integrate the method for P-, B-, and hierarchical B-frames. The impact of the temporal filter as in-loop or prediction filter can be determined separately. Then, it will be very interesting to find out which combination between the in-loop temporal filter, spatial deblocking filter, other in-loop filters (Wiener) and the several prediction modes is the best. Considering the prediction modes, prediction signals generated by common block matching methods and higher-order motion estimation used for the temporal filter can be evaluated comprehensively. Since a higher-order motion model is used for global motion temporal filtering, work is also necessary using the motion vectors already calculated to obtain higher-order motion parameters for temporal filtering that are as accurate as possible. If it is possible to come very close to the pixel-based methods the applicability would be very high. On one side the computational cost would decrease and on the other side, no additional motion parameters have to be transmitted.

- Every aspect described in the hybrid video coding application for single-view videos can also be transfered into multi-view video coding. In such an environment, a temporal filter can also take into account not only consecutive frames of one view but also frames from the spatial views. Using the spatial views, a higher-order disparity estimation between the views is necessary. More precisely, long-term higher-order disparity estimation has to be performed to achieve a good filtering. For example, assuming a multi-view video with eight views and fourth view to be filtered, long-term disparity estimation has to be applied in both directions, from fourth to one and from fourth to eight view. It has to be determined whether the temporal and spatial view filtering together improve the deblocking and/or prediction results of a multi-view video codec. This depends on the long-term higher-order disparity estimation, which could be the main focus of further work in this matter.

- The possibility of the temporal filter and its performance for video quality enhancement of compressed video has been proved in a first theoretical consideration. Here, the quantizer noise was assumed to be white. The motion estimation error was also taken into account. A model has been developed that matches very well with the practical behavior of e.g. the VQVC. However, further work will be done in this field to enhance the theoretical model, to prove that the error coming from blocking artefacts can be treated as white noise, and to design a model for the temporal filter inside a prediction-based hybrid video encoder. Considering the post-processing idea again brings us to additional applications. The concept of temporally aligning images can also be used in superresolution techniques. This means that we are able to improve the video quality of highly compressed and distorted video and even enhance the resolution by combining our approach with a superresolution technique. The increasing amount of video data especially in very popular social networks is immense. A lot of such video material has a good quality. However, there is a huge amount of video that contains compression artifacts, noise and has low resolution. Therefore, if it would be possible to use our proposed technique to build a video enhancement method as post-processing we expect that the quality and even the resolution can be improved. This would be very attractive for social networks, e.g. YouTube, Myvideo, Sevenload etc. Additionally, there is a lot of research activity in analyzing the video content of these video databases. It is clear that the performance of analysis algorithms gets worse with a decrease of the video quality caused by e.g. compression artifacts. If we are able to enhance noisy highly compressed video data, the analysis algorithms will perform better as well. That means that further research could be to design a video analysis system for video databases of e.g. social networks with a video enhancement step using the developed tools as a pre-processing step. A starting point here is the development of an algorithm that can obtain highly accurate higher-order motion parameters from available motion vectors (as already mentioned above). The performance of the entire quality enhancement algorithm is dependent on this basic work. Further steps are the selection of a appropriate superresolution algorithm or a design of a new one based on the temporal filter proposed in this work.

Finally, we can say that standing on the shoulders of the previous works in this field we have come much closer to carry these powerful techniques towards a wide applicability. We have shown a wide range of applications that can be further developed, not only in the video coding area. The implementation of the temporal filter inside an encoding loop make it very attractive for future hybrid video coding research and development. However, the potential of pursuing alternative video coding methods as well as all related applications is also very promising. To close the circle, the huge and permanently increasing amount of video data available at todays Web 2.0 social networks as well as "classical" video transmission systems like TV broadcast and DVD/Bluray storages needs enhanced video processing and coding algorithms more than ever.

