Advances in Sprite-based Video Coding

Towards Universal Usability

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

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

vorgelegt von
Dipl.-Ing. Matthias Kunter

Promotionsausschuss

Vorsitzender: Prof. Dr.-Ing. S. Möller
1. Gutachter: Prof. Dr.-Ing. T. Sikora
2. Gutachter: Prof. Dr.-Ing. J. Ostermann

Tag der wissenschaftlichen Aussprache: 23.01.2008

Berlin 2008

D 83

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Dr.-Ing. Thomas Sikora, who supported and motivated me during my research as part of the Communication Systems Group at Technische Universität Berlin. I also want to thank Prof. Dr.-Ing. Jörn Ostermann for the detailed reading of my work and for the unforgettable and inspiring research meetings on behalf of 3DTV, a European research network.

Special thanks go to my colleagues at the institute for their encouraging words and deeds, sharing time with me inside and outside the university. I would like to thank Carsten Clemens, Michael Dröse, Jangheon Kim, Sebastian Knorr, and Andreas Krutz for coauthoring numerous publications with me. Spending long evenings together in order to meet the particular deadline was a great experience. The unsurpassed sense of humor of George Haboub, who I shared office with, cheered me up during gray winters.

This work would not have been the same without the help and commitment of numerous students working on their diploma or master thesis under my supervision. The complete list would be way too long, but in particular, I would like to express my gratitude to Athalie Tchikangoua Toda and Philipp Krey, whose work is referenced in this thesis.

I thank my good friend Tobias Klauß for his power surplus and for overlooking the one or the other knight fork. Finally, my greatest thanks go to three amazing women. To my grandmother Gerda, who makes the best potato salad ever. To my mother Annelie for all her support and love. And, above all, to Berit for constantly ignoring the fact that our world is flat.

Abstract

This dissertation presents new approaches end extended techniques for the coding of digital video using background sprites, also called background mosaics. Sprites form a visual summarization of the rigid background of a captured scene shot. They are represented in oversized images, which preferably do not contain any foreground objects. This type of redundancy reduction is an ideal tool for video coding since the complete background information can be stored in the sprite image and some additional projection parameters. However, the generation of sprites is only possible for certain scenes. Since a successful coding strategy has to be universally applicable, the development of techniques for facilitating a broader use of sprite-based video coding represents the main focus of this thesis. Early approaches, as the one adopted in the MPEG-4 standard, have not been utilized due to the lack of universality and usability.

For this purpose, we present techniques for the generation of multiple sprites and provide automatic segmentation approaches for the independently moving foreground objects. While multiple sprites prevent the construction of degenerated sprites and simultaneously minimize the impact of geometrical distortions, the segmentation enables the automatic discrimination in foreground and background objects. Thus, it is a fundamental tool for object-based video coding. The presented segmentation techniques are built upon the background sprites and thus, are easy to integrate into the overall coding process.

The improvement of the background modeling using sprites marks another important aspect of this dissertation. Since state-of-the-art hybrid coding strategies work very efficient and yield high quality results, the prediction quality of the background using sprites has to be improved remarkably. In order to achieve this goal, we present novel image registration and sprite generation algorithms. Especially the potential of super-resolution processing will be exploited. Due to the capturing process, we obtain several differently sampled versions of the same image content. This fact can be used for the construction of background sprites of enhanced resolution, which has a positive influence on the resulting coding quality as well as on the rate-distortion results.

Eventually, two techniques for sprite-based video coding are presented. Both approaches utilize above mentioned tools for improving quality and universality of the sprites. The coding gain over latest standards proves their usefulness. A complete coding system for

the processing of any video content is still not achieved, but an outlook of its possible architecture is drafted. Thus, this thesis contributes to a gradual change of the video coding paradigm, where additional instruments from computer vision and computer graphics are utilized to unequally encode independent parts of a video scene.

Zusammenfassung

Diese Dissertation präsentiert neuartige Ansätze und erweiterte Techniken zur Kodierung digitaler Videos mittels Hintergrundsprites bzw. Hintergrundmosaiken. Sprites stellen visuelle Zusammenfassungen des starren Hintergrunds einer aufgenommenen Szene dar. Sie werden in übergroßen Bildern repräsentiert, welche möglichst keine Vordergrundobjekte enthalten. Diese Art der Redundanzreduktion ist ideal für die Videokodierung, da die gesamte Hintergrundinformation in dem Sprite und einigen wenigen Projektionsparametern enthalten ist. Die Erstellung von Hintergundsprites ist allerdings nur für ganz bestimmte Szenen möglich. Da eine erfolgreiche Kodierstrategie vor allem universell, also für jede Art von Videomaterial einsetzbar sein soll, liegt das Hauptaugenmerk dieser Arbeit auf der Entwicklung von Techniken, welche die Möglichkeiten der spritebasierten Videokodierung erweitern. Erste Ansätze, wie sie schon im MPEG-4 Standard festgelegt wurden, finden aufgrund des Mangels an Universalität in der Praxis leider keine Anwendung.

Hierfür werden in dieser Arbeit vor allem Verfahren zur multiplen Spritegenerierung und zur automatischen Segmentierung der sich unabhängig bewegenden Fordergrundobjekte vorgestellt. Während multiple Sprites die Erzeugung degenerierter Sprites verhindern und gleichzeitig den Einfluss geometrischer Verzerrungen bei der Projektion minimieren, ermöglicht die Segmentierung eine automatische Trennung von Hintergrund und Vordergrund und bildet somit eine wesentliche Grundlage zur objektbasierten Videokodierung. Die hier vorgestellten Segmentierungstechniken bauen direkt auf bereits erstellten Hintergrundsprites auf und lassen sich daher sehr gut den Gesamtprozess der Kodierung integrieren.

Einen weiteren wichtigen Aspekt dieser Dissertation bildet die qualitative Verbesserung der Hintergrundmodellierung mittels Sprites. Da moderne hybride Kodierverfahren sehr effizient sind und qualitativ hochwertige Ergebnisse liefern, muss die Qualität herkömmlicher spritebasierter Prädiktion des Hintergrunds erheblich verbessert werden. Hierfür werden exakte Bildregistrierungs- und Spritegenerierungsverfahren vorgestellt. Dabei wird vor allem die Möglichkeit der Superresolution ausgenutzt. Da die Diskretisierung der visuellen Information bei der Videoaufnahme mehrfach und mit verschiedenen Abtastmustern erfolgt, können mehrere Bilder einer Szene zur Erhöhung der Auflösung des Sprites genutzt werden. Dies wirkt sich positiv sowohl auf die Kodierqualität als auch auf das Verhältnis von Qualität zu Übertragungsrate aus.

Letztendlich werden zwei Techniken zur spritebasierten Videokodierung vorgestellt, welche die präsentierten Ansätze zur verbesserten Universalität und Qualität der Hintergrundmosaike verwenden. Der erreichte Kodiergewinn gegenüber neuesten Standardkodierverfahren bestätigt dabei die Nützlichkeit dieser Methoden. Eine vollständig universelle Kodierung jeglichen Videomaterials kann damit noch nicht erreicht werden, ein möglichst vollständiges System hierzu wird aber bereits skizziert. Diese Arbeit liefert somit einen Beitrag zu einem allmählichen Paradigmenwechsel in der Videokodierung, bei dem in zunehmendem Maße Werkzeuge der Computer Vision und der Computergrafik benutzt werden, um verschiedene Teile eine Videoszene unterschiedlich zu kodieren.

Contents

Chapter 1

Introduction

Science must begin with myths, and with the criticism of myths.

Karl R. Popper (1902-1994)

In July 1972 the U.S. National Aeronautics and Space Administration (NASA) launched one of the first earth observation satellites into orbit [5]. The so-called Earth Resource Technology Satellite (ERCTS-1, Landsat-1) was designed to enable an exact observation of the earth applying different types of sensors and to push the scientific frontiers for geography, geology, and related earth sciences (see Figure 1.1). Today, this date can be regarded as the first modern application of digital image stitching and image mosaicing systems. Thenceforward, the scientific community was enabled to obtain oversized images of the earth surface computed fully automatically by digital microprocessors. Of course, at this time the combination of images and photographs to huge maps had a history of more than 100 years being strongly connected with the field of Panoramic Photogrammetry [84]. But preceding projects were mainly based on manually aided methods, e.g., the 1947/48 ANARE photogrammetric map project of the Australian Government Antarctic Division. Today's earth observation maps do not only consist of fused single images but also have enhanced resolution. Figure 1.2 schematically shows the step from 1960's simple photograph (Weather satellite TIROS-1) to 2005's earth image mosaic result (Blue Marble) utilizing one years observation with NASA satellites Terra and Aqua.

With the rapid development of digital visual recording and storage systems and its dissemination into consumer applications in the 1990's the field of digital image mosaicing attracted more and more attention of computer vision and computer graphics researchers, digital artists, and telecommunication specialists. Today's photo processing software supports easy-to-use panoramic image generation tools that enable the user to create very large wide angle panoramas obtained from several digital photographs. The step from still images to moving pictures then seems to be quite logic. The generation of video mosaics summarizing the rigid background into one panoramic image constitutes a powerful tool in many areas of image and video processing and its applications for communications, medicine,

surveillance, authoring, and content creation. See Figure 1.3 for a very popular example of video mosaicing achieved by U.S. mars robot Opportunity.

Video background mosaics, also known as sprites, are a very fascinating research topic for two main reasons. On the one hand, the technique for fusing all static parts of a scene shot recorded by a moving camera into a combined oversized image corresponds substantially to the way the human visual perceptional system processes the reproduction of the natural surroundings. Even irrelevant movements of background objects can be filtered out while constructing video sprites. On the other hand, the more important fact for video communication is that sprite processing allows a very effective reduction of temporal redundancies. In other words, certain video contents being represented in a frame-based manner and thus by a huge amount of data can be reduced to a representation by only one image and a small amount of projection parameters. For those scenes sprite construction is a very clear and efficient method to simultaneously reduce redundancy and irrelevancy, which is a perfect condition for application in video data compression tools.

Above this inherent property the possibility of creation of super-resolution mosaics (SR) can be exploited. As a side effect of image registration and alignment, e.g., the geometric model-based connection of two ore more images in a scene shot, one can enhance the spatial resolution of the final mosaic due to multiple independent samples of the same content. Moreover, super-resolution sprites improve the background reconstruction quality of a scene and therefore also improve the compression and coding efficiency. Also for spatial video upsampling SR mosaics can be utilized.

With this thesis I will present the results and outcomes of my research and research corporations at Technische Universität Berlin as part of the Communication Systems Group. Since the main research goals in this area are effective content representation, advanced compression techniques, and sophisticated analysis of digital visual information, I am presenting in this work my contributions to the field of mosaic or sprite-based video analysis and video coding.

1.1 Motivation

Despite the huge success sprite and mosaic processing techniques have made during the last ten years, the following very important problem motivates researchers to put a lot more efforts into this field of research. It is quite astonishing that there does not exist any commercial or non-commercial video coding software that fully integrates sprite coding as an object-based mode even without generation of the background sprite itself. Just the handling as independent video object layer and a joint coding with other video objects seems to be too difficult. The question that arises is: what are the problems like that automatic coding and video analysis tools applying sprite-based techniques are confronted with? Why

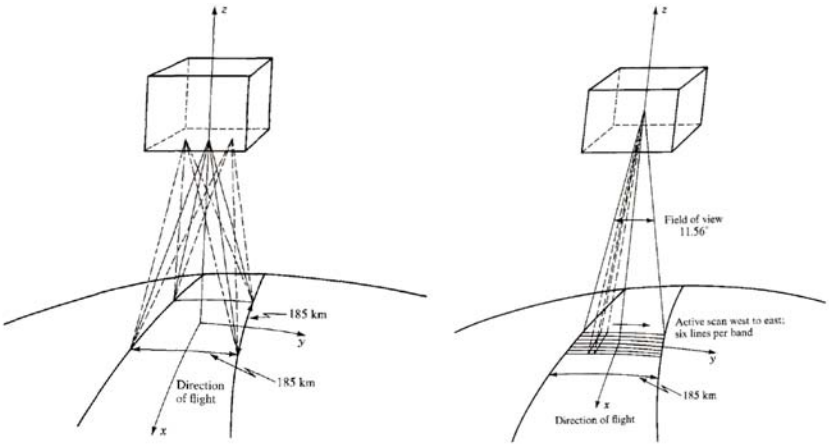


Figure 1.1: Sensors of ERCTS-1 for earth observation; Left: Return Beam Vidicon (RBV) camera; Right: Multispectral Scanner (MSS) [5]

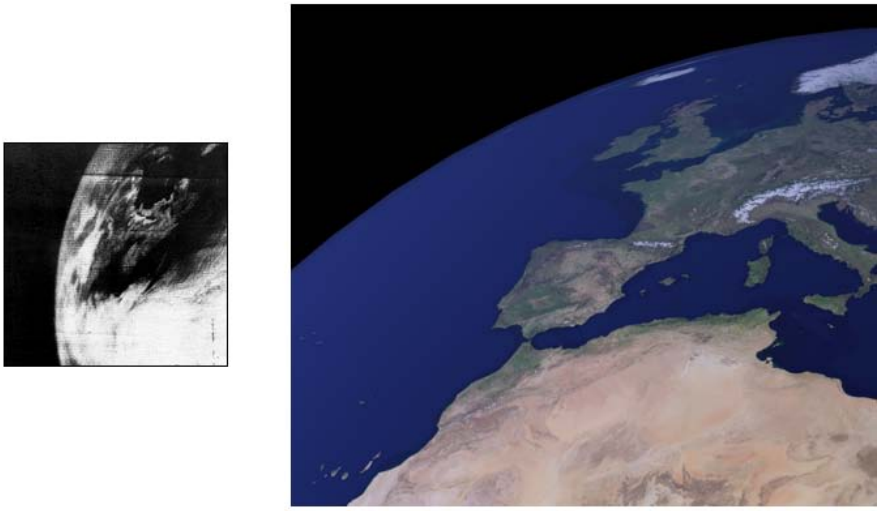


Figure 1.2: Earth observation: From low resolution photographs (1960) to high resolution mosaics (2005)

is it impossible to find matured systems combining the advantages of sprite methods with other well established digital video processing applications?

The answer to these questions can be found if one turns to the issues of generality and universality. As in common hybrid video codecs or several other signal analysis tools many technologies become widely accepted if their applicability is universal within a certain context. That means, applications based on sprite and mosaicing methods either have to be utilizable for the majority of video input signals or have to inherently classify in order to determine if the use of sprites is beneficial or not. Unfortunately, these directions of research lack in a wide range computer vision problems.

Hence, the motivation for this work is twofold. On the one hand this thesis aims to explore and significantly improve the possibilities and applications based on sprite generation - especially the video coding aspect - and to push the scientific frontiers mainly by exploiting

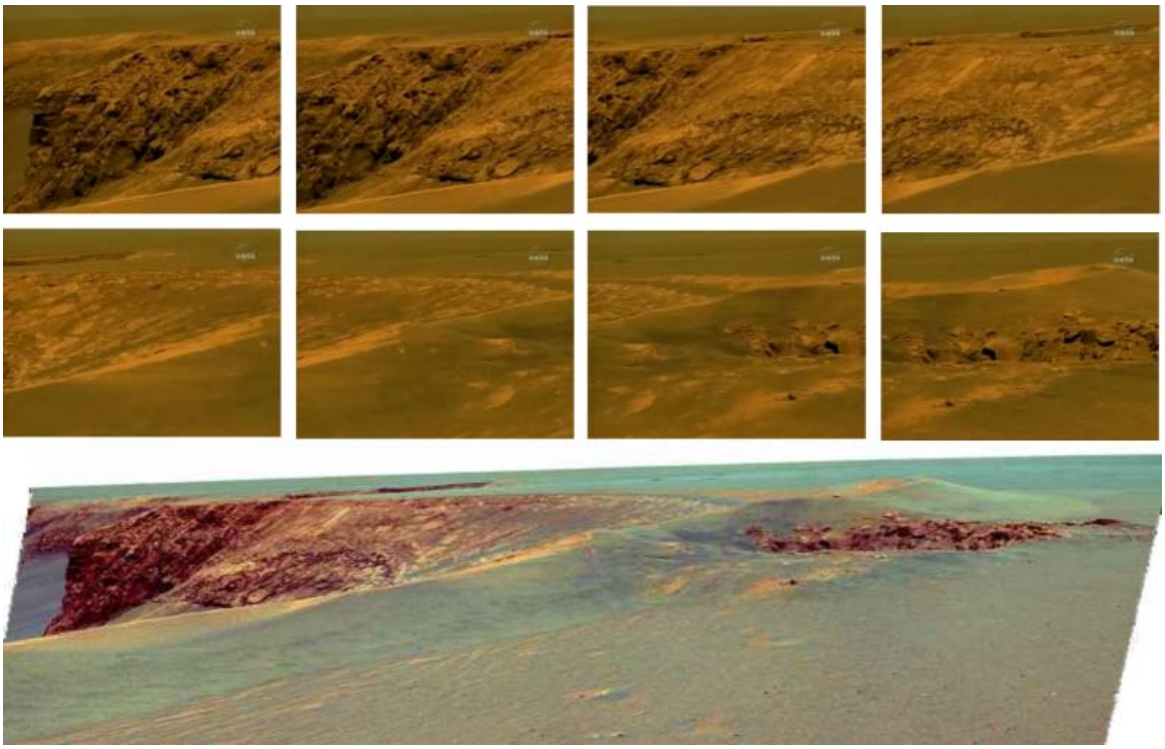


Figure 1.3: Eight frames of a video recorded by mars roboter "Opportunity" and the constructed sprite (Source: NASA, AFP)

the super-resolution approach. On the other hand it is the declared objective of this work to determine the limits of sprite-based methods and to investigate which additional algorithms and tools are necessary in order to make sprite processing, e.g., in object-based video codecs, more applicable in terms of a general use.

The following points outline the fundamental motivations for this thesis

- Improving exact sprite generation by exploiting the aspects of super-resolution using enhanced registration and blending techniques
- Optimal background estimation for scenes with arbitrarily moving foreground objects
- Investigation of the applicability of sprite-based methods for different video material such as:
  - Different recording setups/ capturing camera motions and
  - Temporal video classification
- Determination of the limits of sprites with respect to the sprite image size and the underlying registration fundamentals, e.g., mappings and image transformations
- Exploration of main applications:
  - Video segmentation
  - Video coding

1.2 Literature Survey

The literature on methods and tools for generation, usage, and coding of video background sprites is very vast. Since the early 1990's scientists of different research communities have explored and expanded this topic. It is obvious that at this point only the main contributions and most interesting steps of the development will be mentioned. Nevertheless, there is a much bigger number of researchers in this field who tackled the problems connected to this area and proved several results. Also only a superficial overview of the applied technology will be depicted. The interested reader may refer to the referenced articles in order to get an in-depth description of the applied techniques.

In 1991, M.-C. Lee, W. Chen, C. B. Lin, C. Gu, T. Makoc, S. Zabinsky, and R. Szeliski published a first article on sprite generation using an affine motion model and described its application to video coding [73]. But it was not until 1994 when the well-known report of R. Szeliski triggered a new period of research efforts especially among the Computer Vision (CV) and Pattern Recognition (PR) scientists [140]. In his article he gives a detailed presentation of the applied image alignment method and describes the approach of image and video mosaic construction. An improved version of this approach was later printed in [141] and [142] and additionally expanded to panoramic images by H. Shum and R. Szeliski [122]. In this early publications independently moving foreground objects in a scene shot were not considered. Thus, the only existing motion was due to camera movements. Also in 1991 M. Irani published an article on image registration and sprite resolution improvement [52]. In later works she applied her technique to create mosaics, being used for coding, efficient video representation, video indexing, as well as for temporal resolution enhancement [51], [50], [49], [10].

Another disadvantage of early publications on sprite and mosaic generation and representation issues was the lack of result assessment or quality measurement. Since authors used their own sequences and only visually depicted the automatically generated sprite images, it was very difficult to compare different approaches [140], [47], [50]. With the emerging video coding standard MPEG-2 in 1994 the delivered test sequences enabled researcher for the first time to compare their results at least subjectively. But also an objective assessment of the re-projected sprite content into every single frame coordinate system became possible [132], [18]. Those communication induced development of sprite techniques got a boost in 1999 when sprite coding was taken into account for the standardization process as part of MPEG-4 Visual in the Main Profile [126], [101]. In 1999, A. Smolic et al. presented their approach for hierarchical global motion estimation based on robust computation of the parameterized optical flow applying gradient descent methods. They additionally presented an efficient coding strategy for background and foreground objects not taking care of the segmentation process itself [133], [129].

In 1995 and 1999, Sawhney et al. published papers on motion estimation [113] and mosaic generation [114] where especially the lens distortion effect during the scene recording was explored. Other authors who introduced improved image registration for sprite image construction are F. Dufaux [18] who proposed fast and robust image pyramid-based global motion estimation, H. Nicolas [93], and G. Ye [164], [165].

Several scientists investigated the impact on and results of sprite-based coding techniques. K. Jinzenji, H. Watanabe, S. Okada, and N. Kobayashi showed that for certain ratios of foreground areas to background areas sprite coding does not outperform conventional hybrid video codecs at all [55]. All in all only very view publications present results applying sprite coding and show the effectiveness for lower bit-rates in expressive rate-distortion plots. Y. Lu, W. Gao, and F. Wu proposed an effective offline sprite generation scheme [76], [77], [77], which was proposed as video tool for sprite generation at the MPEG meeting in January 2001 [80]. They also investigated very efficient sprite coding techniques applying arbitrary-shape spatial prediction techniques [78].

Concerning the field of 3D scene reconstruction and the feature-based estimation of image transformations the work of D. Capel and A. Zisserman has to be mentioned. In their contributions robust tracking of feature and corner points as it is usual in scene reconstruction form the base of the mosaicing approach [8], [9], [42]. The utilization of background sprites for depth reconstruction and coding for multiple view sequences was mainly introduced by N. Grammalidis in 1999 [39], [40]. He addressed the construction of background mosaics from scenes captured by multiple cameras. In his work, the author exploits the disparity and occlusion information to add image content from all available views to the background sprite [38]. Thus, e.g., occluded regions of a sprite can be filled by visual information from the other views. Additionally, the sprite can be expanded by content only visible in other views.

In recent publications the research regarding sprite generation follows two directions. On the one hand the processing of scenes with arbitrarily moving foreground objects without any information of their shape is exploited. The generated sprites itself can be used for effective foreground segmentation. This way of motion-based segmentation was intensively studied by D. Farin [30], A. Fusiello et al. [34], and S.-Y. Chien et al. [13]. Since the background mosaic can be used to compute a foreground-free video frame, classical change detection algorithms applying image differences with the original frame sequence yield very exact results. The second research direction is the field of super-resolution. Due to the existence of multiple independent samples with different sample patterns, the spatial resolution of a constructed sprite can be improved. In an early article on multi-resolution mosaicing, C.-T. Hsu and J.-L. Lu describe their technique of robust image blending and mention several applications [47]. Interesting contributions in this field were also made by Marzotto et al. [83], A. Smolic [136], B. Tannenbaum [144], and G. Ye [164].

A very important and new topic for sprite research is the optimization of the two-dimensional mosaics with respect to the sprite size and the re-projection quality. In his awarded paper D. Farin [29] proposed an universal method to part a sequence into several subsequences in order to generate multiple sprites, which minimizes the memory cost for the video background [29], [27], [26]. This non-heuristic approach was a huge step towards efficient sprite representation and a novelty compared to former approaches by Chien at al. [14].

1.3 This Work: Objectives, Novelties and Principal Contributions

This thesis presents new approaches to image registration, super-resolution mosaicing and optimal sprite partitioning as well as their applications for video coding. The objectives of this work are described in the following section.

1.3.1 Objectives

Starting from the geometrical fundamentals of the mapping of real world data onto the camera sensor in this thesis we will derive the appropriate physical-based image transformation models that describe the relation of two images and build up the base for mosaicing algorithms. In order to explore the limitations that the single transformation models provide an in-depth analysis of the applicability for several types of camera recordings - with respect to the camera movements - has to be processed. Additionally, the relation between transformation models and physical camera parameters, i.e., camera calibration [42], will be derived.

One main objective of this work is to present an improved single sprite construction algorithm, which is more reliable in terms of background re-projection than already proposed techniques. That means, established registration methods have to be improved or partially substituted by other algorithms, providing more exact transformation computation by defining novel robustness constraints or applying computational more powerful energy minimization frameworks. Since differential methods comprise the best performance while determining the optimal mapping between two images, it is very important to compute the initial mapping as starting point for those gradient descent methods. A survey and comparison of different approaches for initial parameter estimation will be given.