Appendix A

SSIM and the Approach for Theoretical Modeling

We would like to show that the approach of the theoretical modeling fits also to the GMTF when the SSIM is used for objective image quality measurement. In Chapter 8, the VQVC was designed to optimize the visual quality during the encoding process. For that, the SSIM was applied to measure the visual quality according to the HVS.

We assume that a quantized signal y is the sum of an original signal x and quantization noise e:

$$y = x + e. \tag{A.1}$$

Under the assumption that the quantization error is small (which assumes a sufficiently high number of bits per sample), it is

$$E[x \cdot e] = 0. \tag{A.2}$$

Thus, x and e are uncorrelated. If x is Gaussian distributed, this quantization noise error variance is generated by coding at rate R:

$$E[e^2] = \sigma_e^2 = \sigma_x^2 \cdot 2^{-2R}. \tag{A.3}$$

Assume that it is possible to observe N representations of y, i.e. corresponding quantized pixels from N frames of a video sequence. Averaging these quantized pixels y in temporal direction reduces the error variance by

$$E[e_f^2] = \frac{E[e^2]}{N}, \tag{A.4}$$

with N being the number of noisy versions x. This assumes that the versions of e in temporal direction are also not correlated. Our goal is to see whether it is possible to code all versions of x along the temporal direction with reduced bits/sample when afterwards temporal filtering is applied. In other words: Assuming same quality after reconstruction of x (measured using SSIM), does temporal filtering result in reduced bits/sample? For this aim we request that $\text{SSIM}_{nf} = \text{SSIM}_f$, where nf is not filtered and f is the filtered quality. Without loss of generality we assume that x, y, and e are zero mean processes, thus

$$E[y^2] = \sigma_y^2 = \sigma_x^2 + \sigma_e^2, \tag{A.5}$$

$$\text{cov}(x, y) = E[xy] = \sigma_x^2. \tag{A.6}$$

In this case $\text{SSIM}(x, y)$ reduces to

$$\text{SSIM}_{nf}(x, y) = \frac{2\,\text{cov}(x, y) + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} = \frac{2\sigma_x^2 + C_2}{2\sigma_x^2 + \sigma_{e_{nf}}^2 + C_2}, \tag{A.7}$$

$$\text{SSIM}_f(x, y) = \frac{2\sigma_x^2 + C_2}{2\sigma_x^2 + \sigma_{e_f}^2 + C_2} = \frac{2\sigma_x^2 + C_2}{2\sigma_x^2 + \frac{\sigma_{e_{nf}}^2}{N} + C_2}, \tag{A.8}$$

where $\sigma_{e_{nf}}^2$ is the quantization error variance when coding before temporal filtering is applied. C_2 is a constant factor. It is apparent that the SSIM decreases if σ_e^2 increases. If the quantization noise e is correlated with x, the covariance term will influence SSIM more directly. For our purpose we set $\text{SSIM}_{nf}(x, y) = \text{SSIM}_f(x, y)$, and thus have

$$\sigma_{e_{nf}}^2 = \sigma_{e_f}^2 \tag{A.9}$$

The terms $\sigma_{e_{nf}}^2$ and $\sigma_{e_f}^2$ are generated by coding the signal x at different rates R_{nf} and R_f. According to Equation A.3 this accounts to

$$\begin{aligned} 2^{-2R_{nf}} &= \frac{2^{-2R_f}}{N} \\ R_f &= R_{nf} - \frac{1}{2}\log_2(N) \end{aligned} \tag{A.10}$$

This completes the proof. A bit rate reduction to the amount of $\frac{1}{2}\log_2(N)$ is achieved (assuming same SSIM). In other words: It is possible to encode N identical versions of x with $\frac{1}{2}\log_2(N)$ bits/samples less than without filtering and an identical SSIM is still achieved.

This is the same model achieved in Equ.8.8 not considering the motion estimation error. With Equ. A.9 we have shown that using the SSIM does not influence our theoretical considerations.

Bibliography

[1] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1090–1097, 2001.