Incorporating super-resolution (SR) methods into the sprite generation process is a logical extension of classical mosaicing algorithms. Super-resolution makes use of multiple statistically independent sampling patterns of the same signal, i.e., the visual content [172], [64], [99]. Since this case is a precondition for the ideal camera setup (only

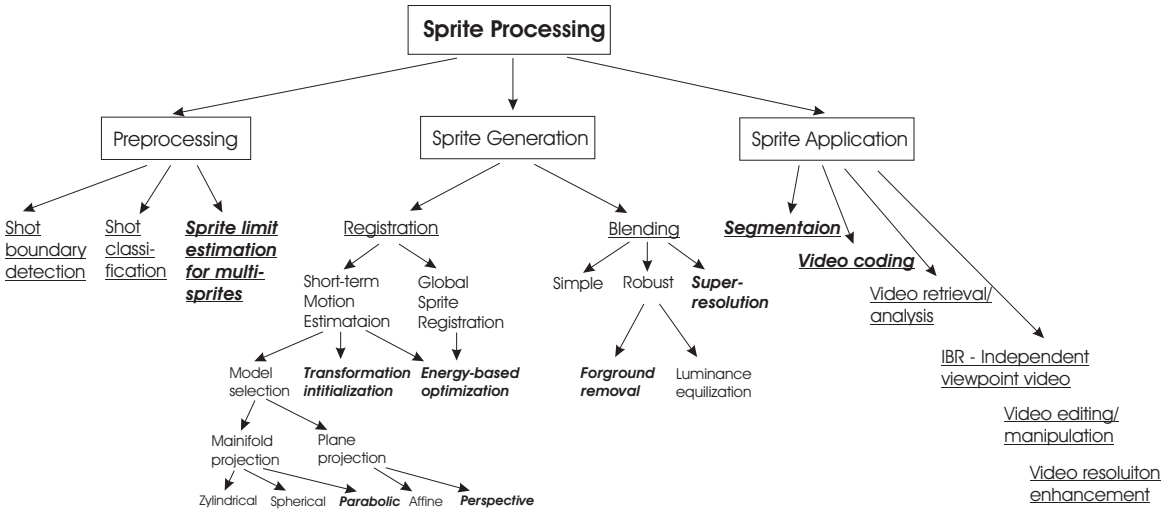


Figure 1.4: Overview of main research topics for universal sprite processing systems: The bold italic tasks are main objectives of this thesis.

rotating camera or plane sources), sprite blending algorithms can theoretically and practically enhance the resolution of the computed sprite due to an overcome of the Shannon limit for single sampling.

A further objective of this work is the investigation of limitations the proposed sprite generation process is subject to. Due to geometrical properties of the image registration, not all types of camera setups and video scenes can be applied to construct background sprites. Moreover, for several types of video the sprite construction will fail even with very robust approaches. Furthermore, a sprite image can grow towards infinity depending on the underlying camera motion. Here we derive a novel multiple sprite construction method which parts a shot into several sub-shots to obtain an optimal number of sub-sprites providing better reconstruction quality and lower memory costs.

Possible applications of conventional and super-resolution sprites, which mainly lie in the field of video coding and video segmentation, constitute an essential part of this thesis. In both research topics new approaches are presented.

1. The coding strategies are based on the latest extensions of the 2003 emerged H.264/ AVC video coding standard standardized by the Joint Video Team (JVT) of MPEG and ITU [160], [110]. The presented coding approaches differ significantly from the object-based video codec proposed in MPEG-4 Part 2.

2. For the segmentation part, advanced change detection methods are used based on the difference between the estimated background pictures and the original picture.

In Figure 1.4 a survey of research topics for a complete sprite system is depicted. It is clear that not all of these tasks can be fully investigated since it would go too far beyond the scope of this work. Therefore, the areas of main interest for the following chapters are marked as bold italic.

1.3.2 Principal Contributions and Novelties

The principal contributions and most important results of this work in hand are:

- A compact overview of the underlying geometrical fundamentals
- An in-depth derivation of fundamental 2D image transformations based on physical properties of camera recordings
- A theoretical examination of limitations of those 2D image transformations
- Survey of image registration approaches to estimate the optimal transformation between two images
- A new energy minimization framework for difference-based image registration applying Levenberg-Marquardt Minimization
- A low-complex estimation technique for the physical camera parameters based on previously computed short-term motion parameters
- Analysis of different parameter initialization techniques
- Development of an direct registration approach for global long-term frame-to-mosaic registration
- Overview of and contribution to robust image blending algorithms to optimally remove independently moving foreground objects
- An introduction to spatial super-resolution techniques
- A new approach of super-resolution sprite generation where temporal robustness as well as minimal image interpolation constraints are integrated
- Analysis of limits of super-resolution image construction in comparison to sophisticated image interpolation
- A new method to construct multiple sprites in an optimal way that memory cost are minimized while the background reconstruction quality is improved
- Development and analysis of two sprite-based object segmentation algorithms based on change detection with background subtraction
- Two new coding strategies for scene shots with freely moving objects recorded with rotating camera based on static sprites

To break down the essence of my work the following points build the main scientific novelties:

- The super-resolution approach for sprite generation

- The multiple sprite technique to overcome construction limitation and minimize memory cost
- The application of foreground segmentation using background mosaics
- H.264/AVC-based coding techniques without a need for pre-segmentation of video-shots

1.3.3 Organization of this Thesis

In the next chapter, geometrical fundamentals for the derivation of the appropriate 2D image transformations will be presented. These transformations build the basis of the sprite construction process. The utilization of projective geometry eases the algebraic handling of the transformations. Therefore, a brief introduction into projective geometry is given. We will as well investigate the necessary capturing conditions sprite construction can be applied for and present the geometrical limitations of plane mosaicing techniques. Since the geometrical process of image projection is closely related to the actual physical camera parameters, e.g., orientation angles and focal length, we will present a low complexity estimation algorithm in order to determine these parameters.

The generation process of background sprites for scenes captured with fixed but rotating camera is explained in Chapter 3. Starting with the process of 2D image registration we will derive a hierarchical short-term parameter estimation that registers temporal adjacent frames of an utilizable scene shot. It uses photometric consistency-based energy minimization and feature-based registration techniques. In order to construct a full sprite, long-term registration has to be performed. Here we present a novel direct frame-to-mosaic registration technique. Finally, a reliable background is estimated using sophisticated blending methods.

The exploitation of a process called super-resolution sprite generation is the topic of Chapter 4. Here, the redundancy of image contents and the different sampling patterns for the particular frames are used to generate sprites of higher resolution. A new low complex super-resolution technique is presented that especially fits into the sprite generation process of Chapter 3. Super-resolution sprites not only provide a higher subjective visual quality but also enable the user to reconstruct the frame backgrounds with much higher objective quality. This makes this technique very attractive for further sprite-based video processing algorithms, e.g., sprite-based video coding.

In Chapter 5, we present a solution for nonlinearly growing sprite memory costs when camera rotation angles – pan and/or tilt – become to wide. These inherent mosaic construction limitations can be overcome by generating multiple sprites. The presented algorithm builds those multiple sprites in an optimal way, which provides equal or even better reconstruction quality and minimizes the image sizes. Thus, the ratio between sprite quality and sprite memory is improved.´

A very useful application for sprites is the motion-based object segmentation. This segmentation process divides the scenes into rigid background objects, being part of the sprite, and independently moving foreground objects. The sprite-based re-projection of background content already provides high quality background estimation for any frame in a shot. Therefore, change detection techniques based on background subtraction promise a good solution of the segmentation task. In Chapter 6 two novel methods of sprite-based segmentation are presented. In a comprehensive evaluation the excellent performance of these methods is approved.

In Chapter 7, the ideas and results of all previous chapters in this thesis are brought together. Thus, it can be regarded as the synthesis of this work. The chapter addresses the video coding of sequences utilizing sprites for background prediction. Since video codecs (coders/ decoders) are very complex systems and have been studied and used for more than 15 years, a performance improvement over actual standards can only be achieved applying a combination of exact and sophisticated methods. Hence, super-resolution sprite coding as well as multiple sprite coding with embedded object segmentation will be applied. For low bit rates, standard video codecs can be outperformed by far. We present two coding approaches that are based on the latest ITU/ISO-MPEG standard H.264/AVC. Both methods differ in their segmentation handling and have a very dissimilar coding performance. In the experiments section the merits and demerits of both methods are intensively evaluated.

Chapter 8 concludes this thesis and underlines the achieved findings in terms of universal usability and compatibility. In addition, we will present an outlook to future sprite coding systems and briefly introduce into other applications of sprites, already investigated by the author, that go beyond the scope of this thesis. Mathematical details and an overview of all sequences, that were used for the evaluation of the proposed methods and algorithms, are presented in Appendix A and Appendix B, respectively.

Chapter 2

Optical Mappings, Camera Setups, and geometrical Fundamentals

'Αγεωμέτρητος μηδεὶς εισίτω. - Ageōmetrētos mēdeis eisitō.
- Let no-one without knowledge of geometry enter.

Motto over the entrance to Plato's Academy

In this chapter, the geometrical and mathematical basis for later utilized image registration approaches is presented. Since the projective geometry and its notation is a very elegant tool for the description of mappings and algebraic geometrical relations [42], I will first give a short introduction into this specific area of geometry. Starting from the mapping of real world data onto a camera sensor (using the pinhole camera model) the relation between points on different camera views will be derived. These functionalities are then used to deduce the appropriate 2D mappings which build the bases for all sprite generation algorithms. The limitations of the mappings, also referred as image transformations, are then analyzed and derivations from the physical model and their impact on the accuracy will be researched.

Image mosaicing is a way of modeling the real world environment. Because it is achieved by exclusively processing a sequence of photographs or images it belongs to the class of image-based modeling or image-based rendering techniques (IBR) [124]. From an exact point of view also simple video sequences form a model of the environment. On the other side, there exist very sophisticated methods of image-based 3D modeling. These methods, more related to geometrical properties, try to reproduce the real 3D geometry. On a scale between geometry-based and image-only-based modeling, sprite methods can be located somewhere in the middle. Figure 2.1 shows an overview of models published in [107].

Aligning all frames of a scene into one 2D-coordinate systems of a manifold [100] presumes an adequate mapping of the pixels of the views. Describing this mapping by just one vector for each pixel in all views would be a possible solution. Unfortunately this method is not applicable for several reasons. First, it is an ill-posed problem to compute the exact vector field (also known as *optical flow*). Second, the amount of data to describe the model would

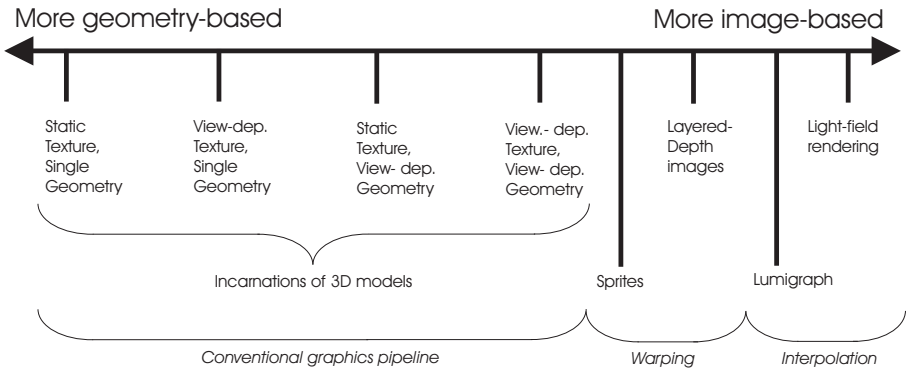


Figure 2.1: Classification of image-based models for a natural environment [107]

not be reduced compared to the original data. It is one advantage of video models that the redundant data will be reduced. Moreover, the creation of new views by re-projecting the model into a virtual camera would be very difficult as well. Therefore, it is inevitable to derive a parameterized mapping for image alignment. Since it is desirable to have a very realistic reproduction of original images from the model we derive the parameterization from the physical view. In order to do so, it is very useful to recapitulate the basics of *Projective Geometry*.

2.1 Projective Geometry and the Pinhole Camera

For the description of elementary relations of structures in 3-dimensional space usually Euclidean geometry is utilized. It has the advantage that it is very close to how humans intuitively perceive their environment. Here points are simply represented by a vector $P \in \mathbb{R}^3$. The single entries of these vectors are linear weights of an orthonormal basis. It is easy to describe simple point relations or directions in that space. Unfortunately, it is more difficult to describe relations and directions between points that form one or several rigid bodies. Thus, keeping algebraic operations simple – resulting from elementary operations in the Euclidean space – is a difficult task.

Projective Spaces provide a good solution for the algebraic description problem. Following [31] there are four main advantages using a projective framework for certain computer vision specific tasks:

- A typical camera is a projective engine itself, i.e., projective geometry is the correct model to derive the mappings from 3D points onto the camera sensor.
- Projective Geometry provides a much simpler algebra.
- The number of special cases, e.g., singularities, is smaller.

- The projective framework unifies several other geometric concepts like Affine and Euclidean geometries.

In order to ease the reading of mathematical relations, I will not strictly stick to the projective algebra. Moreover, it is helpful to switch between the descriptions for optimal simplification. For distinction of Euclidean and projective space throughout the next sections, points of Euclidean space will be written with a *tilde*, e.g., $\tilde{\mathbf{x}}$.

2.1.1 Projective Geometry Fundamentals

A Projective space $\mathbb{P}^n$ is defined by the following equivalence relation. Two points $\mathbf{x_1} \in \mathbb{P}^n$ and $\mathbf{x_2} \in \mathbb{P}^n$ are equivalent if and only if:

$$\forall \lambda \in \mathbb{R} \backslash \{0\} : \ \mathbf{x_1} = \lambda \mathbf{x_2} \tag{2.1}$$

Note, that vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ have $(n+1)$ entries using the representation in *homogeneous coordinates*. For the recovery of the Euclidean coordinates we apply

$$\mathbf{x} = (x_1, \ldots, x_n, w)^T \longmapsto \tilde{\mathbf{x}} = \left(\frac{x_1}{w}, \ldots, \frac{x_n}{w}\right) \in \mathbb{R}^n, \tag{2.2}$$

where w is a free scaling parameter and describes the projection along one dimension of $\mathbb{P}^n$. Usually the last entry of each vector in projective geometry represents the scaling parameter. Thus, each point in $\mathbb{R}^n$ is represented by an equivalence class in $\mathbb{P}^n$. An interesting case is when $w = 0$. This class of points represents all the points at *infinity*. Therefore, vector $(x_1, \ldots x_n, 0)^T$ represents a point at infinity with the direction $(x_1, \ldots x_n)^T$ in Euclidean space.

The conditions in the 2D- projective space $\mathbb{P}^2$ are very easy to understand and explain why projective geometry can be ideally applied for camera projections. Let $\mathbf{x} \in \mathbb{P}^2$ with $\mathbf{x} = (x_1, x_2, w)^T$ be a point in projective space. Actually it is helpful to figure $\mathbf{x}$ as a ray in Euclidean space $\mathbb{R}^3$. Then $\tilde{\mathbf{x}} \in \mathbb{R}^2$ with $\tilde{\mathbf{x}} = (x_1/w, x_2/w)^T$ is a point at Euclidean plane $x_3 = 1$ intersecting with a line defined by direction vector $\mathbf{n} = \lambda\,(x_1, x_2, w)^T$, with $\|\mathbf{n}\| = 1$ through the origin in $\mathbb{R}^3$ (See Figure 2.2 for illustration).

Rotation and Translation

Translating points in Euclidean geometry means simple vector addition. Let $\tilde{\mathbf{x}}$ be a point in Euclidean space and $\tilde{\mathbf{t}}$ a translational vector to add. In Euclidean coordinates follows

$$\tilde{\mathbf{x}}_{trans} = \tilde{\mathbf{x}} + \tilde{\mathbf{t}}. \tag{2.3}$$

Rotating a point about the origin of the world coordinate system means a multiplication with a rotation matrix $\mathbf{R}$. Thus, rotation can be written as

$$\tilde{\mathbf{x}}_{rot} = \mathbf{R} \cdot \tilde{\mathbf{x}}. \tag{2.4}$$

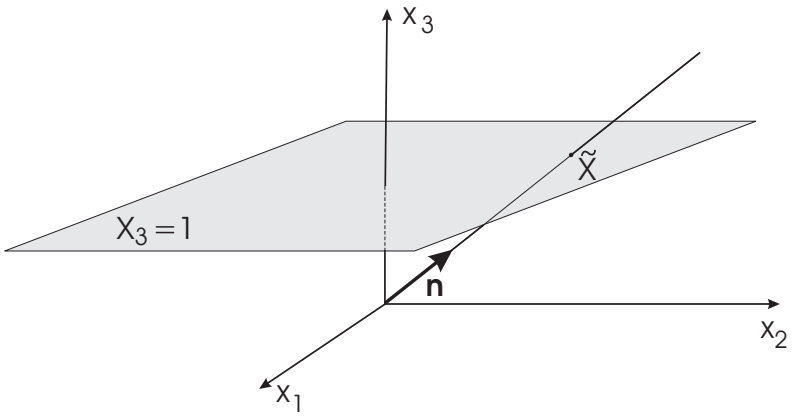


Figure 2.2: On the illustration of relation between points $\tilde{\mathbf{x}}$ in $\mathbb{R}^2$ and $\mathbf{x}$ in $\mathbb{P}^2$

A combination of both processes yields in successively executing Equation (2.3) and Equation (2.4). Applying several translations and rotations would make the handling very difficult, especially when the overall transformation is inverted. Considering projective geometry eases the process enormously. Let $\mathbf{x}$ be the presentation of $\tilde{\mathbf{x}}$ in projective space with $\mathbf{x} = \left(\tilde{\mathbf{x}}^T, 1\right)^T$. Then adding a vector $\tilde{\mathbf{t}}$ becomes

$$\mathbf{x}_{trans} = \begin{bmatrix} \mathbf{I}_{n\times n} & \tilde{\mathbf{t}} \\ 0_n^T & 1 \end{bmatrix} \mathbf{x} \tag{2.5}$$

and rotation yields in

$$\mathbf{x}_{rot} = \begin{bmatrix} \mathbf{R} & 0_n \\ 0_n^T & 1 \end{bmatrix} \mathbf{x}, \tag{2.6}$$

where n is the dimension of the underlying space usually set to 3 for operations in 3-dimensional Euclidean space and 0_n represents the n-dimensional null vector. $\mathbf{I}_{n\times n}$ represents the identity matrix. Therefore, geometric Euclidean transformations can algebraically be accomplished applying simple matrix multiplication in projective space.

Points and lines in $\mathbb{P}^2$

Using projective geometry, a line $\mathbf{l} \in \mathbb{P}^2$ is defined by the following equation:

$$\mathbf{l}^T\mathbf{x} = 0. \tag{2.7}$$

Hence, all points $\mathbf{x} \in \mathbb{P}^2$ lying on a line result in a vector product of zero. Since in Equation (2.7) line $\mathbf{l}$ and points $\mathbf{x}$ can be swapped, we have a duality between points and lines in $\mathbb{P}^2$. That means, each 3-tuple in $\mathbb{P}^2$ stands for a point or a line. Note, that rewriting Equation (2.7) follows in the equation for a line in $\mathbb{R}^2$

$$\mathbf{l}^T\mathbf{x} = (a, b, c) \cdot (x_1, x_2, 1)^T = ax_1 + bx_2 + c. \tag{2.8}$$

In higher dimensions $\mathbf{l}$ would form the hyperplane of the corresponding Euclidean space $\mathbb{R}^n$.

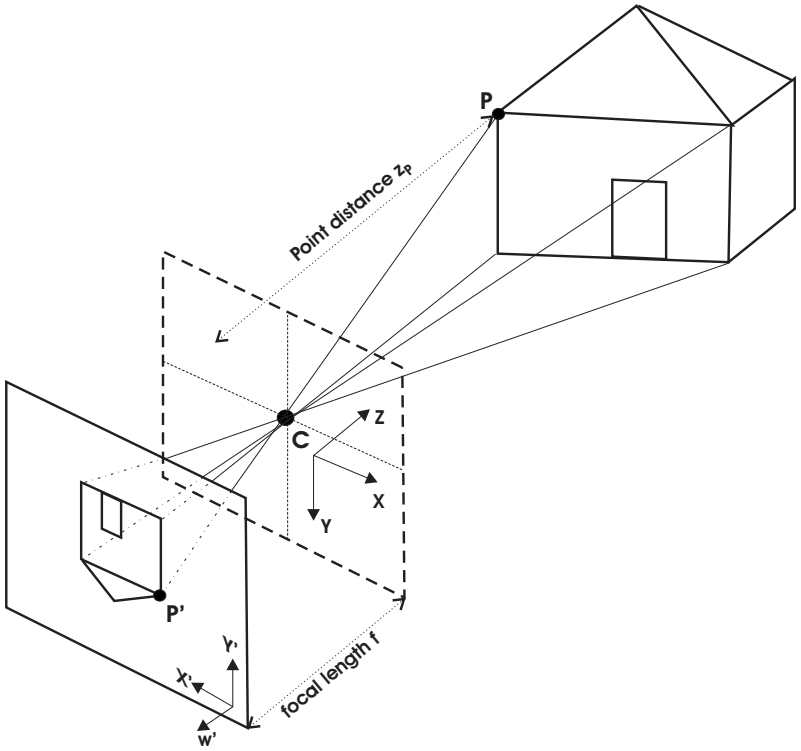


Figure 2.3: Optical mapping for the pinhole camera model.

2.1.2 The Camera Matrix

To explain the mapping of 3D world data on a 2D sensor that applies for a digital video camera, the pinhole camera model is very useful. Figure 2.3 shows the principle of the optical path for a pinhole camera. A pinhole camera is an idealized camera with aperture toward zero. In this model the energy that is necessary to activate the camera sensor is neglected. Under real conditions a pinhole camera would need an exposure time toward infinity. The physical model of the pinhole camera is only characterized by the focal length f, which is the distance of the pinhole - also known as the camera center - and the sensor plane. To describe relations between points $\mathbf{p}$ in space and their images $\mathbf{p}'$ on the sensor we refer to the 1D-projective space depicted in Figure 2.4. Here, without loss of generality, the camera plane is mirrored at the coordinates of the camera hole. For this case with x' as

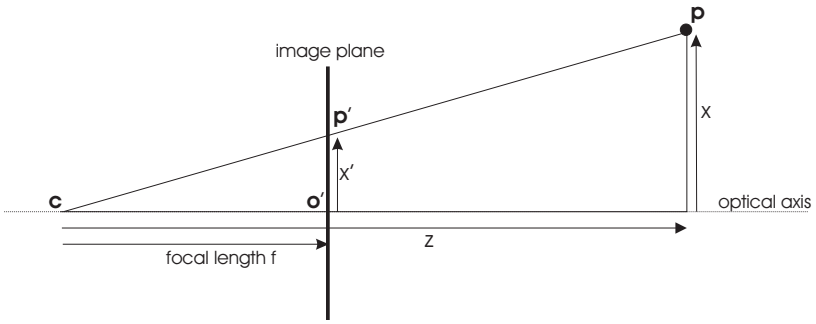


Figure 2.4: On the relation between world points $\mathbf{p}$ and camera sensor points $\mathbf{p}'$.

first entry of vector $\mathbf{p}'$ we obtain

$$x' = \frac{f}{z}x, \tag{2.9}$$

or in projective geometry

$$w\mathbf{p}' = z\begin{pmatrix} x' \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{pmatrix} x \\ z \\ 1 \end{pmatrix} = \mathbf{P}\mathbf{p}. \tag{2.10}$$

The matrix $\mathbf{P}$ is called the *camera projection matrix*. For 3D world points $\mathbf{P}$ is a 3×4 matrix. The above mentioned case is special since camera center $\mathbf{c}$ and the origin of the world coordinate system are the same. Also the camera direction, i.e., the direction of the optical axis, and line $(0, 0, z)^T$ lie upon each other. For the general case $\tilde{\mathbf{c}} = (x_c, y_c, z_c)^T$ (description in Euclidean space) is translated and the pointing direction of the camera is rotated by any rotation matrix $\mathbf{R}$. Thus, we obtain

$$\begin{aligned} w\mathbf{p}' = w\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} &= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{R}\begin{bmatrix} 1 & 0 & 0 & -x_c \\ 0 & 1 & 0 & -y_c \\ 0 & 0 & 1 & -z_c \end{bmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \\ &= \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\tilde{\mathbf{c}}\right]\mathbf{p} \\ &= \underbrace{\mathbf{K}\left[\mathbf{R} \mid -\mathbf{R}\tilde{\mathbf{c}}\right]}_{\mathbf{P}}\mathbf{p}. \end{aligned} \tag{2.11}$$

$\mathbf{K}$ is a 3×3 matrix containing the intrinsic camera parameters also called the *camera calibration matrix*. The right term $[\mathbf{R} \mid -\mathbf{R}\tilde{\mathbf{c}}]$ contains the external camera parameters depending only on the location of the camera in space. In Equation (2.11) the camera calibration matrix $\mathbf{K}$ has a very simple form (only focal length), which is sufficient for most camera setups. In the next section we will investigate the impact of camera specific parameters.

2.1.3 Sensor Properties and Lens Distortions

The operations in Equation (2.11) being applied on a point in $\mathbb{P}^3$ to be projected onto the camera plane can be interpreted by the following flowchart. The right block is represented

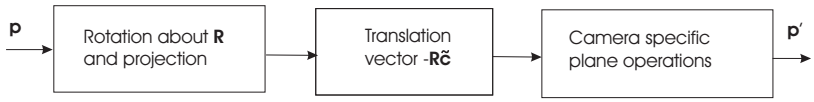


Figure 2.5: Consecutive operations on a point p for projection

by the matrix $\mathbf{K}$. Since $\mathbf{K}$ relates points in $\mathbb{P}^2$ to points in $\mathbb{P}^2$ it can just be regarded as sensor specific. In case the principle point of the sensor $\mathbf{o}'$ does not have the coordinates $(0, 0)^T \in \mathbb{R}^2$ the third column of $\mathbf{K}$, which signifies a 2D translation (Equation (2.5)), is

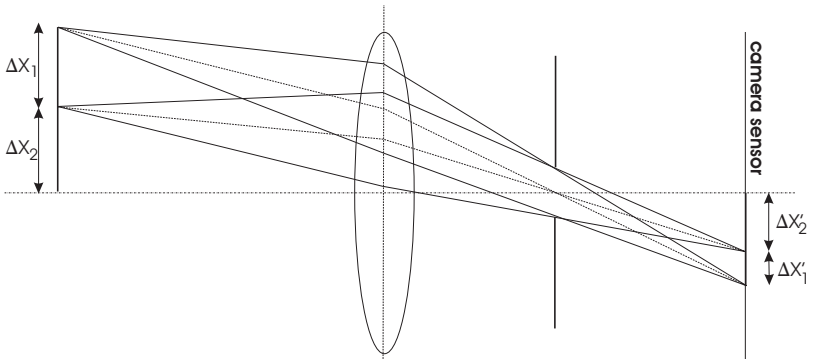


Figure 2.6: Lens distortions: Because of different distances from optical axis the deviating lens refraction properties make an accurate mapping of ratio $\frac{\Delta x_1}{\Delta x_2}$ impossible. Therefore, $\frac{\Delta x_1}{\Delta x_2} \neq \frac{\Delta x'_1}{\Delta x'_2}$.

$(x'_o, y'_o, 1)^T$. The sampling grid of the sensor itself is represented by a skewness factor τ and a sampling ratio $\eta = \Delta y'_s / \Delta x'_s$, where $\Delta y'_s$ is the vertical distance between two adjacent sensor elements in column and $\Delta x'_s$ the horizontal distance between two elements in a row. Finally $\mathbf{K}$ becomes

$$\mathbf{K} = \begin{bmatrix} f & \tau & x'_o \\ 0 & \eta f & y'_o \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.12}$$

For commonly utilized cameras τ equals zero and η equals one.

Lens distortion

For most cameras with finite aperture an optical lens or lens system bundles the rays of light. This has two advantages: First, the pinhole model can be approximated with very short exposure times, and second, the way of light can artificially be elongated, so that the focal length has a broad dynamic in a single apparatus. Unfortunately, the principles derived from the pinhole model do only apply for rays close to the optical axis. From the sensor point of view the distortions can be regarded as radial. The reason is, that especially lenses of low precision have deviating optical properties toward the edge. The observed effects are also known as barrel or pincushion effects. See Figure 2.6 for explanation. Since most lenses are circular, the distortions can be approximated by a correction term $L(\cdot)$, which is a *radial basis function* (RBF). Let $\tilde{\mathbf{x}}' \in \mathbb{R}^2$ be the coordinates of a distortion free image of any 3D point on the camera sensor. Then, the distorted image and thus the coordinates of point $\tilde{\mathbf{x}}'' \in \mathbb{R}^2$ can be derived by

$$\tilde{\mathbf{x}}'' = \tilde{\mathbf{x}}' + L(\tilde{\mathbf{x}}', \tilde{\mathbf{x}}'_o) \left(\tilde{\mathbf{x}}' - \tilde{\mathbf{x}}'_o \right), \tag{2.13}$$

with

$$L(\tilde{\mathbf{x}}', \tilde{\mathbf{x}}'_o) = \Phi \left((\tilde{\mathbf{x}}' - \tilde{\mathbf{x}}'_o)^T \mathbf{M} (\tilde{\mathbf{x}}' - \tilde{\mathbf{x}}'_o) \right), \tag{2.14}$$

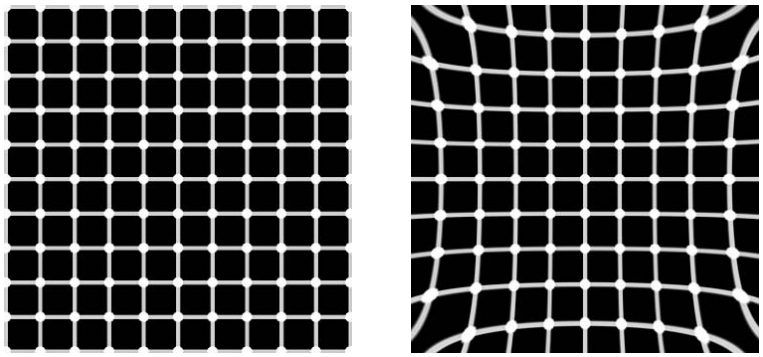


Figure 2.7: Example of lens distortion, e.g., pincushion effect; Left: original image (350x350 pixels); Right: distortion parameters $\kappa_1 = 1 \cdot 10^{-6}$ and $\kappa_2 = 5 \cdot 10^{-11}$

$\Phi(z)$ is a radial basis function. M describes the metric and can be set to Identity if we refer to pixel measure. Typical utilized functionalities are $\Phi_1(z) = \kappa_1 z$ or $\Phi_2(z) = \kappa_1 z + \kappa_2 z^2$. An example for lens distortion is shown in Figure 2.7. κ_1 and κ_2 belong to the internal camera parameter set in the next sections. Methods for their estimation are proposed in [114] and [164]. For high quality cameras, such as capturing devices for TV broadcast, this effect can be neglected. Only for extreme wide angle recordings one will have to deal with this type of distortion. In this thesis we will not consider this effect directly.

2.2 From 3D World to 2D Images

In this section we will figure out under which conditions 2D transformation-based image registration is the appropriate mapping between points in different images. The question that arises is: How do two cameras - or two different positions of one camera - have to relate to each other, that a definite transformation between points on both sensors exists, independent from the position of the original points in 3D space?

2.2.1 Camera Motions and Environment Models

In order to derive the conditions that apply for image transformations, we will look at the general case for two views. In Figure 2.8 the two views have different camera centers ($\mathbf{o}$ and $\mathbf{c}$) and the principal axis (optical axis) are rotated about rotation $\mathbf{R}$. Without loss of generality we can set the first view's camera center equal to the origin of the world coordinate system and the principal axis equal to the z-axis. Hence, for both views we obtain camera

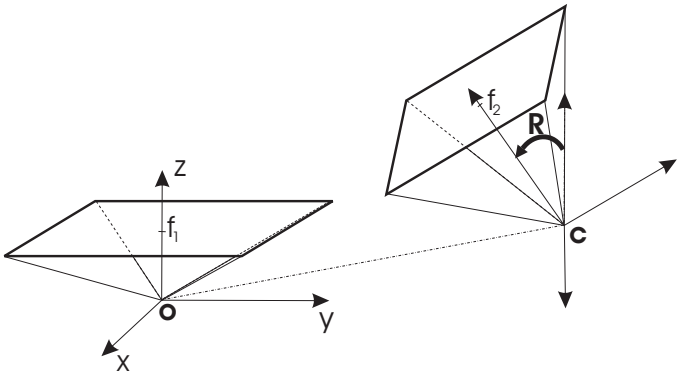


Figure 2.8: General case for two view geometry.

matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ applying Equations 2.11 and 2.12

$$\mathbf{P}_1 = \mathbf{K}_1\mathbf{I}\left[\mathbf{I} \mid -\tilde{\mathbf{0}}\right] = \begin{bmatrix} f_1 & 0 & x'_{o1} \\ 0 & f_1 & y'_{o1} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{I} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.15}$$

$$\mathbf{P}_2 = \mathbf{K}_2\mathbf{R}\left[\mathbf{I} \mid -\tilde{\mathbf{c}}\right] = \begin{bmatrix} f_2 & 0 & x'_{o2} \\ 0 & f_2 & y'_{02} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R} \begin{bmatrix} 1 & 0 & 0 & -x_c \\ 0 & 1 & 0 & -y_c \\ 0 & 0 & 1 & -z_c \end{bmatrix}. \tag{2.16}$$

Thus, we obtain the following mappings for real world points $\mathbf{p}$ onto the camera sensors:

$$w_1\mathbf{p}'_1 = w_1\left(x'_1, y'_1, 1\right)^T = \mathbf{K}_1\left[I \mid 0\right]\mathbf{p} = \mathbf{K}_1\tilde{\mathbf{p}} \tag{2.17}$$

$$w_2\mathbf{p}'_2 = w_2\left(x'_2, y'_2, 1\right)^T = \mathbf{K}_2\left[\mathbf{R} \mid -\mathbf{R}\tilde{\mathbf{c}}\right]\mathbf{p} = \mathbf{K}_2\mathbf{R}\tilde{\mathbf{p}} - \mathbf{K}_2\mathbf{R}\tilde{\mathbf{c}} \tag{2.18}$$

Here w_1 and w_2 are the specific scaling parameters for the normalized $\mathbb{P}^2$-points $\mathbf{p}'_1$ and $\mathbf{p}'_2$. Points $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{p}}$ are the Euclidean representations of the projective counterparts $\mathbf{c} \in \mathbb{P}^3$ and $\mathbf{p} \in \mathbb{P}^3$. In the following we will examine how points $\mathbf{p}'_1$ and $\mathbf{p}'_2$ can be related to each other. Note that $\mathbf{p}'_1$ and $\mathbf{p}'_2$ are the only measurable coordinates while taking pictures from both camera views.

The general case - parallax

Substituting $\tilde{\mathbf{p}}$ in Equations 2.18 by re-arranging 2.17 yieldss

$$\begin{aligned} w_2\mathbf{p}'_2 &= \mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}w_1\mathbf{p}'_1 - \mathbf{K}_2\mathbf{R}\tilde{\mathbf{c}} \\ w_2\left(x'_2, y'_2, 1\right)^T &= w_1 \cdot \tilde{\mathbf{a}} + \tilde{\mathbf{b}}, \end{aligned} \tag{2.19}$$

with

$$\begin{aligned} \tilde{\mathbf{a}} = \left(x_a, y_a, z_a\right)^T &= \mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}\mathbf{p}'_1 \text{ and} \\ \tilde{\mathbf{b}} = \left(x_b, y_b, z_b\right)^T &= -\mathbf{K}_2\mathbf{R}\tilde{\mathbf{c}} \end{aligned}$$

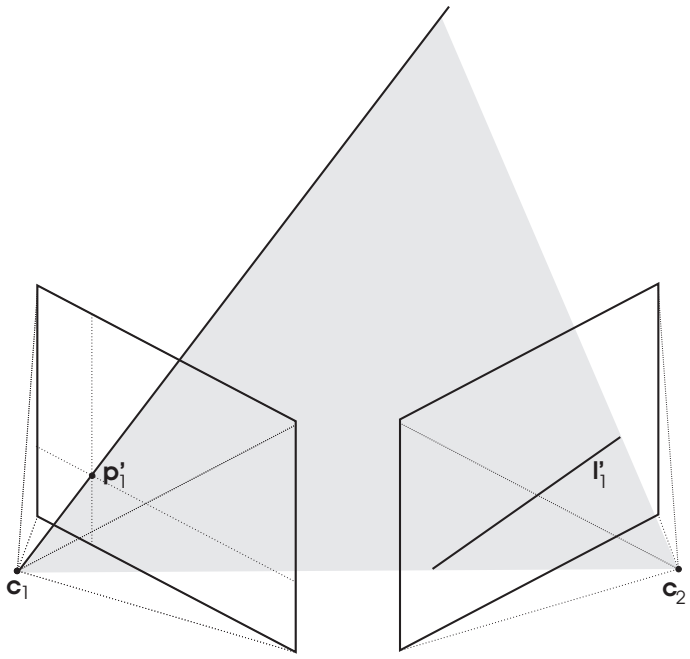


Figure 2.9: Point $\mathbf{p}'_1$ and corresponding epipolar line $\mathbf{l}'_1$.

The right side of Equation 2.19 describes a line in three-dimensional Euclidean space for any arbitrarily chosen point $\mathbf{p}'_1$ on the sensor of view one. Equation 2.19 can be rewritten as

$$w_2 \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix} = \begin{pmatrix} w_1 x_a + x_b \\ w_1 y_a + y_b \\ w_1 z_a + z_b \end{pmatrix}, \tag{2.20}$$

where $w_2 = w_1 z_a + z_b$. The right term of Equation 2.20 represents the implicit parametric form of a line. The explicit form can be given by

$$0 = (y_b z_a - y_a z_b)x'_2 + (x_a z_b - x_b z_a)y'_2 + (x_b y_a - x_a y_b). \tag{2.21}$$

The obtained line $\mathbf{l}'_1 = (y_b z_a - y_a z_b, x_a z_b - x_b z_a, x_b y_a - x_a y_b)^T$ represents all possible projections $\mathbf{p}'_2$ of the unknown original 3D point $\mathbf{p}$ on view 2 under the constraint that the projection on view 1 is exact the point $\mathbf{p}'_1$. $\mathbf{l}'_1$ is also called the *epipolar line* in the corresponding view. A full derivation of Equation 2.21 can be found in appendix A.1. A schematic depiction of the general case mappings of point to lines is shown in Figure 2.9.

Therefore, a unique mapping of points in two views does not exist in the general case. Moreover, each single point in one view can have an infinite number of corresponding points in the other view depending on the distance to the camera center $\mathbf{c}_1$. Hence, 2D image transformations do not properly apply for the arbitrary camera motion. The estimation of transformations for video material of any kind will be erroneous if no constraints are given to the recording setup.

Camera rotation and zoom

In case the two cameras have only different viewing directions, i.e., the principle axis intersect at the camera center, the whole problem becomes much easier. Note, that changing the focal length from f_1 to f_2 does not affect the pointing direction. Here $\tilde{\mathbf{c}} = 0$ the right term of Equation 2.19 equals zero and therefore $\tilde{\mathbf{b}} = 0$. Hence, Equation 2.20 becomes

$$w_2 \begin{pmatrix} x_2' \\ y_2' \\ 1 \end{pmatrix} = \begin{pmatrix} w_1 x_a \\ w_1 y_a \\ w_1 z_a \end{pmatrix}. \tag{2.22}$$

Thus, $x_2' = x_a/z_a$ and $y_2' = y_a/z_a$ are fixed values independent of the scaling parameter w_1. This fact can be interpreted as existence of one unique mapping that relates all points in one image to all points in the second image. The parallax effect of the general case does not occur since the absolute direction of the projection beam in the world coordinate system remains the same. Setting $w = w_1/w_2$ we can write

$$w\mathbf{p}_2' = \mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}\mathbf{p}_1' = \mathbf{H}\mathbf{p}_1'. \tag{2.23}$$

Matrix $\mathbf{H}$ is also known as *homography*. It is a 3×3 matrix describing the mapping between points $\mathbf{p}_1'$ in the first view and points $\mathbf{p}_2'$ in the second view. We can conclude that for cameras, applying rotation and zoom only, a 2D image transformation between two views can be estimated. The estimated mapping is the homography $\mathbf{H}$.

Up to now we have only considered the relations of the two camera views without restricting the space of points in the 3D world. In the next sections we will investigate under which conditions for the original positions a 2D image transformation will be appropriate.

The general case and points at infinity

For points $\mathbf{p}$ located far away, $\|\tilde{\mathbf{c}}\| << \|\tilde{\mathbf{p}}\|$, the scaling value w_1 of Equation 2.19 becomes $w_1 \to \infty$. Thus, we can disregard vector $\tilde{\mathbf{b}}$. The result is again a homography that applies as approximated 2D image transformation (see Equation 2.23). The effect can be observed while watching the fixed celestial bodies from a moving car. For the observer the stars seem to move with the vehicle and thus, the observed image does not change at all. This can be explained by the fact that independent from the point of view (place of the camera), the projection beams into the different camera centers will be parallel.

The general case and points on a plane

In case the original points are located on a plane, which is the hyperplane of $\mathbb{R}^3$, we reduce the projection source space by one dimension. Now the coordinates of 3D point $\mathbf{p} = (x, y, z, 1)^T$

are not independent any more. Moreover, for a plane we can write

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ \alpha x + \beta y + \gamma \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & \beta & \gamma \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.24}$$

Replacing $\mathbf{p}$ in Equation 2.11 we obtain

$$\begin{aligned} w\mathbf{p}' &= \mathbf{P}\mathbf{p} \\ &= \mathbf{K}\left[\mathbf{R} \mid -\mathbf{R}\tilde{\mathbf{c}}\right] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & \beta & \gamma \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \\ &= \mathbf{M} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \end{aligned} \tag{2.25}$$

Note that Equation 2.24 does not represent every possible plane in three-dimensional space, which can easily be achieved by replacing vector $(x, y, 1)^T$ by $(x, z, 1)^T$ or $(y, z, 1)^T$. Since camera matrix $\mathbf{P}$ is a 3×4 matrix $\mathbf{M}$ is in general a non-singular invertible 3×3 matrix. M describes the invertible projection from any plane onto the camera plane. Thus, for every camera view i we can find a matrix $\mathbf{M}_i$ representing the bijective mapping from source to sensor plane. Since the mapping can be inverted we can finally derive the mapping from one sensor to another by

$$\begin{aligned} w_2\mathbf{p}_2' &= \mathbf{M}_2\mathbf{M}_1^{-1}w_1\mathbf{p}_1' \\ &= \mathbf{H}w_1\mathbf{p}_1'. \end{aligned} \tag{2.26}$$

From Equation 2.26 we can conclude that the 2D image transformation between two arbitrarily chosen camera viewpoints is a homography $\mathbf{H}$ if the source space is restricted to a plane. Another explanation for the application of homographies as correct image transformation is given in Figure 2.10. In general we can describe this homography by

$$\mathbf{H} = \mathbf{K}_2\mathbf{R}\left(\mathbf{I} - \tilde{\mathbf{c}}\frac{\mathbf{n}^T}{d}\right)\mathbf{K}_1^{-1}, \tag{2.27}$$

where $\mathbf{R}$ is the rotation matrix between both sensors and $\tilde{\mathbf{c}}$ the center of the second camera in Euclidean coordinates. $\mathbf{n}$ and d are the normal vector of the source plane and the distance to the plane, respectively ($d > 0$). They satisfy the Euclidean plane equation

$$\mathbf{n}^T\tilde{\mathbf{p}} = d. \tag{2.28}$$

A full derivation of Equation 2.27 can be found in Appendix A.2.

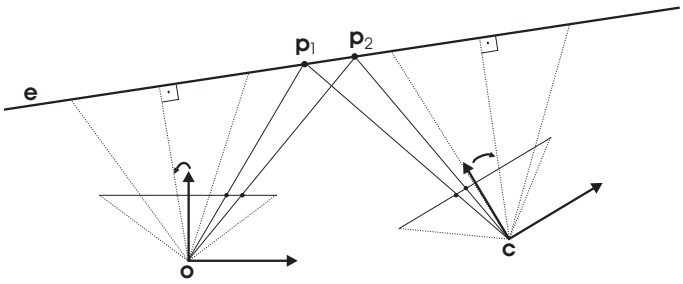


Figure 2.10: Projection of plane points $\mathbf{p}_1$ and $\mathbf{p}_2$ onto the camera sensor planes: If the camera centers **o** and **c** are not in the source plane **e** the cameras can be rotated and zoomed by a homography, so that the source plane is the sensor plane itself. Both cameras see then the same image, just differing in a translational offset. Since the translation occurs now in the sensor space, image homographies still apply for sensor-to-sensor transformation.

Summary

The cases where a unique image transformation for two camera views applies are:

1. Rotational camera motion, represented by rotation matrix $\mathbf{R}$
2. Camera zoom, represented by different intrinsic parameter matrices $\mathbf{K}_1$ and $\mathbf{K}_2$
3. Any combination of rotation and zoom
4. Arbitrary camera motion with source points located at infinity
5. Arbitrary camera motion with source points located at a plane in 3D space

The particular appropriate image transformation is always a homography also known as *perspective transformation* [133]. It is represented by a 3×3 matrix $\mathbf{H}$ and has, due to projective scaling invariance, basically 8 independent entries. Later we will show that the number of degrees of freedom is less than eight if we consider all physical constraints mentioned above.

2.2.2 Image Homographies: Properties and Constraints

The perspective 2D image transformation, represented by homography $\mathbf{H}$ with

$$\mathbf{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix},$$

has several important and useful properties. One main property is the invertibility of matrix $\mathbf{H}$. We now will investigate under which conditions $\mathbf{H}$ can be inverted. Starting from Equation 2.27, which is the most general case for any homography-based image mapping, we can derive the conditions for inversion. It is important to know that the rotation only case is covered by setting $\tilde{\mathbf{c}} = \mathbf{0}$. Since the internal camera parameter matrices $\mathbf{K}_1$ and

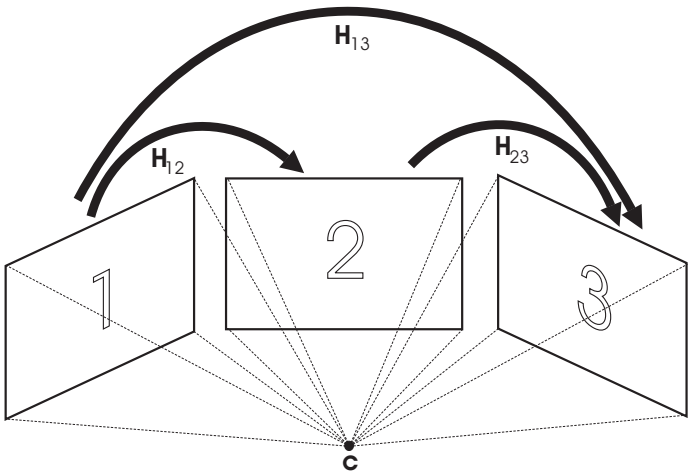


Figure 2.11: Concatenation of homographies: Because of scale invariance of matrices in projective space, for the overall homography $\mathbf{H}_{13}$ applies $\mathbf{H}_{13} = \mathbf{H}_{23} \cdot \mathbf{H}_{12}$

$\mathbf{K}_2$ are upper triangular matrices with non-zero entries at the diagonal they have full rank and thus can be inverted. The rotation matrix $\mathbf{R}$ is an orthogonal non-singular matrix with orthonormal row and column vectors. It can always be inverted. Hence, only term $\mathbf{N} = \left(\mathbf{I} - \tilde{\mathbf{c}}\frac{\mathbf{n}^T}{d}\right)$ has to be researched. For rotation only camera movement $\mathbf{N}$ becomes identity. In case of arbitrary motion a plane world object $\mathbf{N}$ is invertible if

$$\begin{aligned} \det(N) &= \det\left(\mathbf{I} - \tilde{\mathbf{c}}\frac{\mathbf{n}^T}{d}\right) \\ &= \left(1 - \tilde{\mathbf{c}}^T\frac{\mathbf{n}}{d}\right) \neq 0. \end{aligned} \tag{2.29}$$

Rearranging Equation 2.29 yields

$$\tilde{\mathbf{c}}^T\mathbf{n} \neq d, \tag{2.30}$$

which can be interpreted as the constraint that the center of camera two should not be a part of the object plane. In practice this constraint is always complied with.

A second important property is the possibility of homography concatenation. Because the multiplication of any 2 homographies $\mathbf{H}_1$ and $\mathbf{H}_2$ yields a homography again - all homographies build a *projective linear group* $PL(3)$, noted as $(H, \cdot)$ - we can write

$$\mathbf{H}_{12} = \mathbf{H}_1 \cdot \mathbf{H}_2. \tag{2.31}$$

Thus, an image transformation $\mathbf{H}_{1,N}$ from frame 1 to frame N within a video sequence can be defined in a recursive manner

$$\mathbf{H}_{1,N} = \mathbf{H}_{N-1,N} \cdot \mathbf{H}_{1,N-1} \;\; with \;\; N \in \mathbb{N}, \tag{2.32}$$

where $\mathbf{H}_{1,N-1}$ is the transformation from frame 1 to $N-1$ and $\mathbf{H}_{N-1,N}$ the transformation from frame $N-1$ to N. See Figure 2.11 for an illustration of the concatenation property.

Taking invertibility and the concatenation property into account we are enabled to relate all views of a scene shot captured under homography conditions by combination and

inversion of the short-term transformations, i.e., the homographies between temporal adjacent frames. The short-term transformation or motion is sometimes referred to as the global motion (GM), in contrast to local block motion. This fact is especially important for the mosaicing technique, since we want to relate all frames F_i of a shot into the coordinate system of one reference frame F_r. We remarkably reduce the complexity of the registration being only dependent on the short-term estimation. For real data we additionally have to adjust the obtained concatenated transformation such that the final estimation will overcome the accumulation of short-term errors. This will be of major interest in chapter 3.

2.3 2D Image Transformations and Properties

In the next three sections we will discuss the properties of the derived image transformations, i.e., the homography and its approximations.

2.3.1 The Perspective Transformation

From the last sections we have derived the homography as the appropriate image transformation. Reinterpreting the matrix notation of $w\mathbf{p}'_2 = \mathbf{H}\mathbf{p}'_1$ yields

$$\begin{aligned} wx'_2 &= h_{00}x'_1 + h_{01}y'_1 + h_{02} \\ wy'_2 &= h_{10}x'_1 + h_{11}y'_1 + h_{12} \\ w &= h_{22}\left(\frac{h_{20}}{h_{22}}x'_1 + \frac{h_{21}}{h_{22}}y'_1 + 1\right), \end{aligned}$$

from what follows the *perspective transformation*

$$x'_2 = \frac{a_0 + a_1x'_1 + a_2y'_1}{1 + c_1x'_1 + c_2y'_1} \tag{2.33}$$

$$y'_2 = \frac{b_0 + a_1x'_1 + b_2y'_1}{1 + c_1x'_1 + c_2y'_1} \tag{2.34}$$

$$\begin{aligned} \text{with} \quad a_0 &= \frac{h_{02}}{h_{22}},\ a_1 = \frac{h_{00}}{h_{22}},\ a_2 = \frac{h_{01}}{h_{22}}, \\ b_0 &= \frac{h_{12}}{h_{22}},\ b_1 = \frac{h_{10}}{h_{22}},\ b_2 = \frac{h_{11}}{h_{22}}, \\ c_1 &= \frac{h_{20}}{h_{22}},\ c_2 = \frac{h_{21}}{h_{22}}. \end{aligned}$$

The perspective transformation has up to eight independent parameters, described by parameter vector $\mathbf{k}_{pers} = (a_0, \ldots, c_2)^T$ which can be fully determined by 4 pairs of image points, where each three of them are not collinear. A special subgroup of the perspective image transformations is the group of affine transformations.

Figure 2.12: "Dolly zoom" in the movie "Goodfellas": The characters in the foreground remain unchanged while the background objects virtually seem to come closer. (Courtesy: Warner Bros.)