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:497–501, February 2004.

[3] A. Cavallaro and T.Ebrahimi. Change detection based on color edges. *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, 2:141–144, May 2001.

[4] S.-Y. Chien, C.-Y. Chen, W.-M. Chao, C.-W. Hsu, Y.-W. Huang, and L.-G. Chen. A fast and high subjective quality sprite generation algorithm with frame skipping and multiple sprites techniques. In *Int. Conf. on Image Processing (ICIP'02)*, Rochester, New York, USA, September 2002.

[5] R. Crinon and I. Sezan. Sprite-based video coding using on-line segmentation. In *IEEE Proc. ICASSP'98*, pages 2981–2984, Seattle, WA, USA, May 1998.

[6] H. Schwarz D. Marpe and T. Wiegand. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, July 2003.

[7] F. Dufaux and F. Moschen. Background mosaicking for low bit rate video coding. In *IEEE Proc. ICIP'96*, pages 673–676, Lausanne, Switzerland, Sep 1996.

[8] Frederic Dufaux and Janusz Konrad. Efficient, robust, and fast global motion estimation for video coding. *IEEE Transactions on Image Processing*, 9:497–501, 2000.

[9] M.P. Eckert and A.P. Bradley. Perceptual quality metrics applied to still image compression. *Signal Processing*, 7:177–200, Nov. 1998.

[10] D. Farin. *Automatic Video Segmentation Employing Object/Camera Modeling Techniques.* PhD thesis, University of Eindhoven, December 2005.

[11] D. Farin and P. H. N. de With. Enabling arbitrary rotational camera motion using multisprites with minimum coding cost. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(4):492–506, April 2006.

[12] D. Farin, P. H. N. de With, and W. Effelsberg. Video object segmentation using multi-sprite background subtraction. In *Int. Conf. on Multimedia and Expo (ICME)*, Taipei, Taiwan, June 2004.

[13] H. Foroosh, J. Zerubia, and M. Berthod. Extension of phase correlation to subpixel registration. *IEEE Trans. Image Processing*, 11(3):188–200, 2002.

[14] F.Pereira and T.Ebrahimi. *The MPEG-4 Book*. Springer, 2002.

[15] B. Girod. What's wrong with mean-squared error. *Digital Images and Human Vision*, pages 207–220, A.B. Watson, Ed. Cambridge, MA: MIT Press,1993.

[16] A. Glantz. Objektsegmentierung in 2d-videosequenzen mit bewegter kamera. Diploma thesis, TU Berlin, Berlin, Germany, May 2008.

[17] M. Haller, A. Krutz, and T. Sikora. A generic approach for motion-based video parsing. In *Proc. EUSIPCO*, pages 713–717, 2007.

[18] C.-T. Hsu and Y.-C. Tsan. Mosaics of video sequences with moving objects. *Signal Processing: Image Communication*, 19:81–98, 2004.

[19] M. Irani and P. Anandan. Video indexing based on mosaic representations. *Proc. of the IEEE*, 86(5):905–921, May 1998.

[20] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and theirapplications. In *IEEE Proc. ICCV'95*, pages 605–611, Cambridge, MA, USA, Jun 1995.

[21] M. Irani, S. Hsu, and P. Anandan. Video compression using mosaic representations. *Signal Processing: Image Communication, special issue on Coding Techniques for Low Bitrate Video*, 7(4-6):529–552, November 1995.

[22] K. Jinzenji, H. Watanabe, S. Okada, and N. Kobayashi. MPEG-4 very low bit-rate video compression using sprite coding. In *IEEE International Conference on Multimedia and Expo, (ICME)*, August 2001.

[23] J. N. Kapur and P. K. Sahoo. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision Graphics Image Processing*, 29:273–285, 1985.

[24] Y. Keller and A. Averbuch. Fast gradient methods based on global motion estimation for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 13:300–309, April 2003.