2.3.2 Affine Image Transformations

In case the object of an observation is far away from the camera center and the object is captured using large zoom, i.e., large focal depth f, we approximately obtain a parallel projection into the camera sensor. The focal depth of the camera then tends toward infinity and the projection beams become parallel. Cameras with this property are known as *affine cameras*. Mathematically they can be described by modeling the camera center $\tilde{\mathbf{c}}$ at infinity ($\mathbf{c} = (x_c, y_c, z_c, 0)^T$). The transition from projective to affine camera is a well used effect in cinematography, also known as "Dolly zoom" or "Vertigo zoom". In order to achieve this effect the camera is moved away from the object while the lens zoom is adjusted in such way that the projected main objects remains at same size and same position. See Figure 2.12 for an example in the movie "Goodfellas".

Following [42] the affine camera matrix $\mathbf{P}_\infty$ can be written as

$$\mathbf{P}_\infty = \mathbf{K} \begin{bmatrix} \mathbf{r}_0^T & -\mathbf{r}_0^T\tilde{\mathbf{c}} \\ \mathbf{r}_1^T & -\mathbf{r}_1^T\tilde{\mathbf{c}} \\ \mathbf{0}^T & d_0 \end{bmatrix}. \tag{2.35}$$

$\mathbf{r}_0^T$ and $\mathbf{r}_1^T$ are the first two rows of rotation matrix R and d_0 is the distance of the world origin to the camera center $\mathbf{c}$ along the principal ray. Taking scale invariance into account $\mathbf{P}_\infty$ can be rewritten as [31]

$$\begin{aligned} \mathbf{P}_\infty &= \begin{bmatrix} d_0^{-1} & & \\ & d_0^{-1} & \\ & & 1 \end{bmatrix} \mathbf{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{O}} \begin{bmatrix} \mathbf{R} & \tilde{\mathbf{t}} \\ \mathbf{0}^T & 1 \end{bmatrix} \\ &= \hat{\mathbf{K}}\mathbf{O} \begin{bmatrix} \mathbf{R} & \tilde{\mathbf{t}} \\ \mathbf{0}^T & 1 \end{bmatrix}. \end{aligned} \tag{2.36}$$

Matrix $\mathbf{O}$ can be regarded as an orthogonal (parallel) mapping onto the normalized camera sensor plane $z = 1$. Its left hand 3×3 sub-matrix is now singular. Considering the sensor-to-sensor mapping for those affine cameras as done for the perspective ones in section 2.2.1

while the 3D object is plane, we obtain

$$\begin{aligned} w_1\mathbf{p}'_1 &= \mathbf{P}_{\infty,1}\cdot\mathbf{p} \\ &= \hat{\mathbf{K}}_1\mathbf{O}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & \beta & \gamma \\ 0 & 0 & 1 \end{bmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \hat{\mathbf{K}}_1\mathbf{I}_{3\times 3}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \end{aligned} \tag{2.37}$$

and

$$\begin{aligned} w_2\mathbf{p}'_2 &= \mathbf{P}_{\infty,2}\cdot\mathbf{p} \\ &= \hat{\mathbf{K}}_2\mathbf{O}\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha & \beta & \gamma \\ 0 & 0 & 1 \end{bmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \\ &= \hat{\mathbf{K}}_2\underbrace{\begin{bmatrix} r_{00}+\alpha r_{02} & r_{01}+\beta r_{02} & \gamma r_{02}+x_t \\ r_{10}+\alpha r_{12} & r_{11}+\beta r_{12} & \gamma r_{12}+y_t \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \end{aligned} \tag{2.38}$$

In Equation 2.38 vector $\mathbf{t}$ describes a translation with $\mathbf{t} = -\mathbf{R}\tilde{\mathbf{c}}$. Combining Equations 2.37 and 2.38 yields in

$$\begin{aligned} w_2\mathbf{p}'_2 &= \hat{\mathbf{K}}_2\mathbf{A}\hat{\mathbf{K}}_1^{-1}w_1\mathbf{p}'_1 \\ &= \hat{\mathbf{A}}w_1\mathbf{p}'_1 \\ &= \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix}w_1\mathbf{p}'_1. \end{aligned} \tag{2.39}$$

Matrix $\hat{\mathbf{A}} = \hat{\mathbf{K}}_2\mathbf{A}\hat{\mathbf{K}}_1^{-1}$ describes a special case of the perspective transformation (Equations 2.33 and 2.34) with $c_1 = c_2 = 0$. It can also be written in Euclidean coordinates

$$\tilde{\mathbf{p}}'_2 = \hat{\mathbf{A}}_{2\times 2}\tilde{\mathbf{p}}'_2 + \begin{pmatrix} a_0 \\ b_0 \end{pmatrix}, \tag{2.40}$$

where it still remains a linear mapping. $\hat{\mathbf{A}}_{2\times 2}$ is the upper left 2×2 sub-matrix of homography $\hat{\mathbf{A}}$. This special case is commonly known as *affine image transformation*. In general $\hat{\mathbf{A}}_{2\times 2}$ is non-singular and thus, can be decomposed using *singular value decomposition* (SVD):

$$\hat{\mathbf{A}}_{2\times 2} = \mathbf{U}\mathbf{D}\mathbf{V}^T = \underbrace{\mathbf{U}\mathbf{V}^T}_{\mathbf{R}(\Theta)}\ \underbrace{\mathbf{V}}_{\mathbf{R}(-\Phi)}\ \mathbf{D}\ \underbrace{\mathbf{V}^T}_{\mathbf{R}(\Phi)}. \tag{2.41}$$

Following Equation 2.41 an affine 2D transformation complies a rotation $\mathbf{R}(\Phi)$, a scaling along the new basis, a rotation $-\mathbf{R}(\Phi)$ back on the old basis and finally a rotation $\mathbf{R}(\Theta)$

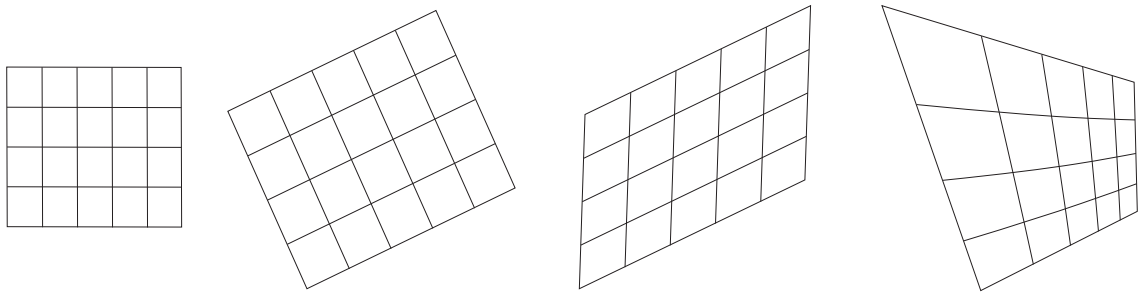


Figure 2.13: Hierarchy of transformations based on the perspective image mapping: identity, similarity, affine, and perspective transformation (from left to right).

about angle θ. Since rotation and scaling are only in-plane operations it can be concluded that the affine transformation preserves parallelism, which is opposed to the properties of the general perspective transformation. Further, if we constrain diagonal matrix $\mathbf{D}$ to be $\mathbf{D} = d\mathbf{I}$, i.e., the singular values λ_1 and λ_2 equal d, we obtain a special case of the affine transformation known as *similarity transformation*. It additionally has the property of angle preservation. The hierarchy of special cases of the perspective transformation is depicted in Figure 2.13.

Affine Transformation as Approximation

The affine image transformation can also be regarded as an approximation of the real-world-driven perspective transformation using *Taylor series expansion*. For further examination we assume that the capture conditions comply with the constraints for homography mappings (see Section 2.2.1). Let $\mathbf{x} = (x, y)^T \in \mathbb{R}^2$ be a point in one frame of the video shot. For the ease of reading we disregard the tilde ($\tilde{\cdot}$) for points in Euclidean space as introduced in Equation 2.2. The corresponding point of $\mathbf{x}$ in a different frame is $\mathbf{x}' = (x', y')^T \in \mathbb{R}^2$. Thus, $\mathbf{x}'(\mathbf{x}) = \mathcal{T}_{pers}\left(\mathbf{k}_{pers}, \mathbf{x}\right)$ with $\mathcal{T}_{pers}(\cdot)$ meaning the perspective transformation, which is a mapping dependent on transformation vector $\mathbf{k}_{pers}$ and the coordinates of point $\mathbf{x}$. Using Taylor series expansion about a fixed point $\mathbf{x}_0$ we can write

$$\mathbf{x}'(\mathbf{x}) = \mathbf{x}'(\mathbf{x}_0) + \nabla(\mathbf{x}')|_{\mathbf{x}=\mathbf{x}_0}\left(\mathbf{x} - \mathbf{x}_0\right) + \mathbf{R}_1, \tag{2.42}$$

where $\nabla(\mathbf{x}')$ is the Jacobian of $\mathbf{x}'(\mathbf{x})$ with

$$\nabla(\mathbf{x}') = \begin{bmatrix} \frac{\partial \mathbf{x}'}{\partial \mathbf{x}} & \frac{\partial \mathbf{x}'}{\partial \mathbf{y}} \\ \frac{\partial \mathbf{y}'}{\partial \mathbf{x}} & \frac{\partial \mathbf{y}'}{\partial \mathbf{y}} \end{bmatrix}$$

and $\mathbf{R}_1$ being the remainder when terminating the expansion after terms of first order we obtain

$$\begin{aligned} x' &= \frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\left(1+\frac{c_1x_0}{\Gamma(\mathbf{x}_0)}+\frac{c_2y_0}{\Gamma(\mathbf{x}_0)}\right)-\frac{a_1x_0+a_2y_0}{\Gamma(\mathbf{x}_0)} \\ &\quad +\left(\frac{a_1}{\Gamma(\mathbf{x}_0)}-c_1\frac{A(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)}\right)x+\left(\frac{a_2}{\Gamma(\mathbf{x}_0)}-c_2\frac{A(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)}\right)y+R_{1,x} \qquad (2.43) \\ y' &= \frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\left(1+\frac{c_1x_0}{\Gamma(\mathbf{x}_0)}+\frac{c_2y_0}{\Gamma(\mathbf{x}_0)}\right)-\frac{b_1x_0+b_2y_0}{\Gamma(\mathbf{x}_0)} \\ &\quad +\left(\frac{b_1}{\Gamma(\mathbf{x}_0)}-c_1\frac{B(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)}\right)x+\left(\frac{b_2}{\Gamma(\mathbf{x}_0)}-c_2\frac{B(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)}\right)y+R_{2,x}. \qquad (2.44) \end{aligned}$$

Here the terms $A(\mathbf{x})$, $B(\mathbf{x})$, and $\Gamma(\mathbf{x})$ signify the numerators and denominators of the perspective transformation

$$\begin{aligned} x' &= \frac{A(\mathbf{x})}{\Gamma(\mathbf{x})}=\frac{a_0+a_1x+a_2y}{1+c_1x+c_2y} \\ y' &= \frac{B(\mathbf{x})}{\Gamma(\mathbf{x})}=\frac{b_0+b_1x+b_2y}{1+c_1x+c_2y}. \end{aligned}$$

Coefficient comparison of Equations 2.43 and 2.44 with Equation 2.39 and negligence of the remainder leads to the affine transformation:

$$\begin{aligned} x'_{aff} &= a_{0,aff}+a_{1,aff}x+a_{2,aff}y \qquad (2.45) \\ y'_{aff} &= b_{0,aff}+b_{1,aff}x+b_{2,aff}y \qquad (2.46) \end{aligned}$$

with

$$\begin{aligned} a_{0,aff} &= \frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\left(1+\frac{c_1x_0}{\Gamma(\mathbf{x}_0)}+\frac{c_2y_0}{\Gamma(\mathbf{x}_0)}\right)-\frac{a_1x_0+a_2y_0}{\Gamma(\mathbf{x}_0)} \\ a_{1,aff} &= \frac{a_1}{\Gamma(\mathbf{x}_0)}-c_1\frac{A(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)} \qquad a_{2,aff}=\frac{a_2}{\Gamma(\mathbf{x}_0)}-c_2\frac{A(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)} \\ b_{0,aff} &= \frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\left(1+\frac{c_1x_0}{\Gamma(\mathbf{x}_0)}+\frac{c_2y_0}{\Gamma(\mathbf{x}_0)}\right)-\frac{b_1x_0+b_2y_0}{\Gamma(\mathbf{x}_0)} \\ b_{1,aff} &= \frac{b_1}{\Gamma(\mathbf{x}_0)}-c_1\frac{B(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)} \qquad b_{2,aff}=\frac{b_2}{\Gamma(\mathbf{x}_0)}-c_2\frac{B(\mathbf{x}_0)}{\Gamma^2(\mathbf{x}_0)} \end{aligned}$$

2.3.3 The Parabolic Transformation

Extending the Taylor series expansion to second order terms utilizing tensor notation we write

$$\mathbf{x}'(\mathbf{x})^a=\mathbf{x}'(\mathbf{x}_0)^a+\nabla(\mathbf{x}')^a_b|_{\mathbf{x}=\mathbf{x}_0}\left(\mathbf{x}^b-\mathbf{x}_0^b\right)+\frac{1}{2}\left(\mathbf{x}^c-\mathbf{x}_0^c\right)\mathcal{H}^a_{bc}|_{\mathbf{x}=\mathbf{x}_0}(\mathbf{x}^b-\mathbf{x}_0^b)+\mathbf{R}^a_2, \qquad (2.47)$$

where the superscripts and subscripts a, b, and c signify the single entries of the tensors with $a,b,c\in\{1,2\}$. $\mathcal{H}$ represents the 3rd rank *Hessian tensor* of $\mathbf{x}'(\mathbf{x})$ with

$$\begin{aligned} \mathcal{H}^a_{b1} &= \begin{bmatrix} \frac{\partial^2x'}{\partial x^2} & \frac{\partial^2x'}{\partial x\partial y} \\ \frac{\partial^2x'}{\partial x\partial y} & \frac{\partial^2x'}{\partial y^2} \end{bmatrix}, \\ \mathcal{H}^a_{b2} &= \begin{bmatrix} \frac{\partial^2y'}{\partial x^2} & \frac{\partial^2y'}{\partial x\partial y} \\ \frac{\partial^2y'}{\partial x\partial y} & \frac{\partial^2y'}{\partial y^2} \end{bmatrix}. \end{aligned}$$

Thus, we obtain a parabolic dependency for $\mathbf{x}'(x)$ and the coordinate transformation yields

$$x'_{par} = x'_{aff} + \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^T \left[\begin{array}{cc} \frac{\partial^2 x'}{\partial x^2} & \frac{\partial^2 x'}{\partial x \partial y} \\ \frac{\partial^2 x'}{\partial x \partial y} & \frac{\partial^2 x'}{\partial y^2} \end{array} \right]_{\mathbf{x}=\mathbf{x}_0} (\mathbf{x}-\mathbf{x}_0) \tag{2.48}$$

$$\begin{aligned} &= \frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\left(1+\frac{c_1 x_0}{\Gamma(\mathbf{x}_0)}+\frac{c_2 y_0}{\Gamma(\mathbf{x}_0)}\right) - \frac{a_1 x_0 + a_2 y_0}{\Gamma(\mathbf{x}_0)} \\ &\quad + \frac{1}{\Gamma(\mathbf{x}_0)^2}\left(\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}(c_1^2 x_0^2 + c_1 c_2 x_0 y_0 + c_2^2 y_2^2)\right. \\ &\qquad \left. -(a_1 c_1 x_0^2 + \frac{1}{2}(a_1 c_2 + a_2 c_1) x_0 y_0 + a_2 c_2 y_0^2)\right) \\ &\quad + \left(\frac{1}{\Gamma(\mathbf{x}_0)}\left(a_1 - c_1\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\right) - \frac{1}{\Gamma(\mathbf{x}_0)^2}\left(\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}(2c_1^2 x_0 + 2c_1 c_2 y_0)\right.\right. \\ &\qquad \left.\left. -2a_1 c_1 x_0 - a_1 c_2 y_0 - a_2 c_1 y_0\right)\right) x \\ &\quad + \left(\frac{1}{\Gamma(\mathbf{x}_0)}\left(a_2 - c_2\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\right) - \frac{1}{\Gamma(\mathbf{x}_0)^2}\left(\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}(2c_2^2 y_0 + 2c_1 c_2 x_0)\right.\right. \\ &\qquad \left.\left. -2a_2 c_2 y_0 - a_1 c_2 x_0 - a_2 c_1 x_0\right)\right) y \\ &\quad + \left(\frac{1}{\Gamma(\mathbf{x}_0)^2}\left(c_1^2\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)} - a_1 c_1\right)\right) x^2 \\ &\quad + \left(\frac{1}{\Gamma(\mathbf{x}_0)^2}\left(c_2^2\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)} - a_2 c_2\right)\right) y^2 \\ &\quad + \left(\frac{1}{\Gamma(\mathbf{x}_0)^2}\left(c_1 c_2\frac{A(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)} - \frac{1}{2}(a_1 c_2 + a_2 c_1)\right)\right) xy \end{aligned} \tag{2.49}$$

$$= a_{0,par} + a_{1,par}x + a_{2,par}y + a_{3,par}x^2 + a_{4,par}y^2 + a_{5,par}xy \tag{2.50}$$

and

$$y'_{par} = y'_{aff} + \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^T \left[\begin{array}{cc} \frac{\partial^2 y'}{\partial x^2} & \frac{\partial^2 y'}{\partial x \partial y} \\ \frac{\partial^2 y'}{\partial x \partial y} & \frac{\partial^2 y'}{\partial y^2} \end{array} \right]_{\mathbf{x}=\mathbf{x}_0} (\mathbf{x}-\mathbf{x}_0) \tag{2.51}$$

$$\begin{aligned} &= \frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\left(1+\frac{c_1 x_0}{\Gamma(\mathbf{x}_0)}+\frac{c_2 y_0}{\Gamma(\mathbf{x}_0)}\right) - \frac{b_1 x_0 + b_2 y_0}{\Gamma(\mathbf{x}_0)} \\ &\quad + \frac{1}{\Gamma(\mathbf{x}_0)^2}\left(\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}(c_1^2 x_0^2 + c_1 c_2 x_0 y_0 + c_2^2 y_2^2)\right. \\ &\qquad \left. -(b_1 c_1 x_0^2 + \frac{1}{2}(b_1 c_2 + b_2 c_1) x_0 y_0 + b_2 c_2 y_0^2)\right) \\ &\quad + \left(\frac{1}{\Gamma(\mathbf{x}_0)}\left(b_1 - c_1\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\right) - \frac{1}{\Gamma(\mathbf{x}_0)^2}\left(\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}(2c_1^2 x_0 + 2c_1 c_2 y_0)\right.\right. \\ &\qquad \left.\left. -2b_1 c_1 x_0 - b_1 c_2 y_0 - b_2 c_1 y_0\right)\right) x \end{aligned}$$

$$
\begin{aligned}
&+\left(\frac{1}{\Gamma(\mathbf{x}_0)}\left(b_2 - c_2\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}\right) - \frac{1}{\Gamma(\mathbf{x}_0)^2}\left(\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)}(2c_2^2 y_0 + 2c_1c_2x_0)\right.\right.\\
&\quad\left.\left. -2b_2c_2y_0 - b_1c_2x_0 - b_2c_1x_0\right)\right)y\\
&+\left(\frac{1}{\Gamma(\mathbf{x}_0)^2}\left(c_1^2\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)} - b_1c_1\right)\right)x^2\\
&+\left(\frac{1}{\Gamma(\mathbf{x}_0)^2}\left(c_2^2\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)} - b_2c_2\right)\right)y^2\\
&+\left(\frac{1}{\Gamma(\mathbf{x}_0)^2}\left(c_1c_2\frac{B(\mathbf{x}_0)}{\Gamma(\mathbf{x}_0)} - \frac{1}{2}(b_1c_2 + b_2c_1)\right)\right)xy \qquad (2.52)\\
=\;& b_{0,par} + b_{1,par}x + b_{2,par}y + b_{3,par}x^2 + b_{4,par}y^2 + b_{5,par}xy. \qquad (2.53)
\end{aligned}
$$

The so called *parabolic transformation* is a parametrized 2D image mapping based on 12 parameters stored in transformation parameter vector $\mathbf{k}_{par} = (a_0, ..., b_5)^T$. A full derivation of the parameters based on Taylor series expansion can be found in A.3.

Rewriting the parabolic transformation using projective coordinates while omitting the subscript *par* yields

$$
0 = \begin{pmatrix} x & y & 1 \end{pmatrix}
\begin{bmatrix} a_3 & \frac{a_5}{2} & \frac{a_1}{2} \\ \frac{a_5}{2} & a_4 & \frac{a_2}{2} \\ \frac{a_1}{2} & \frac{a_2}{2} & a_0 - x' \end{bmatrix}
\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad (2.54)
$$

$$
0 = \begin{pmatrix} x & y & 1 \end{pmatrix}
\begin{bmatrix} b_3 & \frac{b_5}{2} & \frac{b_1}{2} \\ \frac{b_5}{2} & b_4 & \frac{b_2}{2} \\ \frac{b_1}{2} & \frac{b_2}{2} & b_0 - y' \end{bmatrix}
\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad (2.55)
$$

The right hand side of Equations 2.54 and 2.55 generally describes a conic section, i.e., an intersection of a conic with a plane. Conic sections can be points, lines, hyperbolas, parabolas, and ellipses. Thus, since the parabola is only a special case it would be more convenient to call this transformation *quadratic* or *2D quadratic* transformation as proposed in [51]. From these equations we can easily derive that for every coordinate in the transformed image $I'(x', y')$ the set of source points (x, y) describe a conic section. In other words, every point (x', y') has as origin a set of points generated by the intersection of the two conic sections specified by Equations 2.54 and 2.55. Two conic sections on a plane intersect in one or two points. The case of non-intersecting conic sections can practically be neglected. If both curves have one point in common, the transformation is unique in both directions and can be inverted. Generally, this is not the case. The consequence is that the mapping is not unique into both directions, i.e., it is not bijective. This fact significantly increases the computational complexity of algorithms handling with the parabolic transformation.

2.3.4 Properties of the Parabolic Transformation

Because of its 12 parameters the parabolic image transformation has maximally 12 degrees of freedom. Though it is an approximation of the perspective mapping it can handle derivations

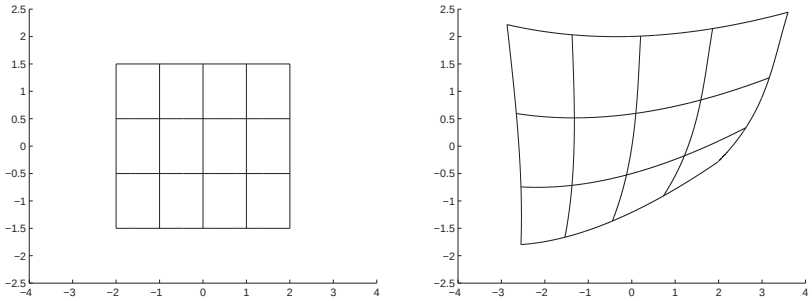


Figure 2.14: Example for parabolic transformation of a grid image (left) according to Equation 2.57 (right) using the following parameters: $\Theta = 15°$, $\Phi = 30°$, $\lambda_{11} = 1.5$, $\lambda_{12} = 0.1$, $\lambda_{21} = 1.0$, $\lambda_{22} = 0.2$, and $(a_0, b_0)^T = (0, 0)^T$.

from the assumed physical model more precisely due to four additional degrees of freedom. In the next chapter we will show that for real world recordings the parabolic model is more exact in terms of globally motion-compensated background reconstruction quality. Possible derivations from the assumed physical model are caused by small lens distortions and the fact that the optical camera center is not the same as the rotation center for cameras mounted on a tripod.

The parabolic transformation can be regarded it as extension of the affine model, which was decomposed according to Equation 2.41. The affine transformation is formed of a linear scaling along the basis vectors of a rotation matrix $\mathbf{R}(\Phi)$, a rotation only represented by matrix $\mathbf{R}(\Theta)$, and a translation:

$$\mathbf{x}_{aff} = \mathbf{R}(\Theta)\bar{\mathbf{x}} + \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} \qquad \text{with} \qquad \bar{\mathbf{x}} = \mathbf{R}(-\Phi)\mathbf{D}\mathbf{R}(\Phi)\mathbf{x}$$

If we change the linear scaling described by a multiplication with diagonal matrix $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2)$ into a nonlinear quadratic scaling operator defined by

$$D_{par}(\mathbf{x}) = D_{par}((x, y)^T) = \begin{pmatrix} \lambda_{11}x + \lambda_{12}x^2 \\ \lambda_{21}y + \lambda_{22}y^2 \end{pmatrix} \tag{2.56}$$

the parabolic form yields

$$\mathbf{x}_{par} = \mathbf{R}(\Theta)\mathbf{R}(-\Phi)D_{par}\left(\mathbf{R}(\Phi)\mathbf{x}\right) + \begin{pmatrix} a_0 \\ b_0 \end{pmatrix}. \tag{2.57}$$

Thus a main advantage of the parabolic transformation is that it allows dilation and contraction of the original image content along the same direction. Figure 2.14 shows an example for image transformation applying Equation 2.57.

In order to determinate the set of all 12 parabolic parameters for a transformation between two images I_a and I_b six pairs of corresponding points are necessary. Because of its nonlinearity, the direction the transformation is of high importance. Thus, it is impossible to directly recover the transformation vector from image I_b to I_a if only the transformation from I_a to I_b is known.

Inversion of the Parabolic Transformation

Because of not being unique into both directions the inversion of the parabolic transformation is algebraically very complex. Locally there exist more than one solution for $\mathcal{T}_{par}^{-1}$. In fact, we will see that the origin of any transformed pixel coordinate utilizing the parabolic mapping can only be estimated numerically.