[25] A. Krutz. Unsupervised global motion estimation with application to video mosaicing. Diploma thesis, TU Berlin, Berlin, Germany, January 2006.

[26] A. Krutz, M. Droese, M. Kunter, M. Mandal, M. Frater, and T. Sikora. Low bit-rate object-based multi-view video coding using mvc. In *First International 3DTV-Conference*, Kos, Greece, May 2007.

[27] A. Krutz, M. Frater, M. Kunter, and T. Sikora. Windowed image registration for robust mosaicing of scenes with large background occlusions. In *Int. Conf. on Image Processing (ICIP'06)*, Atlanta, USA, October 2006.

[28] A. Krutz, M. Frater, and T. Sikora. Improved image registration using the up-sampled domain. In *Int. Conf. on Multimedia Signal Processing (MMSP'06)*, Victoria, Canada, October 2006.

[29] A. Krutz, A. Glantz, T. Borgmann, M. Frater, and T. Sikora. Motion-based object segmentation using local background sprites. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, April 2009.

[30] A. Krutz, A. Glantz, T. Borgmann, M. Frater, and T.Sikora. Motion-based object segmentation using local background sprites. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009)*, Taipei, Taiwan, April 2009.

[31] A. Krutz, A. Glantz, M. Frater, and T. Sikora. Local background sprite generation. In *2008 International Workshop on Local and Non-Local Approximation in Image Processing, A satellite event of the 16th European Signal Processing Conference (EUSIPCO 2008)*, Lausanne, Switzerland, August 2008.

[32] A. Krutz, A. Glantz, M. Haller, M. Droese, and T.Sikora. Multiple background sprite generation using camera motion characterization for object-based video coding. In *3DTV Conference 2008, The True Vision Capture, Transmission and Display of 3D Video, May 2008, Istanbul, Turkey*, Istanbul, Turkey, May 2008.

[33] A. Krutz, A. Glantz, T. Sikora, P. Nunes, and F. Pereira. Automatic object segmentation algorithms for sprite coding using MPEG-4. In *50th International Symposium ELMAR-2008*, Zadar, Croatia, September 2008.

[34] A. Krutz, M. Kunter, M. Droese, M. Frater, and T. Sikora. Content-adaptive video coding combining object-based coding and H.264/AVC. In *26th Picture Coding Symposium (PCS)*, Lisbon, Portugal, November 2007.

[35] A. Krutz, M. Kunter, M. Droese, A. Glantz, M. Frater, and T. Sikora. Automatic sprite coding using H.264/AVC. In preparation.

[36] A. Krutz, M. Kunter, M. Mandal, M. Frater, and T. Sikora. Motion-based object segmentation using sprites and anisotropic diffusion. In *8th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, Santorini, Greece, June 2007.

[37] A. Krutz, M. Kunter, M. Mandal, M. Frater, and T. Sikora. Motion-based object segmentation using sprites and anisotropic diffusion. *8th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '07)*, 2007.

[38] C.D. Kuglin and D.C. Hines. The phase correlation image alignment method. In *Proc. IEEE 1975 Int. Conf. Cybernetics and Society*, pages 163–165, September 1975.

[39] M. Kunter. *Advances in Sprite Coding towards general usability.* PhD thesis, TU Berlin, January 2008.

[40] M. Kunter, J. Kim, and T. Sikora. Super-resolution mosaicing using embedded hybrid recursive flow-based segmentation. In *IEEE Int. Conf. on Information, Communication and Signal Processing (ICICS'05)*, Bangkok, Thailand, December 2005.

[41] M. Kunter, A. Krutz, M. Droese, M. Frater, and T. Sikora. Object-based multiple sprite coding of unsegmented videos using H.264/AVC. In *IEEE International Conference on Image Processing (ICIP2007)*, San Antonio, USA, September 2007.

[42] M. Kunter, A. Krutz, M. Mandal, and T. Sikora. Optimal multiple sprite generation based on physical camera parameter estimation. In *Visual Communications and Image Processing (VCIP'07)*, San Jose, USA, January 2007.