Concatenation of Parabolic Transformations

Due to nonlinearity the parabolic transformation does not form a group, i.e., the composition of two mappings, $\mathcal{T}_{par,1} \otimes \mathcal{T}_{par,2}$ is generally no parabolic mapping. Moreover, the order of the overall transformation doubles with every new composition. Therefore, the concatenation of two parabolic transformations $\mathcal{T}_{par,12}$ and $\mathcal{T}_{par,23}$ can only be approximated by a parabolic transformation $\mathcal{T}_{par,13}$. Estimating this parabolic parameter set remains complex and can be achieved only by methods of high computational costs.

2.4 Limitations of 2D Mappings

There exist two classes of limitations for the useful registration of camera-captured images utilizing only 2D image mapping functionalities $\mathcal{T}^{2D} : \mathbb{R}^2 \mapsto \mathbb{R}^2$. The fist class, the *scenario-based limit*, is determined by the validity of the mappings and their underlying capturing and scene models. In other words, every derivation of the real world setting from the underlying model causes a deterministic registration error, which from a certain point on will degrade the registration tremendously. Since all the previously discussed models are simplified abstractions of reality one always has to deal with these kinds of derivations. In our case this happens when the types of camera movements are not covered by the model with respect to the relation between capturing device and the captured objects. That is, if the camera translation cannot be neglected in relation to the object's distance from the camera sensor plane, e.g., if not applies: $\|\tilde{\mathbf{c}}\| << z_{1,obj}$. Here $\tilde{\mathbf{c}}$ is the translational vector between two views of a scene shot and $z_{1,obj}$ is the depth of any object with respect to the coordinate system of the first camera view.

The second class of limitations, *the projection-based limit*, is determined by the single 2D transformation itself and its singularities, inconsistencies, and ambiguities. In case of the perspective transformation for panning and tilting camera specified by homography $\mathbf{H}_{12} = \mathbf{K}_2\mathbf{R}_{12}\mathbf{K}_1^{-1}$ this happens if the result of the transformation of any point $\mathbf{p}_1 \in \mathbb{P}^2$ yields an image point $\mathbf{p}_2 \in \mathbb{P}^2$ at infinity (singularity), i.e., $\mathbf{p}_2 = (x, y, w)^T = (x, y, 0)^T$ or if a change of sign in the scaling factor w (inconsistency) occurs. Regarding the transformation

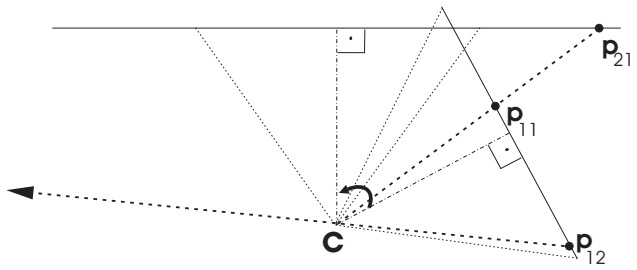


Figure 2.15: Limitations of the perspective transformation: While point $\mathbf{p}_{11}$ can be mapped correctly onto the image plane of the second plane, point $\mathbf{p}_{12}$ will be backward projected through the projection center. Thus, the transformed first view will topologically become inconsistent.

as projection of points on a new basis, where the basis vectors are described by the rows of $\mathbf{H}_{12}$ the transformation is only valid if it holds

$$\mathbf{h}_2^T \mathbf{p}_1 = \begin{pmatrix} h_{20} & h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} > 0. \tag{2.58}$$

The last row $\mathbf{h}_2^T$ of $\mathbf{H}_{12}$ is the direction of the principle axis of the second camera in the coordinate system of the first camera. Since the inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is dependent on the angle between both with $\mathbf{a}^T\mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\|\cos\angle(\mathbf{a},\mathbf{b})$, the perspective transformation becomes inconsistent if $\left|\angle(\mathbf{h}_2^T, \mathbf{p}_1)\right| \geq 90°$. Figure 2.15 shows how inconsistency can occur during the transformation of one single image. Note that for points located perpendicular to the optical axis of the corresponding view, the images will be at infinity.

For the parabolic transformation we have the problem of ambiguities due to the character of quadratic functions. That means that certain points of the mapping $\mathbf{p}_2 = \mathcal{T}_{par}(\mathbf{k}_{par}, \mathbf{p}_1)$ are local extrema from where on the order of pixel will be mirrored and pixels are mapped to positions already filled by other image regions. Those extrema are located where the partial derivatives of the mapping equal zero: $\partial x_2/\partial x_1 = 0$, $\partial x_2/\partial y_1 = 0$, $\partial y_2/\partial x_1 = 0$, and $\partial y_2/\partial y_1 = 0$. To interpret the null of this derivatives we can approximate them by the difference quotient for small Δ. As it can be deduced easily (see Figure 2.16) the places of the extrema in the images are located around the image points in image I_2 that have a projection direction perpendicular to the optical axis of Image I_1. Note that in Figure 2.16 the transformation direction is contrary to the one of Figure 2.15.

Overcoming both classes of limitations, scenario-based and projection-based, is one of the main tasks to solve in sprite-based video processing. In order to apply the derived 2D-mappings for robust mosaic generation of a wide range of video scenes we will present an solution for the projection-based limit in chapter 5.

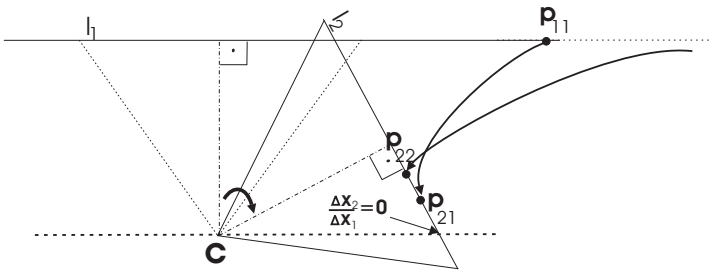


Figure 2.16: Limitations of the parabolic transformation: While point $\mathbf{p}_{11}$ can be mapped correctly onto the image plane of the second plane, point $\mathbf{p}_{12}$ is projected on already discovered image area. The area where $\Delta x_2/\Delta x_1$ equals zero will be the farthest area an original point can be mapped onto.

2.5 Physical Camera Parameters and Image Transformations

The homography $\mathbf{H_{12}}$ between two frames I_1 and I_2 can be estimated with maximally eight degrees of freedom, due to scale invariance of the projective space. In the next sections we will show that, based on the physical model for scene capturing, not all entries $h_{00}, \ldots, h_{21}$ are independent. Furthermore, this fact can be exploited to estimate the physical camera parameters of a scene without prior knowledge of the capturing setup. Generally, this estimation is also known as *camera calibration* and has been a research field of high interest for several years [167], [150].

For this work we follow the concept of decomposing the camera matrix into an intrinsic parameter matrix $\mathbf{K}$ containing camera specific internal parameters and an extrinsic parameter matrix $[\mathbf{R} \mid -\mathbf{R}\tilde{\mathbf{c}}]$ containing all scene specific external parameters (Equation 2.11) like rotation angles and distances. Since for real cameras, digital or analogue, skewness τ of the sampling grid can be omitted and the sampling ratio η can be set to one the internal camera matrix only depends on the focal length f and the coordinates of the principal point $(x'_o, y'_o)^T$, which can be set to $(0,0)^T$. Thus,

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2.5.1 Decomposition of Homographies for Rotational Camera Movement

Following Equation 2.23 a mapping of points $\mathbf{x}_1 = (x_1, y_1, 1)^T$ of image $I_1(\mathbf{x}_1)$ to points $\mathbf{x}_2 = (x_2, y_2, 1)^T$ on image $I_2(\mathbf{x}_2)$ is defined by

$$w\mathbf{x}_2 = \mathbf{K}_2\mathbf{R}_{12}\mathbf{K}_1^{-1}\mathbf{x}_1 = \begin{bmatrix} f_2 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R}_{12} \begin{bmatrix} \frac{1}{f_1} & 0 & 0 \\ 0 & \frac{1}{f_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_1. \tag{2.59}$$

Rotation matrix $\mathbf{R}_{12}$ signifies a rotation of 3D points around the center of the camera that captured I_1 and I_2 respectively. This decomposition is only valid for cameras fixed at the optical center, hence only rotational camera movements are allowed. $\mathbf{R}_{12} = \{r_{ij}\}$ with $i, j = 0 \dots 2$ is orthonormal matrix where all rows (r_i^T) and columns (r_j) are orthonormal, i.e.,

$$\sum_i r_{ij} r_{ik} = \begin{cases} 0, & j \neq k \\ 1, & j = k \end{cases}$$
$$\sum_j r_{ij} r_{kj} = \begin{cases} 0, & i \neq k \\ 1, & i = k \end{cases} .$$

The rotation matrix can be represented in many different ways. One possibility is a rotation φ about the rotation unit vector $\mathbf{v} = (v_x, v_y, v_z)^T$. $\mathbf{R}_{12}$ then yields

$$\mathbf{R}_{12} = \begin{bmatrix} \cos\varphi + v_x^2 (1 - \cos\varphi) & v_x v_y (1 - \cos\varphi) - v_z \sin\varphi & v_x v_y (1 - \cos\varphi) + v_y \sin\varphi \\ v_y v_x (1 - \cos\varphi) + v_z \sin\varphi & \cos\varphi + v_y^2 (1 - \cos\varphi) & v_y v_z (1 - \cos\varphi) - v_x \sin\varphi \\ v_z v_x (1 - \cos\varphi) - v_y \sin\varphi & v_z v_y (1 - \cos\varphi) + v_x \sin\varphi & \cos\varphi + v_z^2 (1 - \cos\varphi) \end{bmatrix}$$

Unfortunately the rotation axis, represented by $\mathbf{v}$, will be different for every pair of images I_1 and I_2 of a scene shot. Hence, the rotation angles φ_n for any frame I_n of a scene shot with respect to a certain reference frame are not comparable. It is still very difficult to follow the camera movement.

The use of Euler angles for the decomposition of homographies can be difficult because of possible singularities having impact on the numerical stability [24]. On the other hand the order of the three independent rotations (y-axis, x-axis, and z-axis) can be chosen arbitrarily and the three angles compose a metric space when calculated in relation to a certain reference frame I_0. Thus, we can write the homography $\mathbf{H}_{0,n}$ between any frame I_n and the reference frame as

$$\mathbf{H}_{0,n} = \mathbf{K}_n \cdot \mathbf{R}_{\varphi_z} \cdot \mathbf{R}_{\varphi_x} \cdot \mathbf{R}_{\varphi_y} \cdot \mathbf{K}_0^{-1} = \mathbf{R}_{\varphi_z} \cdot \mathbf{K}_n \cdot \mathbf{R}_{\varphi_x} \cdot \mathbf{R}_{\varphi_y} \cdot \mathbf{K}_0^{-1}, \tag{2.60}$$

where $\mathbf{R}_{\varphi_x}$, $\mathbf{R}_{\varphi_y}$, and $\mathbf{R}_{\varphi_z}$ represent single rotations about the x-axis, y-axis, and z-axis of the reference coordinate system. Hereby it is important to mention that the order of rotations is important but can be chosen arbitrarily and should be kept for all calculations.

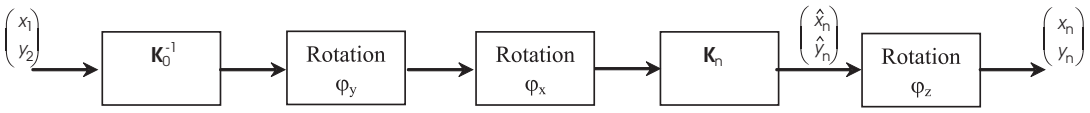


Figure 2.17: Scheme for decomposing homography $\mathbf{H}_{0,n}$ into several physical-based extrinsic and intrinsic parameter matrices. Note that vector $\hat{\mathbf{x}}_n$ and $\mathbf{x}_n$ lie in the same image plane and have the same vector norm: $\|\hat{\mathbf{x}}_n\| = \|\mathbf{x}_n\|$.

The rotation matrices are specified in the following:

$$\mathbf{R}_{\varphi_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_x & -\sin\varphi_x \\ 0 & \sin\varphi_x & \cos\varphi_x \end{bmatrix} \tag{2.61}$$

$$\mathbf{R}_{\varphi_y} = \begin{bmatrix} \cos\varphi_y & 0 & \sin\varphi_y \\ 0 & 1 & 0 \\ -\sin\varphi_y & 0 & \cos\varphi_y \end{bmatrix} \tag{2.62}$$

$$\mathbf{R}_{\varphi_z} = \begin{bmatrix} \cos\varphi_z & -\sin\varphi_z & 0 \\ \sin\varphi_z & \cos\varphi_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.63}$$

In this work we define the order of rotation according to the flowchart depicted in Figure 2.17 following Equation 2.60. Due to their properties, matrices $\mathbf{K}_n$ and $\mathbf{R}_{\varphi_z}$ can be swapped without any impact to the transformation result. As can be seen in Equations 2.59 to 2.63 homography $\mathbf{H}_{0,n}$ is only dependent on five parameters: f_0, f_n, φ_x, φ_y, and φ_z. Thus, the underlying image transformation has actually only five degrees of freedom. This *constrained perspective motion model* can further be exploited to roughly estimate the physical camera parameters from a set of registered images.

2.5.2 A Coarse and Robust Calibration Approach

For our calibration approach we assume the short-term homographies, i.e., the perspective transformation between temporal adjacent frames of a scene shot captured by rotating and zooming camera, as given. Methods for their robust estimation, i.e., *global motion estimation* (GME), will be discussed in detail in chapter 3. Since for natural scenes the global motion between two frames is rather small we do not have to deal with any limitations mentioned in section 2.4. The first step is the computation of long-term motion parameters with respect to a reference frame of the scene shot by recursively applying homography multiplication

$$\mathbf{H}_{0,n} = \mathbf{H}_{n-1,n} \cdot \mathbf{H}_{0,n-1}. \tag{2.64}$$

Here, $H_{a,b}$ with $a, b \in \{0, \ldots, N-1\}$ represents the long-term estimation of the image mapping between frame I_a and I_b of a scene shot consisting of N frames. Without loss of

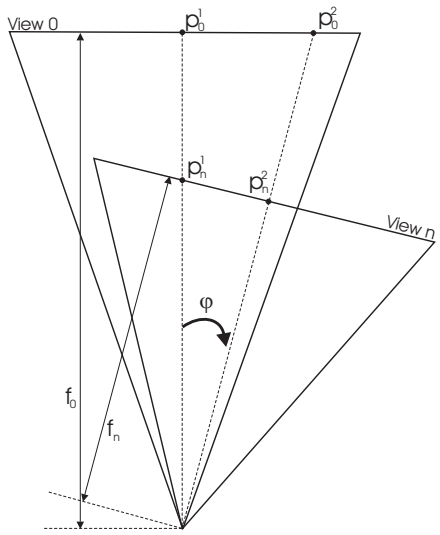


Figure 2.18: On the derivation of ratio $\alpha_{0,n}$ between the focal lengths of two views of a scene captured with rotating and zooming camera.

generality, frame I_0 can be chosen as reference frame. In this case short-term estimation errors will accumulate over the concatenation process. But for a rough estimate this is absolutely sufficient. With the long-term homographies we can approximately determine the image of every reference frame pixel. For further examination we especially concentrate on the image of the principal points, i.e., the intersection of the image plane and optical axis. This is exemplarily shown in Figure 2.18. The whole method for rough camera parameter estimation is shown in Algorithm 2.1.

Since for common cameras the principal point $\mathbf{p} = (x_p, y_p, 1)^T = (0, 0, 1)^T$ lies in the center of the image, the focal ratio can be computed with

$$\alpha_{0,n} = \frac{f_0}{f_n} = \frac{\|\mathbf{p}_0^2\|/\tan\varphi}{\|\mathbf{p}_n^1\|/\tan\varphi} = \frac{\|\mathbf{p}_0^2\|}{\|\mathbf{p}_n^1\|}. \tag{2.65}$$

$\mathbf{p}_n^1$ is the image of the principal point $\mathbf{p}_0^1$ of view 0 in view n and $\mathbf{p}_0^2$ the image of the principal point $\mathbf{p}_n^2$ of view n in view 0. See Figure 2.18 for derivation of Equation 2.65. Having the ratio $\alpha_{0,n}$ of focal length for all image pairs 0, n with $(n = 1, \ldots, N-1)$ the value for the focal length f_0 is computed as median value of all real solutions applying Szeliski's linear method [142]. Defining matrix $\mathbf{R}$ as

$$\mathbf{R} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix}$$

and considering scale invariance the homography $\mathbf{H}_{0,n}$ can be written as

$$\mathbf{H}_{0,n} = \frac{1}{\alpha_{0,n}} \begin{bmatrix} r_{00} & r_{01} & f_0 r_{02} \\ r_{10} & r_{11} & f_0 r_{12} \\ r_{20}\alpha_{0,n}/f_0 & r_{21}\alpha_{0,n}/f_0 & r_{22}\alpha_{0,n} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}. \tag{2.66}$$

Objective

Computing the physical camera parameters (f_n, φ_x^n, φ_y^n, φ_z^n) for rotational camera movement in a scene shot with a priory knowledge of the short-term homographies $\mathbf{H}_{n-1,n}$.

Algorithm

(i) Determine long-term parameters with respect to reference frame I_0 by concatenating short-term homographies following Equation 2.64.
(ii) Calculate the ratios $\alpha_{0,n}$ of focal length using the images of the principal points according to Equation 2.65.
(iii) Estimate the focal length f_0 of the reference frame I_0 as median value of all real solutions for the constraints of $\mathbf{H}_{0,n}$ derived by Szeliski's method. (Equations 2.66 to 2.68)
(iv) Calculate angles φ_y and φ_x by relating principal point $\mathbf{p}_n^2$ of frame I_n to its image in the reference frame according to Equation 2.70.
(v) Computation of angle φ_z about the optical axis of frame I_n using Equations 2.71 and 2.72

Algorithm 2.1: Robust and rough estimation of physical camera parameters.

Taking the orthogonality constraint for row vectors and column vectors of $\mathbf{R}$ as well as the vector norm constraint for rows and columns into account, we can arrange 12 independent equations for the calculation of focal length f_0. Algebraically this can be formulated as

$$\begin{aligned}
0 &= h_{00}h_{10} + h_{01}h_{11} + h_{02}h_{12}/f_0^2 \\
&= h_{00}h_{20}f_0/\alpha_{0,n} + h_{01}h_{21}f_0/\alpha_{0,n} + h_{02}h_{22}/(f_0\alpha_{0,n}) \\
&= h_{10}h_{20}f_0/\alpha_{0,n} + h_{11}h_{21}f_0/\alpha_{0,n} + h_{12}h_{22}/(f_0\alpha_{0,n}) \\
&= h_{00}h_{01} + h_{10}h_{11} + h_{20}h_{21}f_0^2/\alpha_{0,n}^2 \\
&= h_{00}h_{02}/f_0 + h_{10}h_{12}/f_0 + h_{20}h_{22}f_0/\alpha_{0,n}^2 \\
&= h_{01}h_{02}/f_0 + h_{11}h_{12}/f_0 + h_{21}h_{22}f_0/\alpha_{0,n}^2
\end{aligned} \tag{2.67}$$

for the orthogonality and

$$\begin{aligned}
h_{00}^2 + h_{01}^2 + h_{02}^2/f_0^2 &= h_{10}^2 + h_{11}^2 + h_{12}^2/f_0^2 \\
&= h_{20}^2 f_0^2/\alpha_{0,n}^2 + h_{21}^2 f_0^2/\alpha_{0,n}^2 + h_{22}^2/\alpha_{0,n}^2 \\
h_{00}^2 + h_{10}^2 + h_{20}^2 f_0^2/\alpha_{0,n}^2 &= h_{01}^2 + h_{11}^2 + h_{21}^2 f_0^2/\alpha_{0,n}^2 \\
&= h_{02}^2/f_0^2 + h_{12}^2/f_0^2 + h_{22}^2/\alpha_{0,n}^2
\end{aligned} \tag{2.68}$$

for equal vector norms of rows and columns of rotation matrix $\mathbf{R}$. Note, that we obtain 12 equations for every homography $\mathbf{H}_{0,n}$ with $n = 1, \ldots, N-1$. As experimental results show, this method yields accurate results for a minimum number of frames in a scene, which lies between 15 and 25 frames depending on the sequences. After the estimation of f_0 we simply can calculate all other focal length using Equation 2.65.

The rotation angles for every single frame are then estimated directly using the images of both principal points. In fact we firstly calculate angles φ_x and φ_y since the final z-rotation does not affect the points on the optical axis. Thus, from Equation 2.60 follows

$$w \cdot \mathbf{p}_0^2 = \mathbf{K}_0 \mathbf{R}_{\varphi_y}^{-1} \mathbf{R}_{\varphi_x}^{-1} \mathbf{K}_n^{-1} \mathbf{R}_{\varphi_z}^{-1} \mathbf{p}_n^2 = \mathbf{K}_0 \mathbf{R}_{\varphi_y}^{-1} \mathbf{R}_{\varphi_x}^{-1} \mathbf{K}_n^{-1} \mathbf{p}_n^2 = F_0 \begin{pmatrix} \sin\varphi_y \cos\varphi_y \\ \sin\varphi_x \\ \cos\varphi_y \cos\varphi_y \end{pmatrix}, \tag{2.69}$$

with $\mathbf{p}_n^2 = (0,0,1)^T$ and $\mathbf{p}_0^2 = \mathbf{H}_{0,n}^{-1} \mathbf{p}_n^2$, which yields in

$$\mathbf{p}_0^2 = \begin{pmatrix} f_0 \tan\varphi_y \\ f_0 \frac{\tan\varphi_x}{\cos\varphi_y} \\ 1 \end{pmatrix}. \tag{2.70}$$

After calculating angles φ_y and φ_x we can determine the image of principal point $\mathbf{p}_0^1$ in the reference frame before ($\hat{\mathbf{p}}_n^1$) and after ($\mathbf{p}_n^1$) the final z-rotation using

$$\begin{aligned} w_1 \cdot \hat{\mathbf{p}}_n^1 &= \mathbf{K}_n \mathbf{R}_{\varphi_x} \mathbf{R}_{\varphi_y} K_0^{-1} \cdot \mathbf{p}_0^1 \\ w_2 \cdot \mathbf{p}_n^1 &= \mathbf{H}_{0,n} \cdot \mathbf{p}_0^1, \end{aligned} \tag{2.71}$$

with $\mathbf{p}_0^1 = (0,0,1)^T$. φ_y is now calculated as difference of the absolute angles of vector $\hat{\mathbf{p}}_n^1 = (\hat{x}_n^1, \hat{y}_n^1, 1)^T$ and $\mathbf{p}_n^1 = (x_n^1, y_n^1, 1)^T$ with the horizontal line.

$$\begin{aligned} &\varphi_z = \varphi_{z,p} - \varphi_{z,\hat{p}} \quad \text{with} \\ &\tan\varphi_{z,p} = \tfrac{y_n^1}{x_n^1} \quad \text{and} \quad \tan\varphi_{z,\hat{p}} = \tfrac{\hat{y}_n^1}{\hat{x}_n^1} \end{aligned} \tag{2.72}$$

2.5.3 Evaluation of Parameter Estimation Quality

In order to asses the quality and exactness of the described camera estimation method (Algorithm 2.1), two different evaluation methods were conducted. First, ground truth data was generated by rendering a virtual scene shot using Autodesk's commercial rendering software 3ds Max. The advantage of this method is that the transitions between different camera poses can be defined exactly. Thus, a calculation of the camera parameters as ground truth data is facilitated. The drawbacks are the lack of visual distortions and image noise since the virtual cameras represent only a model of the real camera optics. In the experiments we rendered 150 views of a virtual room. The sequence "TUB-room1" (see Figure 2.19) is rendered over 151 frames with four partially overlapping camera movements: horizontal pan, vertical tilt, roll (in-plane rotation), and zoom. Table 2.1 gives an overview of the order of camera movement.

The ground-truth-based evaluation of the robust coarse calibration approach is shown in Figures 2.20 and 2.21. The maximum relative error for the focal length is less than 2% and the maximum angle deviation is about 2 degree. In order to strengthen the parameter estimation results, the vector norms $\left\|\hat{\mathbf{p}}_n^1\right\|$ and $\left\|\mathbf{p}_n^1\right\|$ (images of principal point $\mathbf{p}_0^1$ in the

Figure 2.19: Exemplary frames of rendered sequence "TUB-room": frame 0, 50, 100, 150 (left to right - original size: 352x240)

Frame Numbers	0 . . . 49	50 . . . 99	100 . . . 150
pan / degree	0 . . . 20	20 . . . 0	. . .
tilt / degree	. . .	0 . . . 20	20 . . . 0
roll / degree	. . .	. . .	0 . . . − 20
zoom / pixels	383 . . . 312	312 . . . 383	383 . . . 492

Table 2.1: Overview of camera movements for artificial scene "TUB-room1"

reference frame) are compared. Because they are obtained in two different ways according to Equation 2.71, their difference is a measure for exactness of the parameter estimation. Smaller values indicate a more exact estimation. Note, in the ideal case both points have equal distance to the principal point. The diagram in Figure 2.22 depicts the distance differences $\left\|\hat{\mathbf{p}}_n^1\right\| - \left\|\mathbf{p}_n^1\right\|$ over all frames of the sequence. The maximal deviation for sequence "TUB-room1" is only about $4 \cdot 10^{-14}$ pixels.