[43] M. Kunter and T. Sikora. Super-resolution mosaicing for object based video format conversion. In *7th Works. on Im. Analysis f. Multimedia Interac. Serv. (WIAMIS'06)*, Seoul, Korea, April 2006.

[44] R. Lienhart. Comparison of automatic shot boundary detection algorithms. *SPIE Storage and Retrieval for Still Image and Video Databases VII*, 3656:290–301, January 1999.

[45] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *IEEE Transactions on Circuits and Systems for Video Technology*, 13:614–619, July 2003.

[46] Y. Lu, W. Gao, and F. Wu. Fast and robust sprite generation for MPEG-4 video coding. In *IEEE Pacific Rim Conference on Multimedia (PCM'01)*, Bejing, China, October 2001.

[47] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proc. 7th Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[48] R. Mech and M. Wollborn. A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera. In *Proc. Workshop on Image Analysis for Multimedia Interactive Services WIAMIS97*, Louvain-la-Neuve, Belgium, June 1997.

[49] R. Mech and M. Wollborn. A noise robust method for segmentation of moving objects in video sequences. *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97, 1997 IEEE International Conference on*, 4:2657–2660 vol.4, Apr 1997.

[50] F. Moscheni, F. Dufaux, and M. Kunti. A new two-stage global/local motion estimation based on a background/foreground segmentation. In *IEEE Proc. ICASSP'95*, pages 2261–2264, Detroit, May 1995.

[51] P. Nunes and F. Pereira. Object-based rate control for the MPEG-4 Visual Simple Profile. In *Proc Workshop on Image Analysis for Multimedia Interactive Services WIAMIS*, Berlin, Germany, May 1999.

[52] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 338–343, San Juan, Puerto Rico, June 1997.

[53] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence archive*, 12(7):629–639, July 1990.

[54] M. H. Pinson and S. Wolf. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting*, 50(3):312–322, September 2004.

[55] A. Zisserman R. Hartley. *Multiple view geometry.* Cambridge University Press, UK, 2003.

[56] J.I. Ronda, M. Eckert, F. Jaureguizar, and N. Garcia. Rate control and bit allocation for MPEG-4. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1243–1258, December 1999.

[57] P. L. Rosin. Unimodal thresholding. *Pattern Recognition Letters*, pages 2083–2096, 2001.

[58] P. L. Rosin and E. Ioannidis. Evaluation of global image thresholding for change detection. *Pattern Recognition Letters*, 24(14):2345–2356, October 2003.

[59] J. Guiterez Sanchez. Content analysis for video codec selection for object-based video coding. Diploma thesis, TU Berlin, Berlin, Germany, July 2008.

[60] H. S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2-D & 3-D dominant motion estimation for mosaicing and video representation, Cambridge, MA. In *IEEE Int. Conf. on Computer Vision*, June 1995.

[61] H. S. Sawhney, S. Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *5th European Conference on Computer Vision (ECCV)*, pages 103–119, Freiburg, Germany, June 1998.

[62] H. Schwarz, D. Marpe, and T. Wiegand. Analysis of hierarchical B pictures and MCTF. In *IEEE Int. Conference on Multimedia & Expo (ICME'06)*, Toronto, Canada, July 2006.

[63] J. Shi and C. Tomasi. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[64] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(9):1445–1453, September 1988.

[65] H.-Y. Shum and R. Szeliski. Construction of panoramic mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, February 2000. Erratum published July 2002,48(2):151-152.

[66] T. Sikora. The mpeg-4 video standard verification model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7:19–31, February 1997.

[67] T. Sikora. Trends and perspectives in image and video coding. *Proceedings of the IEEE*, 93:6–17, January 2005.

[68] A. Smolic. *Globale Bewegungsbeschreibung und Video Mosaiking unter Verwendung parametrischer 2D-Bewegungsmodelle, Schaetzverfahren und Anwendungen.* PhD thesis, RWTH University Aachen, May 2001.