As second method of evaluation we compare the calibration results with values obtained from a commercial calibration tool. Here, 2d3's software Bojou was used, which is a *state of the art* implementation for unconstrained calibration tasks. Extrinsic and intrinsic camera parameters for the well-known "Stefan" test sequence (352x240 - 300 frames) were estimated in this experiment. Figure 2.23 shows the obtained focal length for this sequence compared to the result calculated with Boujou. The relative difference is about 8.5%. Note that the commercial result is no real ground truth data. The rotation angles (Figure 2.24) are very similar to those estimated with nonlinear methods in Farin's approach [24]. As indication of the calibration accuracy of this method again the vector difference for the images of the principle points $\left\|\hat{\mathbf{p}}_n^1\right\|$ and $\left\|\mathbf{p}_n^1\right\|$ are given. (Figure 2.25). The maximum difference is about $1.5 \cdot 10^{-13}$ pixels.

2.6 Chapter Summary

In this chapter an introduction into geometrical fundamentals for 2D image registration was given. Utilizing the advantages of projective geometry we examined the conditions for which those two-dimensional transformations are applicable. The most important case is a rotational camera movement where the camera center is fixed in space. Those types of cameras are also called PTZ (pan tilt zoom) cameras. Also recording settings that do only approximate the ideal camera are covered by those transformation models. In other words,

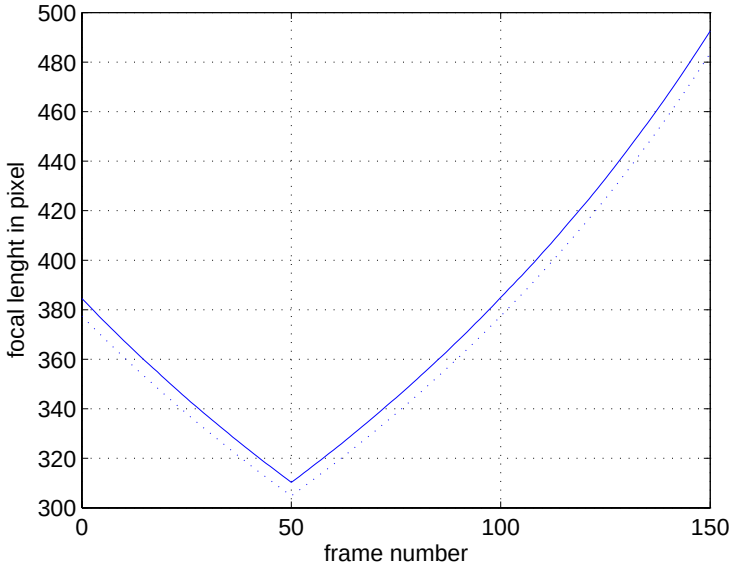


Figure 2.20: Comparison of estimation (solid) with ground truth values (dashed) for focal length of sequence "TUB-room1". The curvilinear behavior derives from the definition in 3ds Max. Not the focal length can be defined but the *field of view* (FOV), i.e., the angle of view dependent on focal length and the sensor dimensioning.

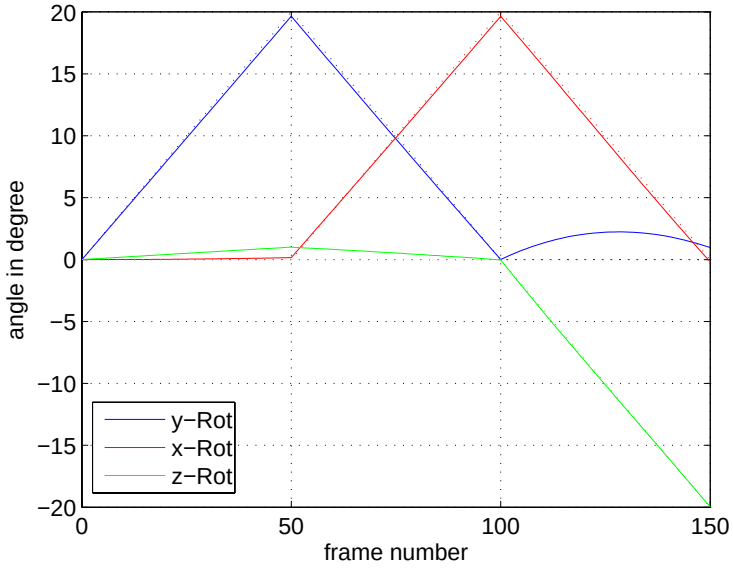


Figure 2.21: Comparison of estimation (solid) with ground truth values (dashed) for rotation angles of sequence "TUB-room1"

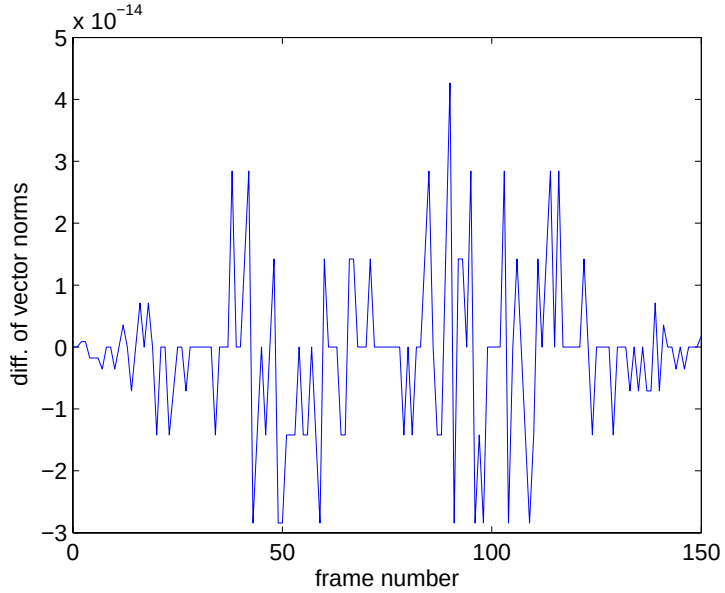


Figure 2.22: Difference $\left\|\hat{\mathbf{p}}_n^1\right\| - \left\|\mathbf{p}_n^1\right\|$ for vector norm of images $\hat{\mathbf{p}}_n^1$ and $\mathbf{p}_n^1$ of principal point $\mathbf{p}_0^1$ (view 0), sequence "TUB-room1": Small values indicate an accurate estimation of rotation angles φ_y and φ_x and focal length f_n with respect to homography $\mathbf{H}_{0,n}$ and its inverse $\mathbf{H}_{n,0} = \mathbf{H}_{0,n}^{-1}$.

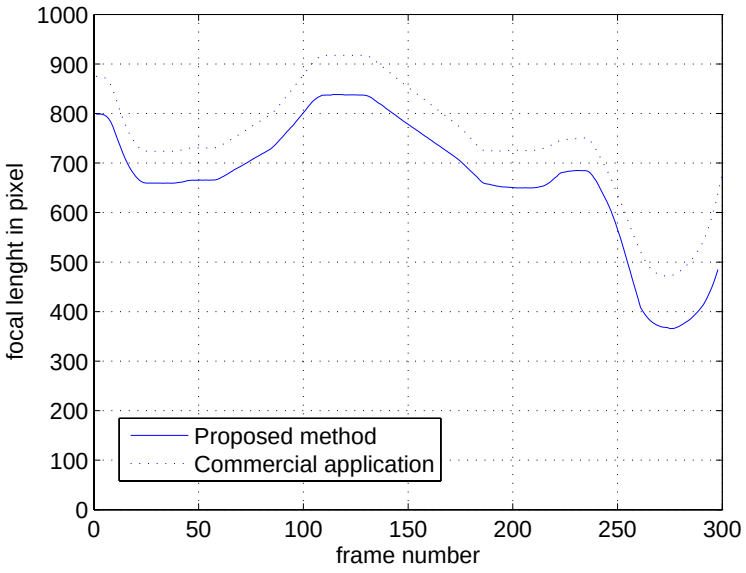


Figure 2.23: Comparison of estimation (solid) with values obtained by a commercial software (dashed) for focal length of sequence "Stefan".

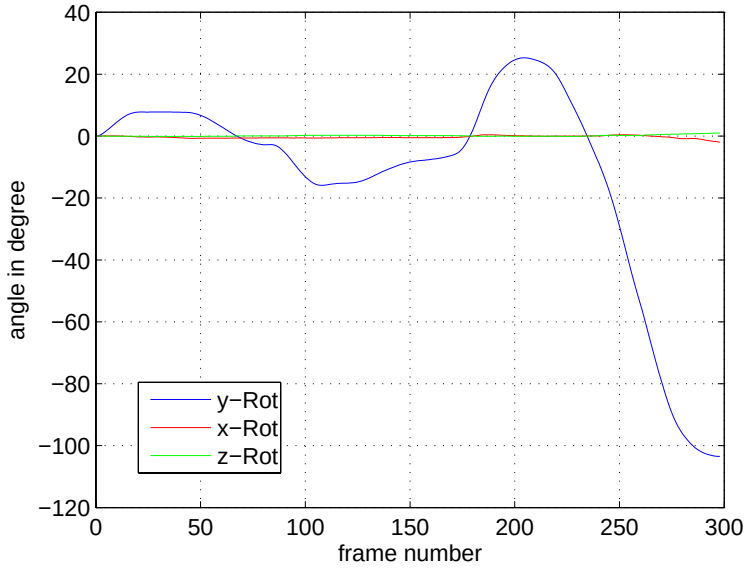


Figure 2.24: Results for rotation angle estimation for sequence "Stefan".

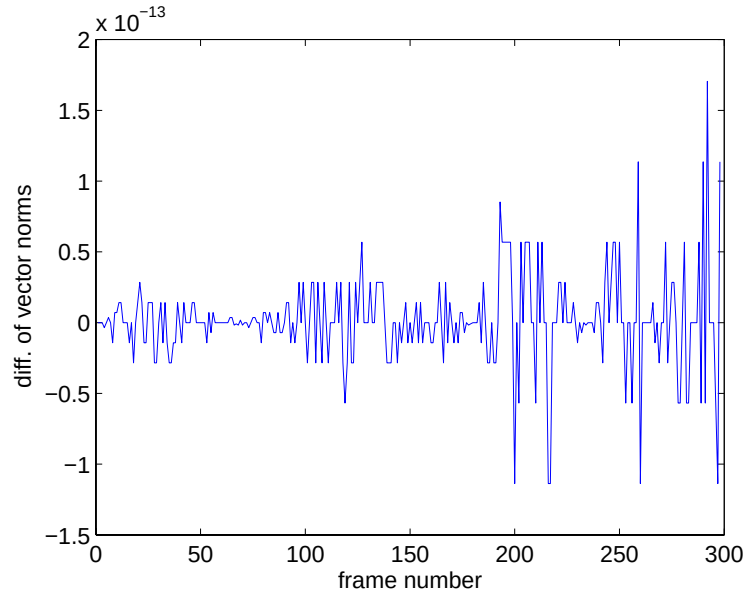


Figure 2.25: Difference of vector magnitudes of reference principle point images (sequence "Stefan").

if the camera translation is rather small with respect to the objects depth, 2D registration still yields good results in terms of background re-projection.

In the middle part of this chapter, parametrized 2D models where derived based on the physical fundamentals for a simplified pinhole camera model. As has been shown the 8 parameter perspective model is the appropriate transformation. But also first and second order approximations based on Taylor series expansion can be useful under certain circumstances. Especially the 12 parameter second order parabolic model promises good registration results due to a higher degree of freedom in the parameter estimation process. Since it is nonlinear its application for mosaic generation comes along with higher computational complexity.

Finally, a rough and robust physical parameter estimation algorithm was presented. Based on short-term frame-to-frame homographies we proposed a new and low complex approach to estimate internal and external camera parameters for rotating and zooming cameras. As the results show, the differences between estimations and ground truth data for angles and focal length are acceptable small. In the next chapters we will use the results of the calibration approach to construct multiple sprites in order to improve the registration and to minimize the sprite image sizes.

Chapter 3

2D Image Registration and Single Sprite Generation

Every object [...] is perfect if it can serve its purpose.

Gotthold Ephraim Lessing (1729-81), from 'Philotas'

Chapter 2 presented a way of description of camera movements with the affine, perspective, and parabolic transformation model. In this chapter, we address the problem of transformation parameter estimation from recorded scene shots. In the second part of this chapter we also describe how we can summarize the background of the scene by creating background sprites based on the estimated 2D transformation.

The transformation parameter computation between two images is mainly achieved in two steps. First, a robust method is used to approximately calculate the transformation between two images. In order to increase the robustness, i.e., a low sensitivity against image noise and geometric deviations, the degree of freedom is reduced by applying transformations with small numbers of parameters. This first estimation is achieved using linear methods. This is also referred to as robust parameter initialization (Section 3.1.1). Second, in several steps the transformation is expanded to a higher order model and the parameter estimation is refined minimizing an image consistency-based energy term utilizing a numerical energy minimization framework.

The short-term transformations $\mathcal{T}_{n-1,n}$ are then the starting point to initialize a long-term parameter estimation $\mathcal{T}_{r,n}$ with respect to any reference frame I_r of the shot. We will derive an approach of preliminary sprite generation combined with direct frame-to-mosaic registration to overcome the error accumulation problem connected with the concatenation of adjacent short-term transformations.

Finally, a single background sprite is constructed by blending all long-term registered global motion-compensated images into the coordinate system of the reference frame. The blending is important to minimize the difference between sprite and sequence pixels due to illumination variation, image noise, and color drifts. In order to filter out the foreground objects sophisticated blending methods can be used too.

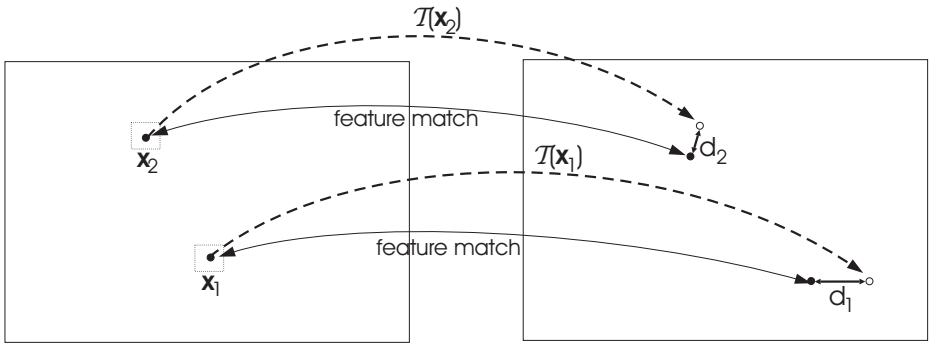


Figure 3.1: Principle of feature matching based registration: A priori computed feature pairs are used to determine the optimal image transformation that minimizes the distances (d_1, d_2) between matching partners and transformed image points $\mathcal{T}(\mathbf{x}_i)$.

3.1 2D Registration Approaches

There exist various methods to register two or more images utilizing 2D image transformations. Generally they can be classified into two categories: (1.) *feature based approaches* and (2.) *photometric consistency based approaches* [9]. For both categories the goal is to find the best transformation, i.e., to find the optimal parameter set of an a priori specified transformation model that minimizes some registration error metrics. In case of feature based registration the metrics is the distance between transformed feature points and previously tracked features. For the photometric consistency based methods the metrics is based on correlation measures between the compensated overlapping images. While the former method is sparse and spatially distance based the latter is dense and integrates the measure of color or illumination values.

3.1.1 Feature-based Approaches

Feature based methods for the estimation of 2D image transformation are very popular due to their low computational complexity [8]. The main argument for their utilization is that only the strong correspondences between two images have impact on the result of the parameter computation. That means that established feature correspondences have a high likelihood to be supportive for the calculation of the appropriate mapping. Thus, the number of outliers caused by wrong feature allocation due to ambiguities, homogeneities or aperture is comparatively small. Nevertheless, outliers will still exist and have to be taken into account. In order to estimate the image transformation feature correspondences have to be established first. In most cases strong image corners are fully satisfactory. Only in special cases edges or features of higher semantic abstraction are utilized. Well investigated corner detectors are the *Harris corner detector* [41] or the Hessian criteria which marks points where change of intensity derivations is maximally presented by

$$c = \max_{\forall x,y} \det\left(\mathcal{H}(I(x,y))\right) = \max_{\forall x,y} \det \begin{bmatrix} \frac{\partial^2 I(x,y)}{\partial x^2} & \frac{\partial^2 I(x,y)}{\partial x \partial y} \\ \frac{\partial^2 I(x,y)}{\partial x \partial y} & \frac{\partial^2 I(x,y)}{\partial y^2} \end{bmatrix}. \tag{3.1}$$

To find the correspondences in the second frame, correlation based matching for a feature support region – usually a rectangular area containing all neighboring pixels – within a certain search range is performed. As matching criterion the *normalized cross-correlation*, the *sum of squared differences* (SSD), or the *sum of absolute differences* (SAD) is used.

Once feature pairs in both images are established, the best fitting transformation can be estimated. We distinguish between two types of estimators to achieve this goal:

- Linear estimation and
- Distance based *maximum likelihood* optimization.

Both approaches minimize the difference d_i between measured feature correspondences $\mathbf{x}'_i$ using the matching algorithm and the transformed feature positions $\hat{\mathbf{x}}'_i = \mathcal{T}(\mathbf{x}_i)$, which can be written as

$$\mathcal{T}_{opt} = \arg\min_{\mathcal{T}} \sum_{i=1}^{N_f} d_i^2 = \arg\min_{\mathcal{T}} \sum_{i=1}^{N_f} \left|\mathbf{x}'_i - \mathcal{T}(\mathbf{x}_i)\right|^2, \tag{3.2}$$

where N_f is the number of feature correspondences between the two images. Figure 3.1 schematically shows how feature based methods work. Some likelihood based approaches do minimize both, forward differences d_i and backward differences $\bar{d}_i$ with $\bar{d}_i = \mathbf{x}'_i - \mathcal{T}^{-1}(\mathbf{x}'_i)$ so that

$$\mathcal{T}_{opt} = \arg\min_{\mathcal{T}} \sum_{i=1}^{N_f} d_i^2 + \bar{d}_i^2 = \arg\min_{\mathcal{T}} \sum_{i=1}^{N_f} \left|\mathbf{x}'_i - \mathcal{T}(\mathbf{x}_i)\right|^2 + \left|\mathbf{x}_i - \mathcal{T}^{-1}(\mathbf{x}'_i)\right|^2. \tag{3.3}$$

In Equation 3.3 the inverse transformation $\mathcal{T}^{-1}$ is used together with $\mathcal{T}$. This is only useful if the transformation is easily and unique invertible. Thus, it is not applicable for higher order transformations, e.g., the proposed parabolic image mapping $\mathcal{T}_{par}$.

3.1.1.1 Linear Feature-based Estimation

Since linear methods provide a closed-form solution for the parameter estimation, they are easy to compute and are often preferred to the maximum likelihood approaches. The solution of the linear method is derived in the following: Having a number N_f of point correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$ in two images $I(\mathbf{x})$ and $I'(\mathbf{x}')$ we can write N_f equations, assuming that $\mathbf{x}'_i = \mathcal{T}_{opt}(\mathbf{x}_i)$

$$\begin{aligned} 0 &= \mathbf{x}'_1 - \mathcal{T}_{opt}(\mathbf{x}_1) \\ &\vdots \quad\quad \vdots \\ 0 &= \mathbf{x}'_{N_f} - \mathcal{T}_{opt}(\mathbf{x}_{N_f}). \end{aligned} \tag{3.4}$$

Because every of the equations above consists of two linear equations – one for each image dimension –, we obtain a system of equations with $2N_f$ general independent equations that are linear in the transformation parameter set. Therefore, with a minimum number of

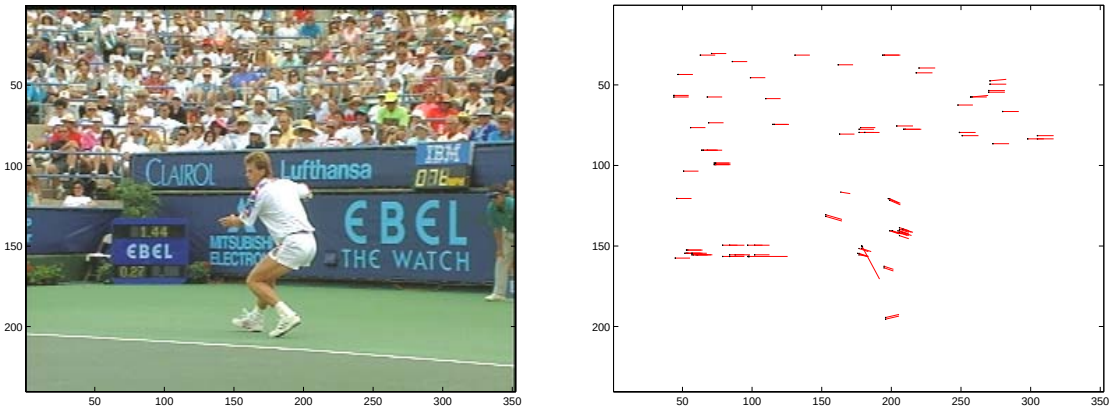


Figure 3.2: Motion vectors of feature points of sequence "Stefan": Original frame 94 (left); Motion vectors of feature points (right). The outliers in the region of the foreground objects disturb the linear estimation. Source: [145]

$N_f \geq N_k/2$ point correspondences we obtain an overdetermined system of equations. Here, N_k represents the number of parameters for a specific transformation model. Rewriting Equation 3.4 in matrix form we get

$$b = \mathbf{B}\mathbf{k}, \tag{3.5}$$

where $\mathbf{k}$ is the parameter vector of the underlying transformation model. For simple translational motion $\mathbf{k}_{tr} = (a_0, b_0)^T$, for affine motion $\mathbf{k}_{aff} = (a_0, a_1, a_2, b_0, b_1, b_2)^T$, for perspective motion $\mathbf{k}_{pers} = (a_0, a_1, a_2, b_0, b_1, b_2, c_1, c_2)^T$, and for parabolic motion $\mathbf{k}_{par} = (a_0, \cdots, a_5, b_0, \cdots, b_5)^T$. Matrix $\mathbf{B}$ is a model specific designed matrix to fulfill Equation 3.4 and vector $\mathbf{b}$ is the inhomogeneous part. Both are specified for different 2D image transformations in Appendix A.4.

The solution of this linear problem is achieved by simple pseudo-inversion of "design matrix" $\mathbf{B}$.

$$\mathbf{k} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{b} \tag{3.6}$$

The estimated parameter vector is the optimal estimation $\mathbf{k}_{opt}$ in a least-square sense.

3.1.1.2 Robust Linear Estimation

The drawback of the above derived linear solution is the handling of outliers. Outliers are feature pairs that do not support the optimal transformation. This could be due to image noise, mismatches due to ambiguities, and matches on independently moving foreground objects. The goal of a robust linear estimation is to suppress the impact of those outliers. In Figure 3.2 the estimated motion vectors – difference between corresponding feature points – are shown. In the region of the foreground object several vectors do not point onto the horizontal direction induced by camera motion. In order to minimize the influence of the outliers two methods can be used:

- *M-estimation* or

- *Random sample consensus*-based methods (RANSAC).

M-estimator for outlier elimination

The M-estimator [48] – "M" for maximum likelihood-type – is a method for *robust regression* that is not as vulnerable as direct least square methods. This is achieved by introducing a functionality $\rho(e)$ where e is the energy term to be minimized robustly. In our case the e is defined in Equation 3.2 and thus the new optimization term is

$$\mathcal{T}_{opt} = \arg\min_{\mathcal{T}} \sum_{i=1}^{N_f} \rho\left(e_i\right) = \arg\min_{\mathcal{T}} \sum_{i=1}^{N_f} \rho\left(d_i\right), \tag{3.7}$$

where d_i represents the L_2 norm of the distance vectors between feature point and transformed corresponding feature point. In [129] Smolic proposed to use the L_1 norm, which physically penalizes diagonal derivations compared with only horizontal or vertical derivations. In case of simple least square minimization the functionality $\rho(e) = e^2$. Here, greater values e_i have more impact into the minimization and thus, a linear estimator is forced to especially minimize that influence. The idea behind M-estimation is to saturate the impact of extreme large energy terms by carefully designing a $\rho(e)$. Typical functionalities are

- Huber's estimator: $\rho(e) = \begin{cases} \frac{1}{2}e^2 & \text{for} \quad |e| \leq c \\ c|e| - \frac{1}{2}c^2 & \text{for} \quad |e| > c \end{cases}$
- Bisquare estimator: $\rho(e) = \begin{cases} \frac{c^2}{6}\left\{1 - \left[1 - \left(\frac{e}{c}\right)^2\right]^3\right\} & \text{for} \quad |e| \leq c \\ c^2/6 & \text{for} \quad |e| > c \end{cases}$

Tuning constant c can also be written as $c = q\mu_e$, where μ_e is a measure for the average deviation. The bisquare estimator is also known as *Tukey's biweight* estimator. In order to determine the solution of Equation 3.7, the term $\sum \rho\left(d_i\right)$ has to be differentiated with respect to transformation parameter vector $\mathbf{k}$ and set to zero. Since the estimator functions are always functions of e_i^2, one has to solve

$$\sum_{i=1}^{N_f} \rho'\left(d_i\right) \nabla_{\mathbf{k}}(d_i) = 0 = \sum_{i=1}^{N_f} d_i w\left(d_i\right) \nabla_{\mathbf{k}}(d_i), \tag{3.8}$$

where $w(d_i)$ is a weighting function, with $w(d_i) = \rho'(d_i)/d_i$. The weighting functions are defined as follows:

- Huber's estimator: $w(e) = \begin{cases} 1 & \text{for} \quad |e| \leq c \\ c/|e| & \text{for} \quad |e| > c \end{cases}$
- Bisquare estimator: $w(e) = \begin{cases} \left[1 - \left(\frac{e}{c}\right)^2\right]^2 & \text{for} \quad |e| \leq c \\ 0 & \text{for} \quad |e| > c \end{cases}$

For certain applications it is useful to calculate with a low complex binary weighting function, that is:

$$w(e) = \begin{cases} 1 & \text{for} \quad |e| \leq c \\ 0 & \text{for} \quad |e| > c \end{cases}$$

For constant weights $w_i = w(d_i)$ Equation 3.8 is a least mean square solution for a weighted error term

$$\mathcal{T} = \arg\min_{\mathcal{T}} \sum_{i=1}^{N_f} w_i \cdot (d_i)^2 \,. \tag{3.9}$$

The weighting function can be incorporated into the closed form linear solution of Equation 3.6 by introducing a diagonal matrix $\mathbf{W} \in \mathbb{R}^{2N_f \times 2N_f}$. It may now be written as

$$\begin{aligned} \mathbf{k} &= (\mathbf{B}^T\mathbf{W}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}\mathbf{b}, \\ \text{diag}(\mathbf{W}) &= \big(w(d_1), w(d_1), \ldots, w(d_{N_f}), w(d_{N_f})\big)^T . \end{aligned} \tag{3.10}$$

The elements of diag($\mathbf{W}$) are the weights obtained by utilizing the results of the Bisquare estimator for all feature pairs.