[69] A. Smolic and J.-R. Ohm. Robust global motion estimation using a simplified m-estimator approach. In *Int. Conf. on Image Processing (ICIP'00)*, Vancover, Canada, 2000.

[70] A. Smolic, T. Sikora, and J.-R. Ohm. Long-term global motion estimation and its application for sprite coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1227–1242, December 1998.

[71] A. Smolic, Y. Vatis, Heiko Schwarz, and T. Wiegand. Long-term global motion compensation for advanced video coding. In *10. ITG-Fachtagung Dortmunder Fernsehseminar*, Dortmund, Germany, September 2003.

[72] A. Smolic, Y. Vatis, Heiko Schwarz, and T. Wiegand. Improved h.264/avc coding using long-term global motion compensation. In *IS&T/SPIE Symposium on Visual Communications and Image Processing (VCIP'04)*, San Jose, CA, USA, January 2004.

[73] A. Smolic, Y. Vatis, and T. Wiegand. Long-term global motion compensation applying super-resolution mosaics. In *IEEE International Symposium on Consumer Electronics (ISCE)*, Erfurt, Germany, September 2002.

[74] D. Steedly, C. Pal, and R. Szeliski. Efficiently registering video into panoramic mosaics. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[75] G. J. Sullivan and R.L. Baker. Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks. In *Global Telecommunication Conference (GLOBECOM 1991)*, 1991.

[76] G. J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15(6):74–90, November 1998.

[77] G. J. Sullivan and T. Wiegand. Video compression - from concepts to the H.264/AVC standard. *Proceedings of the IEEE*, 13(1):18–31, January 2005.

[78] R. Szeliski. Video mosaics for virtual environment. *IEEE Computer Graphics Appl.*, pages 22–30, 1996.

[79] R. Szeliski and H. Shum. Creating full view panoramic image mosaics and environment maps. In *Proc. of SIGGRAPH*, pages 251–258, 1997.

[80] C. Tomasi and T. Kanade. Detection and tracking of point features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, 1991.

[81] Carlo Tomasi and Takeo Kanade. Detection and tracking of point feature. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[82] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London*, pages 1321–1340, 1998.

[83] B. T. Truong, C. Dorai, and S. Venkatesh. New enhancements to cut, fade and dissolve detection processes in video segmentation. *ACM Multimedia 2000*, pages 219–227, November 2000.

[84] W. H. Tsai. Moment-preserving thresholding: A new approach. *Computer Vision Graphics Image Processing*, 29:377–393, 1985.

[85] T.Wiegand and B. Girod. Lagrange multiplier selection in hybrid video coder control. In *IEEE International Conference on Image Processing (ICIP2001)*, Thessaloniki, Greece, October 2001.

[86] T.Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, July 2003.

[87] Z. Wang and A. C. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9:81–84, March 2002.

[88] Z. Wang, A. C. Bovik, H. C. Sheikh, and E. P.Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[89] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra. Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(2):182–190, July 1996.

[90] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):302–311, July 2003.

[91] S. Winkler and P. Mohandas. The evolution of video quality measurement: From PSNR to hybrid metrics. *IEEE Transactions on Boradcasting*, 54(3):660–668, Sep. 2008.

[92] G. Ye. *Image registration and super-resolution mosaicing*. PhD thesis, University of New South Wales, April 2005.

[93] G. Ye, M. Pickering, M. Frater, and J. Arnold. A robust approach to super-resolution sprite generation. In *Int. Conf. on Image Processing (ICIP'05)*, Genova, Italy, September 2005.

[94] G. Ye, J. Xu, G. Herman, and B. Zhang. A practical approach to multiple super-resolution sprite generation. In *8th Workshop on Multimedia Signal Processing (MMSP'08)*, Cairns, Australia, October 2008.

List of Figures

List of Tables