Unfortunately, the weights w_i are not independent of the solution for $\mathcal{T}_{opt}$. Thus, the weighting has to be conducted iteratively until the weights converge. Therefore, the full M-Estimation approach is an iterative *fixed point* algorithm where every new parameter estimation follows a tuning of the weights, which are used for the next parameter estimation step. Algorithm 3.1 shows the whole robust estimation procedure. As estimation for the average deviation μ_e we use

$$\mu_e^k = \frac{1}{N_f} \sum_{i=1}^{N_f} \left\| x_i' - \mathcal{T}^k(x_i) \right\| , \tag{3.11}$$

where $\mathcal{T}^k$ is the kth solution of Equation 3.10.

We conducted different experiments in order to determine which 2D transformation model yields best results for robust feature-based estimation applying the M-Estimation approach. As well we were interested in empirically finding the optimal parameter q for tuning the weighting function for outlier suppression. In Figures 3.3 to 3.6 we show the objective quality comparison of the overlapping background regions for compensated adjacent image pairs for sequence "Stefan". The foreground was filtered out by an already provided alpha mask. For all four different motion models tuning parameter q was varied from 2 to 4. We found out that in average for all motion models a value $q \approx 2 \cdots 3$ is the best choice, although in many cases the tuning constant does not have much influence.

In Figure 3.7 and 3.8 we directly compare the short term compensation results for the sequence background for all four motion models: translation, affine, perspective, and parabolic. We found out that in case of only feature-based estimation the affine yields the best result. This is due to an good approximation of the real physical model by having less degree of freedom. Thus, the influence of outliers can be minimized.

Objective

Robustly estimating the optimal transformation $\mathcal{T}$ from N_f feature pairs using the M-Estimation approach.

Algorithm

(i) Initialize the weighting matrix $\mathbf{W} \in \mathbb{R}^{2N_f \times 2N_f}$ as identity matrix
(ii) Determine the solution for Equation 3.10 by using the appropriate design matrix $\mathbf{B}$ and vector $\mathbf{b}$.
(iii) Estimate the average deviation μ_e derived from Equation 3.11.
(iv) Determine the weights w_i for every feature pair (x_i, x_i') using the Bisquare estimator weighting function.
(v) Normalization of all weights w_i by applying $w_i = \frac{w_i}{\sum_i w_i}$.
(vi) If the differences $\max_i |w_i^k - w_i^{k-1}| > \epsilon$ than goto number (ii), otherwise stop the iteration.

Algorithm 3.1: Iterative M-estimation approach for feature-based registration of adjacent frames in a scene shot.

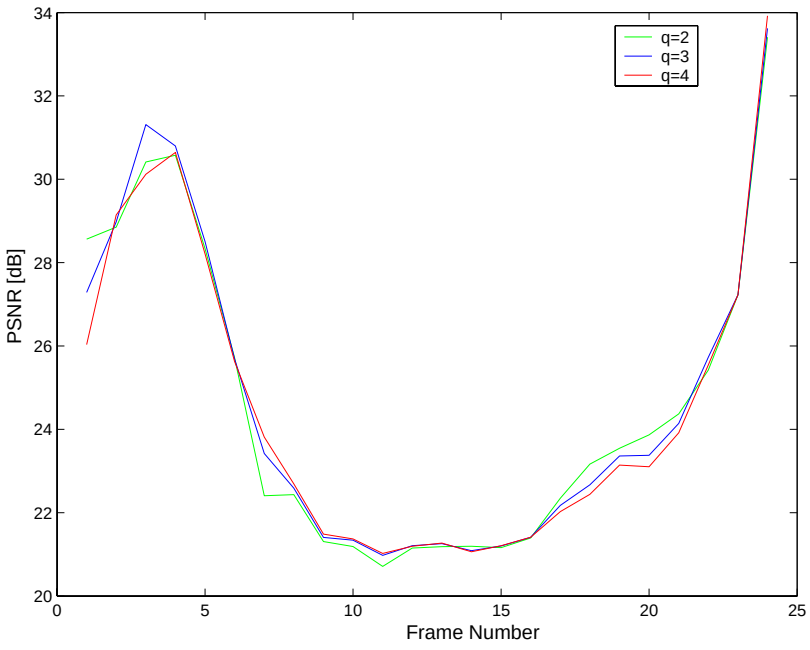


Figure 3.3: Comparison of feature-based short-term registration quality for different tuning constants q using the Bisquare estimator weighting function with the two parameter translational motion model for sequence "Stefan". Source: [145]

RANSAC-based Methods

The Random Sample Consensus (RANSAC) method [32] is a Monte Carlo method based on a randomly chosen subset of point pair estimates in order to determine an optimal transformation, which discriminates inliers from outliers. It's advantage is that it can cope

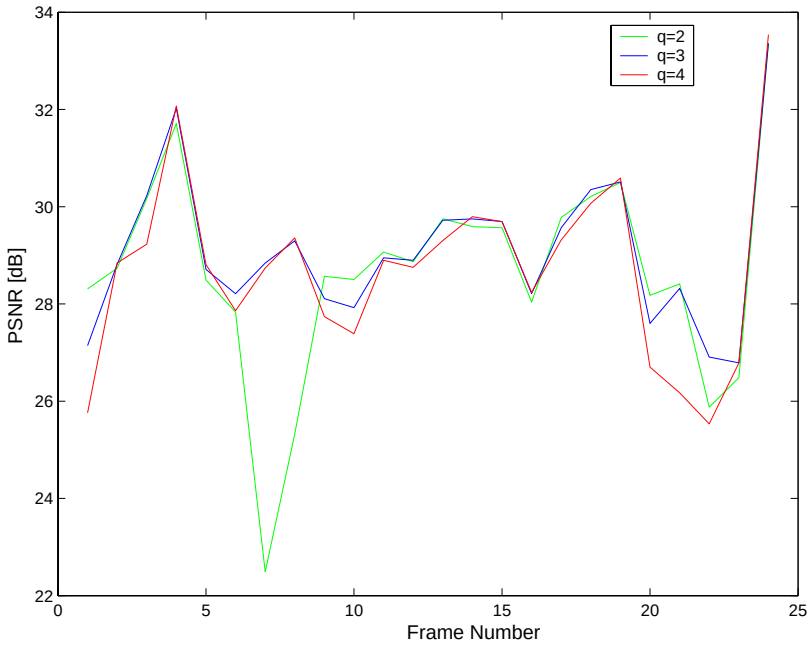


Figure 3.4: Comparison of feature-based short-term registration quality for different tuning constants q using the Bisquare estimator weighting function with the six parameter affine motion model for sequence "Stefan". Source: [145]

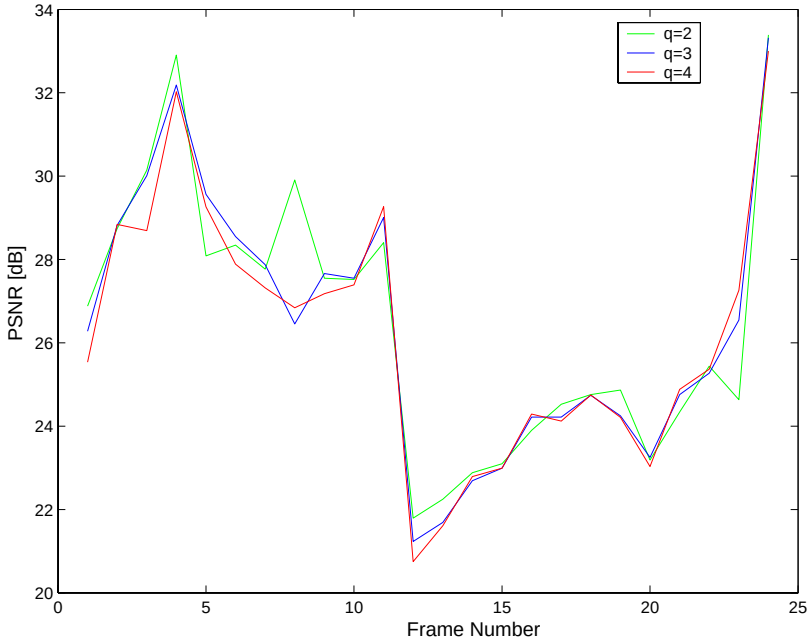


Figure 3.5: Comparison of feature-based short-term registration quality for different tuning constants q using the Bisquare estimator weighting function with the eight parameter perspective motion model for sequence "Stefan". Source: [145]

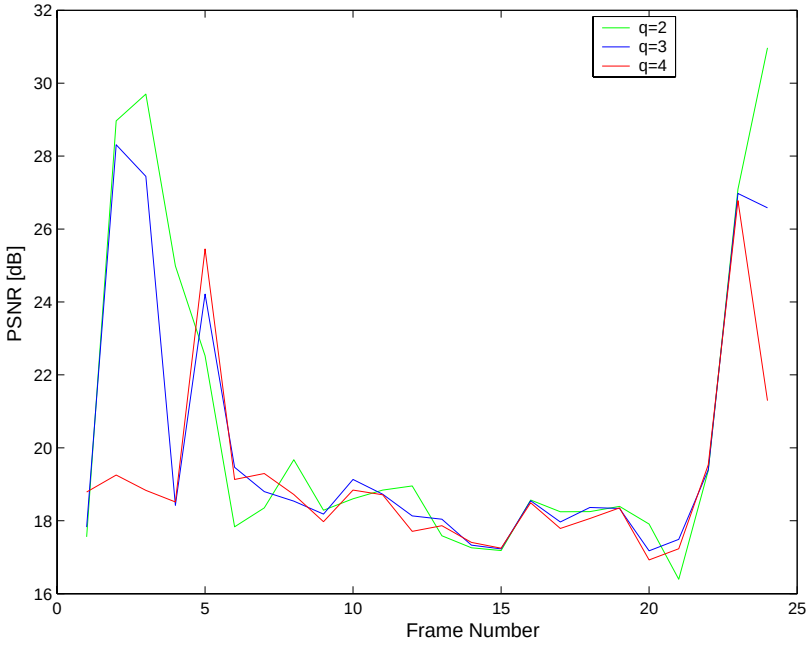


Figure 3.6: Comparison of feature-based short-term registration quality for different tuning constants q using the Bisquare estimator weighting function with the twelve parameter parabolic motion model for sequence "Stefan". Source: [145]

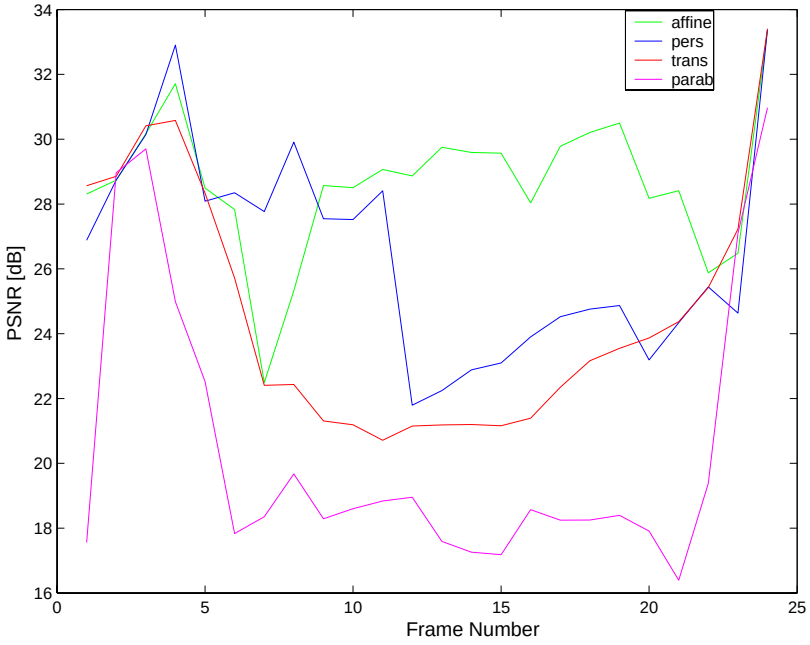


Figure 3.7: Comparison of feature-based short-term registration quality for different motion models and fixed M-estimator tuning constant $q = 2$. (Sequence "Stefan") Source: [145]

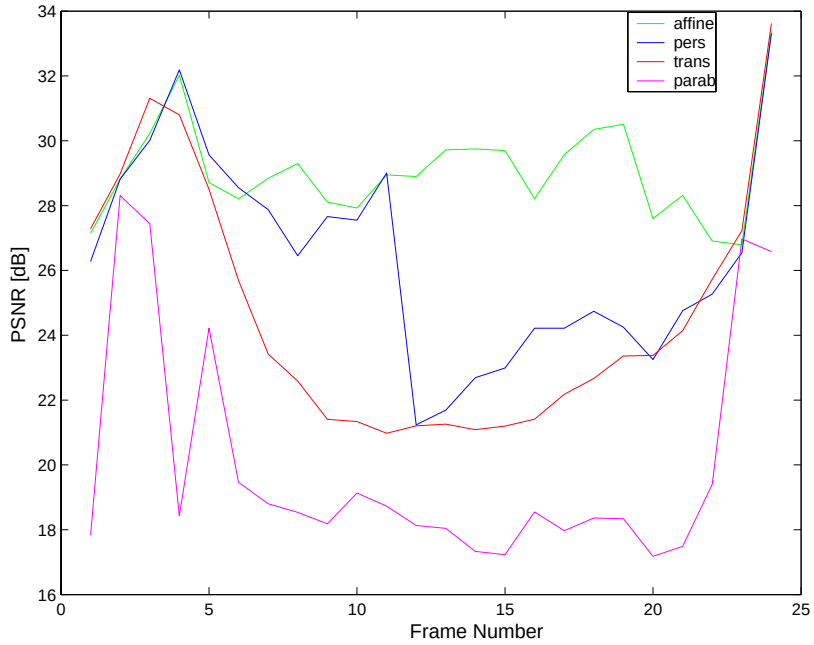


Figure 3.8: Comparison of feature-based short-term registration quality for different motion models and fixed M-estimator tuning constant $q = 3$. (Sequence "Stefan") Source: [145]

with a very large proportion of outliers, usually better than robust M-estimators. The basic idea is to select only a minimum sample of point pairs that are necessary to solve the equation system for a certain type of transformation. Depending of an roughly assumed outlier ratio we can a priory determine how often this random point selection has to be repeated until we find a transformation with a predetermined certainty that accurately describes the mapping for all inliers. The number N of selections is described with

$$N = \frac{\log(1-p)}{\log\left(1-(1-\varepsilon)^s\right)}. \tag{3.12}$$

The variable p represents the probability that at least one subsample of s points is free of outliers. It is usually set to 0.99. ε is the probability that one selected data point or feature pair is an outlier and equals the percentage of outliers for an image pair. It is usually set higher than expected in a range of 40% or more. The minimum number s of point correspondences of a subsample is dependent on the 2D image transformation model. Since every feature pair defines two equations, s is usually half the number of model parameters, i.e., entries in the parameter vector $\mathbf{k}$.

As measure of accuracy of any randomly determined subsample the number S_i of inliers to the specific estimation is counted. The estimated transformation $\mathcal{T}(\mathbf{k})$ using Equation 3.6 that belongs to the largest S_i can then be chosen as "winner". Optionally, as further optimization it can be helpful to reestimate $\mathcal{T}$ by entering all inliers to the least square solution of Equation 3.6. Thus, the variance within the inliers subsample is minimized.

The discrimination of outliers from inliers is accomplished on a statistical basis. If we assume the measurement error of feature correspondences to be 2D Gaussian distributed

Objective

Robustly estimating the optimal transformation $\mathcal{T}$ from N_f feature pairs using a RANSAC-based approach.

Algorithm

(i) Determine the number N of random subsamples in order to have an outlier-free tuple using Equation 3.12
(ii) Choose randomly a minimal number of feature pairs and calculate the transformation vector $\mathbf{k}$ following Equation 3.6
(iii) Determine the Number S_i of inlier amongst all feature pairs that have a distance $d_i = \|x'_i - \mathcal{T}(x_i)\| < t$. (Threshold t is estimated in Equation 3.13.)
(iv) If $S_i > S_{max}$ then set $\mathcal{T}_{opt} = \mathcal{T}$ and $S_{max} = S_i$.
(v) Goto (ii) until the number of subsample-based estimations equals N.
(vi) Reestimate $\mathcal{T}_{opt}$ by choosing all inlier feature pairs for solving Equation 3.6

Algorithm 3.2: RANSAC-based approach for feature-based registration of adjacent frames in a scene shot.

with zero mean the squared distance d_i^2 as squared sum of Gaussian of that error is a χ_2^m distribution with $m = 2$ for all image mappings. In [42] the discrimination threshold t is given as

$$t^2 = 5.99\sigma^2, \tag{3.13}$$

with σ representing the standard deviation of the measurement error. If the threshold is set to the value of Equation 3.13 95% of all inliers will be regarded as inliers. The value of the standard derivation σ of the measurement error is usually set empirically and is normally a subpixel value. A good empirical value is $\sigma = 0.5$ pixels, which was also identified in [116]. Algorithm 3.2 describes the approach of robust feature-based estimation of 2D image transformations using the RANSAC method.

We conducted experiments using RANSAC-based methods and calculated the global motion compensated background PSNR for sequence "Stefan" for different motion models (see Figures 3.9 and 3.10). We found out that only for higher motion models – here, the perspective model – RANSAC-based estimation is slightly more exact than the M-Estimator. Since the overall result of feature-based methods do not fulfill our needs, especially if moving foreground objects are present, we additionally have to refine the registration results by other methods. Thus, for sprite generation it is not very important which robust estimation method is used.

3.1.2 Photometric Consistency-based Approaches

Photometric consistency-based approaches use different energy terms to be minimized than feature-based approaches. In this category of registration methods the cross-correlation of pixel values for certain color channels is being maximized with respect to the transformation

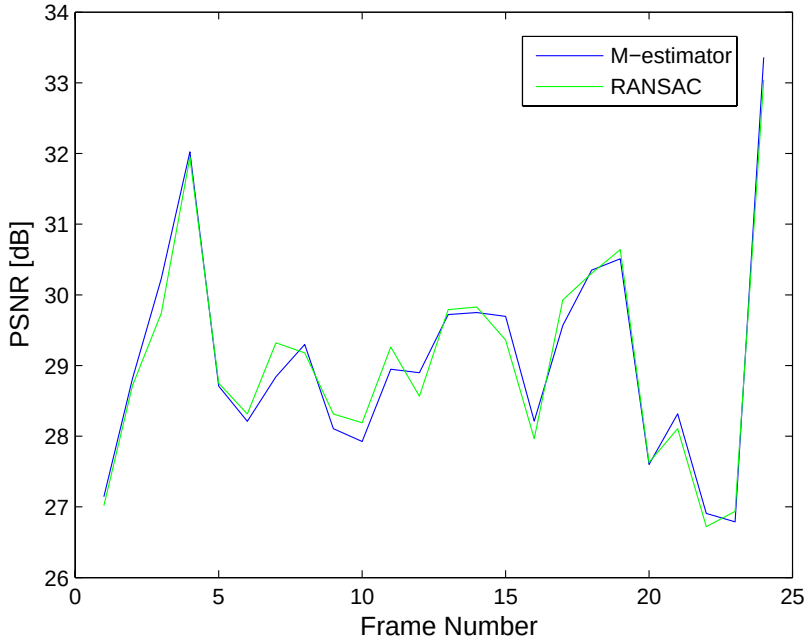


Figure 3.9: Comparison of feature-based short-term registration quality using M-estimator ($q = 3$), RANSAC, and the affine motion model for sequence "Stefan".

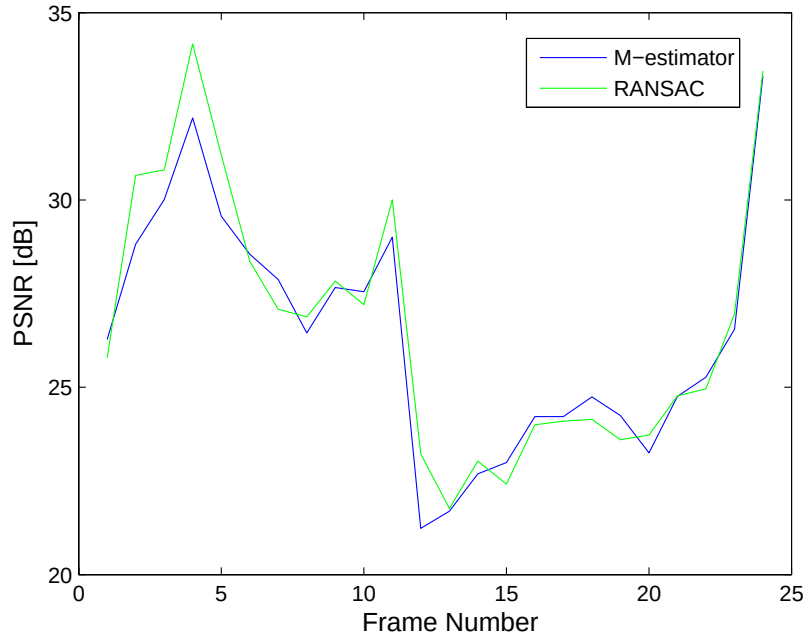


Figure 3.10: Comparison of feature-based short-term registration quality using M-estimator ($q = 3$), RANSAC, and the perspective motion model for sequence "Stefan".

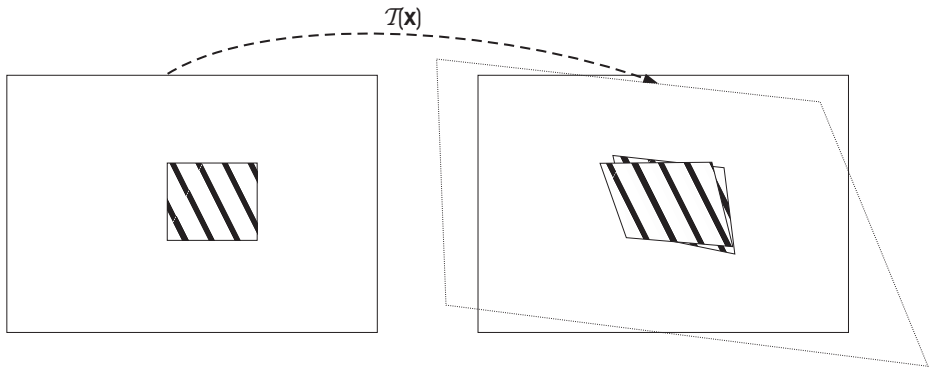


Figure 3.11: Principle of photometric consistency-based registration: Due to tuning of the transformation parameters, the correlation between the second image and the compensated (transformed) first image is maximized.

parameter vector $\mathbf{k}$. Generally these methods are not sparse, i.e., all overlapping regions are used for registration. As measure for the normalized cross correlation the compensated pixel difference is used. Figure 3.11 shows the principle of this optimization. The transformation is optimal when the warped adjacent image exactly overlaps with its neighbor. As measure for the normalized cross-correlation between two images the absolute or squared image differences are often used since they are easy to compute and can be treated very effectively in terms of energy minimization. A typical energy or cost function E that has to be minimized with respect to the transformation parameters is

$$\begin{aligned} E(\mathbf{k}) &= \frac{1}{2}\frac{1}{N_R}\sum_{(x,y)\in R}\big(I(x,y) - I'(x'(x,y),y'(x,y))\big)^2 \\ &= \frac{1}{2}\frac{1}{N_R}\sum_{(x,y)\in R}\big(I(x,y) - I'(\mathcal{T}(\mathbf{k};x,y))\big)^2, \end{aligned} \tag{3.14}$$

where R is the region of overlap for the frame I and the frame I' to register. Notice, that usually every frame consists of three color channels. The energy term is normalized on the number of overlapping pixels N_R. It can be useful to define the energy term as sum of squared differences for all three channels. Since the channel information is in general very redundant, we concentrate only on the luminance-based values, which are defined for the most important color spaces as YC_bC_r, HSV, or $L^*a^*b^*$. Frame I' is dependent on the image coordinates x' and y', which are transformed from x and y via transformation $\mathcal{T}$ with

$$(x', y')^T = \mathcal{T}(\mathbf{k}; x, y)^T. \tag{3.15}$$

The minimization can be achieved using gradient descent methods. The goal of these methods is to find a local minimum of E. This is usually achieved if

$$\begin{aligned}\mathbf{0} &= \nabla_{\mathbf{k}} E(\mathbf{k}) \\ &= \frac{\partial E}{\partial \mathbf{k}} \\ &= \left(\frac{\partial E}{\partial k_1}, \ldots, \frac{\partial E}{\partial k_m}\right)^T \\ &= -\frac{1}{N_R} \sum_{(x,y)\in R} \left[I(x,y) - I'(\mathcal{T}(\mathbf{k}; x, y))\right] \frac{\partial I'(\mathcal{T}(\mathbf{k}; x, y))}{\partial \mathbf{k}}. \end{aligned} \tag{3.16}$$

In the above equation, $\nabla_{\mathbf{k}}$ represents the gradient with respect to the transformation parameter vector $\mathbf{k} = (k_1, \ldots, k_m)^T$ having m entries depending on the image transformation model.

3.1.2.1 Newton-Raphson Approach for Energy Minimization

The Newton-Raphson method is a local convergent method to find the null of any continuously differentiable mapping $f(\mathbf{k}) : \mathbb{R}^m \to \mathbb{R}^m$. The advantage above other gradient descent methods is that the step size for following the gradient of the mapping is determined analytically. Starting at any point $\mathbf{k}_0$ we can iteratively find a null of f using

$$\mathbf{k}_{n+1} = \mathbf{k}_n - J(\mathbf{k}_n)^{-1} f(\mathbf{k}_n). \tag{3.17}$$

$J(\mathbf{k})$ is the quadratic Jacobian matrix with $J_{i,j} = \partial f_i / \partial k_j$. If we set f equal to the gradient of the energy term 3.14 we obtain

$$J(\mathbf{k}_n) = H(\mathbf{k}_n) = \begin{bmatrix} \frac{\partial^2 E}{\partial k_1 \partial k_1} & \frac{\partial^2 E}{\partial k_1 \partial k_2} & \cdots & \frac{\partial^2 E}{\partial k_1 \partial k_m} \\ \frac{\partial^2 E}{\partial k_2 \partial k_1} & \frac{\partial^2 E}{\partial k_2 \partial k_2} & \cdots & \frac{\partial^2 E}{\partial k_2 \partial k_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial k_m \partial k_1} & \frac{\partial^2 E}{\partial k_m \partial k_2} & \cdots & \frac{\partial^2 E}{\partial k_m \partial k_m} \end{bmatrix}, \tag{3.18}$$

which is the Hessian matrix of E with respect to the transformation parameters. The entries of the Hessian are defined as follows:

$$\begin{aligned} H_{ij} &= \frac{\partial E}{\partial k_i \partial k_j} \\ &= \frac{\partial}{\partial k_j} \left(-\frac{1}{N_R} \sum_{(x,y)\in R} \left[I(x,y) - I'(x',y')\right] \frac{\partial I'(x',y')}{\partial k_i} \right) \\ &= -\frac{1}{N_R} \sum_{(x,y)\in R} \frac{-\partial I'(x',y')}{\partial k_j} \frac{\partial I'(x',y')}{\partial k_i} - \frac{1}{N_R} \underbrace{\sum_{(x,y)\in R} \left[I(x,y) - I'(x',y')\right] \frac{\partial^2 I'(x',y')}{\partial k_i \partial k_j}}_{\approx 0} \\ &= \frac{1}{N_R} \sum_{(x,y)\in R} \frac{\partial I'(x',y')}{\partial k_i} \frac{\partial I'(x',y')}{\partial k_j} \end{aligned} \tag{3.19}$$

The difference term $[I(x,y) - I'(x',y')]$ can approximately be regarded as zero mean symmetrical noise and thus the impact of the second derivatives of $I'(x',y')$ can be neglected. In order to compute the first derivatives we use the chain rule

$$\frac{\partial I'(x',y')}{\partial k_i} = \frac{\partial I'(x',y')}{\partial x'}\frac{\partial x'}{\partial k_i} + \frac{\partial I'(x',y')}{\partial y'}\frac{\partial y'}{\partial k_i} \tag{3.20}$$

with $\frac{\partial I'(x',y')}{\partial x'}$ and $\frac{\partial I'(x',y')}{\partial y'}$ being the first derivatives of frame I' in horizontal and vertical direction. They can be determined numerically by filtering I' using the filter kernels of the *Sobel operator* or using a combination of Gaussian filter and differential filters as proposed in [128]. Even simple Euler-backward differential filter ($[-1\ 1]$ and $[-1\ 1]^T$) may be used. In our experiments we did not find a significant difference in the registration result. The partial derivatives of the image coordinates x' and y' with respect to the transformation parameters are dependent on the specific transformation model and can be found in A.5. Finally the rule for iteratively finding the minimum of E becomes

$$\mathbf{k}_{n+1} = \mathbf{k}_n - H(\mathbf{k}_n)^{-1}\nabla_{\mathbf{k}}E(\mathbf{k}_n). \tag{3.21}$$

3.1.2.2 Levenberg-Marquardt for Energy Minimization

If the inverse of the Hessian in Equation 3.21 equals a constant $\lambda \cdot I$, where I is the identity matrix, the Newton-Raphson step would be equivalent to a real gradient descent

$$\mathbf{k}_{n+1} = \mathbf{k}_n - \lambda\nabla_{\mathbf{k}}E,$$

with λ as descent step size. Unfortunately, due to influences of noise and objects that do not fit the transformation model, e.g., foreground objects, the directions of the Newton-Raphson step $H(\mathbf{k}_n)^{-1}\nabla_{\mathbf{k}}E$ and the gradient step $\lambda\nabla_{\mathbf{k}}E$ may differ significantly. Thus, it is not sure that with every further iteration the error term $E(\mathbf{k})$ will decrease. Therefore, in cases of an increasing cost it is necessary to gradually correct the step direction. The Levenberg-Marquardt method does this correction by introducing a parameter τ into the iteration computation. Equation 3.21 becomes

$$\mathbf{k}_{n+1} = \mathbf{k}_n - \left(H + \tau I\right)^{-1}\nabla_{\mathbf{k}}E. \tag{3.22}$$

Thus, in an inner iteration the step τ is varied from the Newton-Raphson direction ($\tau \to 0$) towards the gradient direction with decreasing step size $1/\tau$ ($\tau \to \infty$). First we set τ on a small value like

$$\tau_0 = 0.001 \cdot \max_{\forall i,j} |H_{ij}|\,. \tag{3.23}$$

If $E(\mathbf{k}_{n+1}) > E(\mathbf{k}_n)$ the freshly computed $\mathbf{k}_{n+1}$ is disapproved. τ then is increased by $\tau_{n+1} = 10\tau_n$ and Equation 3.22 is computed again. This procedure is repeated till $E(\mathbf{k}_{n+1})$ decreases or the general terminating condition is valid.

We found out that especially for scene shots with huge portions of foreground objects the Levenberg-Marquardt algorithm improves the registration exactness enormously. It is clear that due to the inner iteration over varying parameter τ the computational load is also higher. Since the components H_{ij} of the Hessian and the gradient $\nabla_{\mathbf{k}} E$ have to be determined only once, the difference in complexity is not very remarkable.

3.1.2.3 Robustness Aspects

Photometric consistency-based methods for 2D image registration densely try to minimize the overlapping pixel difference. Following Equation 3.20 only pixels with a significant gradient in horizontal and vertical direction have influence to the error minimization. Hence, textural edges and corners in the images have a very high impact. Additionally, both nonlinear minimization approaches, Newton-Raphson and Levenberg-Marquardt, can only find local minima of any designed error functionality. Since we are looking for the global minimum, two problems have to be tackled in order to obtain a satisfying solution:

- finding a good starting point $\mathbf{k}_0$ for the nonlinear gradient descent method and
- suppressing the influence of object edges and object texture edges, that move independently from the camera motion.

The former problem is solved by using a hierarchical approach, starting with a more robust low order 2D motion model and using the result as initial guess for the next higher motion model. The lowest level of hierarchy is hereby computed as robust feature-based estimation (see Section 3.1.1). We will present the final hierarchical concept in Section 3.2.

The latter problem is solved by using a M-estimator [131] as already introduced for robust linear feature-based registration approaches. The energy term now is weighted for every pixel pair the two images I and I' overlap. Thus, we can write

$$E(\mathbf{k}) = \frac{1}{2}\frac{1}{N_R}\sum_{(x,y)\in R} w(x,y)\left(I(x,y) - I'(x',y')\right)^2, \tag{3.24}$$

where $w(x,y)$ is a weighting function using Tukey's biweight weighting function or the binary weighting function. Using the binary weighting function

$$w(x,y) = \begin{cases} 1 & \text{for} \quad [I(x,y) - I'(x',y')]^2 \leq q\mu \\ 0 & \text{for} \quad [I(x,y) - I'(x',y')]^2 > q\mu \end{cases} \tag{3.25}$$

can save a lot of computational load. The value μ is estimated as

$$\mu = \frac{1}{N_R}\sum_{(x,y)\in R} [I(x,y) - I'(x',y')]^2. \tag{3.26}$$

Assuming $[I(x,y) - I'(x',y')]$ to be Gaussian distributed with zero mean and codimension one for all inliers, e.g., all pixels that support the global camera motion, then its square

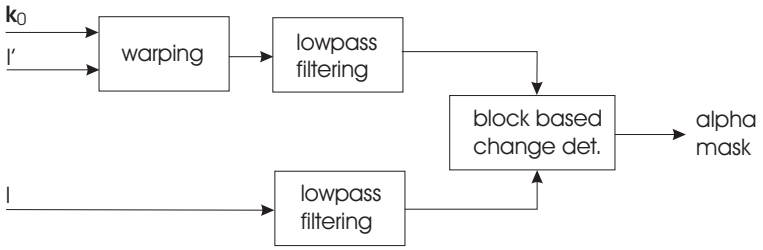


Figure 3.12: Flowchart of coarse previous alpha mask generation for robust gradient descent methods.

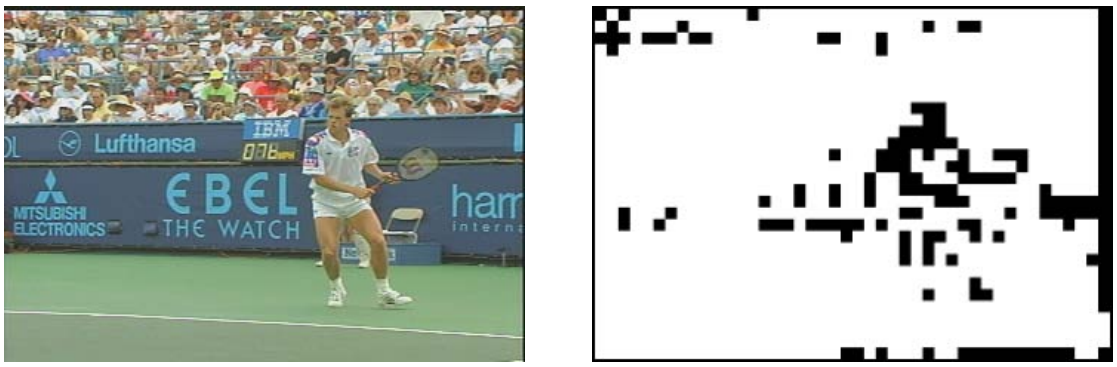


Figure 3.13: Example of coarse binary alpha mask prior to photometric consistency-based nonlinear minimization of error function E: Frame 2 of original sequence "Stefan" (left); binary alpha mask for the same frame - black blocks are not used for gradient calculation

$[I(x,y) - I'(x',y')]^2$ is χ_1^2 distributed. Therefore, 95% of all inlier values lie within the interval $[0, 3.84\mu]$. It is then reasonable to set tuning constant q between 3 and 4.

In order to have only small influences of the outliers belonging to any foreground object we previously calculate an alpha mask masking out regions that do not fit the initial guess of the transformation model. It is based on the block differences between frame I and the motion compensated adjacent frame I'. Both images are subdivided in blocks of size 8×8. Then, for each block the sum of absolute differences (SAD) is computed. A block $B_{I'}(m,n)$ with m being the horizontal and n being the vertical block indices of image I' is only used for gradient-based energy minimization if

$$\sum_{x_B=0}^{7} \sum_{y_B=0}^{7} |I(8m+x_B, 8n+y_B) - I'(x'(8m+x_B, 8n+y_B), y'(8m+x_B, 8n+y_B))| < T. \quad (3.27)$$

Threshold T is set empirically and is in an area between 10 and 15. In Figure 3.12 the flowchart of the coarse binary alpha mask generation algorithm is shown. As example we show the mask for frame two of sequence "Stefan" (Figure 3.13). As one can see, big portions of the foreground object's edges are masked out. Thus, the estimation of μ in Equation 3.26 is much closer to the value obtained by using inliers only.

3.2 Short-term Registration (Frame-to-frame)

The generation of background mosaics or sprites requires an exact registration of all frames of a scene shot with respect to a certain reference image. Since in any of two frames of the shot the image content does not necessarily overlap, a direct registration of frame I_n ($n = 0, \ldots, N-1$) into the coordinate system of frame I_r (subscript r stands for the reference frame) cannot be estimated directly. Moreover, for several applications it is even desirable to register two uncorrelated images into one sprite. In order to achieve such a registration the short-term parameters, i.e., the transformation between two adjacent frames I_n and I_{n+1} are estimated in a first step. This short-term registration is also known as *global motion estimation* (GME) from coding approaches [18], [66]. The utilized algorithm for short-term (frame-to-frame) registration is presented in the following sections.

As we have shown in Section 3.1 nonlinear error minimization methods promise a more exact minimization while feature-based linear solutions yield good initial parameter sets due to low complex robust estimators. On the other hand, transformation models with a lower degree of freedom are more robust against outliers since the error surface to find the minimum in has fewer dimensions. But, in contrast to that fact, higher order image transformations better represent real physics. Therefore, we follow a hierarchical approach for short-term global motion estimation in a double sense: from linear feature-based estimation to nonlinear photometric consistency-based approaches and from low dimensional image transformations to image transformations with a higher number of transformation parameters.

3.2.1 The Hierarchical Approach

In Figure 3.14 the flowchart of our short-term registration approach is shown. In the first part the initial parameter set for gradient-based energy minimization is estimated. This is done by linear feature-based parameter estimation using a robust M-estimator (Section 3.1.1.1). For the sake of robustness, only the two parameter translational motion model is estimated.

Subsequently, the Levenberg-Marquardt energy minimization (Section 3.1.2.2) approach is utilized to estimate the transformation parameters of higher order models. This is achieved by hierarchical passing the parameters of the lower order model as first guess $\mathbf{k}^0$ for the actual model. Previously to the first gradient descent method a preliminary binary alpha minimizing the influence of foreground object boarders is computed (Section 3.1.2.3).

3.2.1.1 A New Termination Condition for Nonlinear Minimization

In Section 3.1.2 we introduced the prevalent methods for nonlinear gradient descent methods by minimizing a defined error function $E(\mathbf{k})$. In classical applications this error term

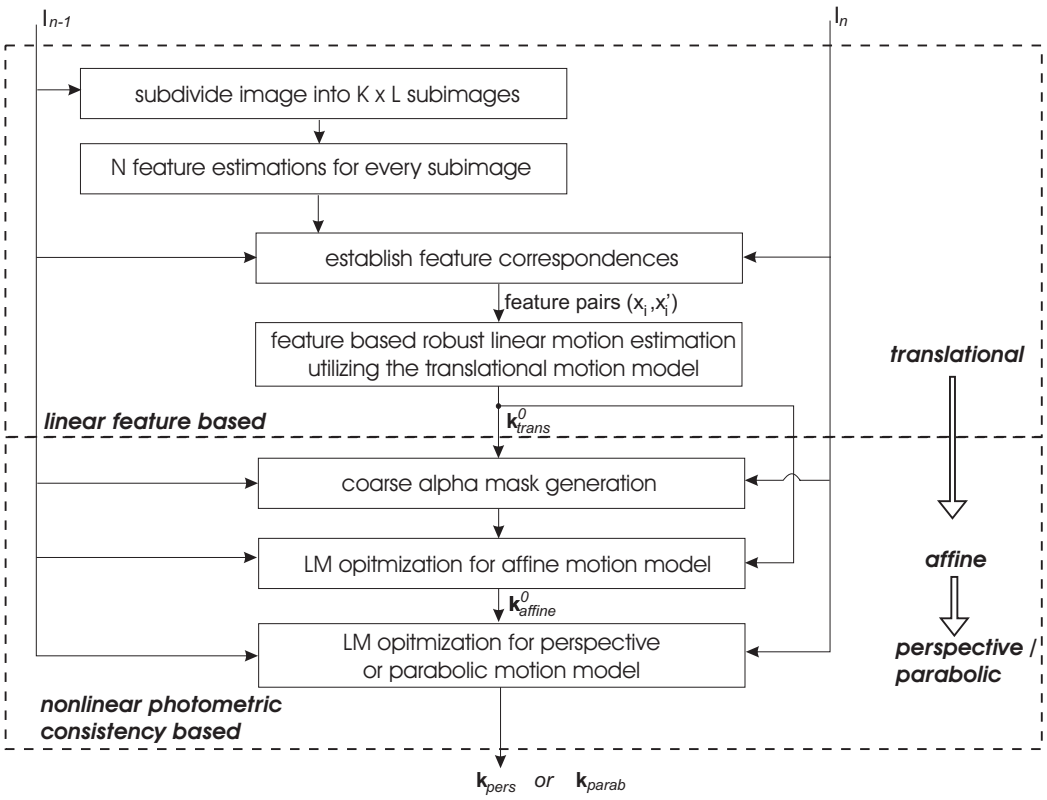


Figure 3.14: Flowchart of the hierarchical short-time registration approach.

is used to define the gradient descent *termination condition*. Normally, the iteration of Equation 3.21 or 3.22 respectively is stopped if the error term

$$|E(\mathbf{k^{n+1}}) - E(\mathbf{k^n})| < \varepsilon. \tag{3.28}$$

In case of the Newton-Raphson method the computation can also be stopped if the error does not decrease any longer. The threshold ε is set empirically to a small value, e.g., $\varepsilon = 10^{-4}$.

Unfortunately, this termination condition is highly dependent on the content of the analyzed sequence. Thus, for a highly active sequence with large portions of foreground objects, fast camera motion, or huge changes in lightening conditions the termination threshold ε may be too high in order to determine the optimal transformation. For scenes with very few changes between the frames, however, the threshold may not be high enough and we will compute more iterations than necessary. We introduce a new content independent determination condition, which is only dependent on the sampling positions of the transformed pixels. Assuming we have computed the $(i+1)th$ iteration of Levenberg-Marquardt transformation estimation $\mathbf{k}^{i+1}$. Then we calculate a new grid displacement error e_g

$$e_g^i(x,y) = d_g^i(x,y) = \left\| \mathcal{T}(\mathbf{k}^{i+1}; x, y) - \mathcal{T}(\mathbf{k}^i; x, y) \right\|, \tag{3.29}$$

which is the L_2 norm of the difference vector of a transformed pixel coordinate before and after the last iteration $(i+1)$. Then, the determination condition is derived from this error

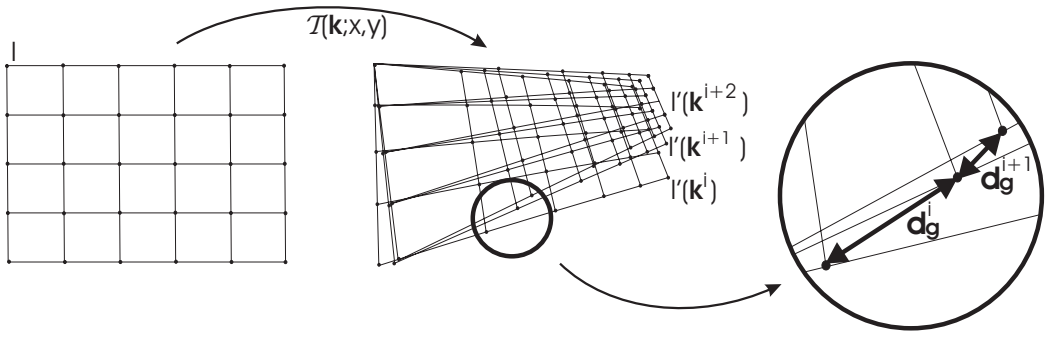


Figure 3.15: Schematical illustration of the proposed termination condition for nonlinear transformation computation:

as

$$\max_{\forall(x,y)} e_g^i(x,y) = \max_{\forall(x,y)} \left\| \mathcal{T}(\mathbf{k}^{i+1};x,y) - \mathcal{T}(\mathbf{k}^i;x,y) \right\| < \varepsilon_g. \tag{3.30}$$

Hence, the iteration will be stopped if the maximum grid displacement for any transformed pixel of image I into image I' is smaller than a given threshold ε_g. For very exact computation this threshold is set to $\varepsilon_g = 0.01$ pixels. Figure 3.15 shows schematically the geometrical fundamentals for the proposed termination condition. We found out, that this termination condition leads to more exact registration results for certain images.

3.2.2 Image Interpolation and Aliasing

In the sections above we often referred to the term $I'(x',y') = I'(\mathcal{T}(\mathbf{k};x,y))$, where the pixel coordinate $x'(x,y) = \mathcal{T}_x(\mathbf{k};x,y)$ and $y'(x,y) = \mathcal{T}_y(\mathbf{k};x,y)$. In general, coordinates x' and y' will not be integer values. Therefore, in order to compute the pixel value of I' at this specific position one has to perform an image interpolation. If we assume the original recorded images being free of spatial aliasing – which of course is not the case – then we can still obtain aliasing problems due to non-ideal interpolation methods. This can significantly influence the registration result. Many authors ([129], [123]) do not address this specific problem and use bilinear or bicubic interpolation methods. Others ([66], [165]) perform the registration in the upsampled domain, where a high order upsampling filter – Daubechie's 7-tap biorthogonal wavelet synthesis filter is used – in combination with simple bilinear interpolation yields very accurate results. Also the computational complexity is kept small due to only one upsampling process per image.

In this work we draw upon the work of P. Thévenaz et al. [146], who intensively studied the use of B-spline interpolation. B-spline interpolation belongs to the class of generalized interpolation where the function is not directly interpolated from the discrete samples, but from some prior estimated supporting coefficients. Since the supporting coefficients for spline interpolation are independent from the actual coordinate to interpolate, one can compute these coefficients in advance. Thus even B-spline interpolation up to $9th$ order does not significantly increase the computational load. B-spline interpolation can be regarded as

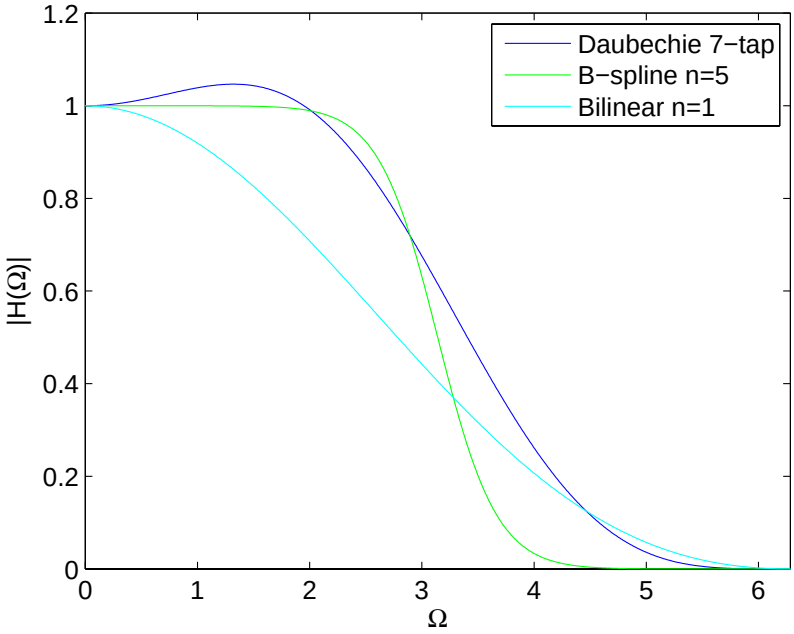


Figure 3.16: Lowpass filter frequency responses for commonly used interpolation filters: bilinear, Daubechie's 7-tap wavelet synthesis filter, and B-spline interpolation filter of 5th order.

convolution of a locally defined spline kernel with the discrete coefficients $c_n(kT)$. In one dimension it yields

$$\hat{s}_n(t) = c_n(kT) * \beta^n. \tag{3.31}$$

Coefficient β^n is the spline kernel of nth order and $c_n(kt)$ are the supporting coefficients. It is defined recursively by

$$\beta^n = \beta^0 * \beta^{n-1} \quad \text{with} \quad \beta^0 = \Pi_T(t). \tag{3.32}$$

Using B-spline interpolation of higher order one obtains a low complex approximation of the ideal interpolation function $(\sin t)/t$. Performing frequency domain analysis of B-splines is nontrivial since β^n is only the approximation function. In order to estimate the frequency response of the whole interpolation process, the approximation using β^n and the supporting coefficient estimation have to be taken into account. Mihajlovic et al. [87] performed this analysis for B-splines up to order $n = 5$. The frequency response becomes:

$$H_{\beta,5}(\omega) = \frac{60\text{sinc}^5(\frac{\omega}{2\pi})}{33 + 26\cos(\omega) + \cos(2\omega)} \tag{3.33}$$

In Figure 3.16 the frequency responses of usually utilized interpolation filters are shown. Even with order $n = 5$ the B-spline interpolation exceeds other methods in terms of accuracy and smoothness in the pass-band. Note that the bilinear interpolation is equivalent with B-spline interpolation of order $n = 1$.

Transformation model	bPSNR with mask	bPSNR without mask
affine	27.74	27.71
perspective	30.65	30.49
parabolic	30.39	30.38
perspective in [131]	28.58	28.50

Table 3.1: Average background PSNR in dB for short-term motion compensated images of sequence "Stefan" (SIF, 300 frames).

3.2.3 Experimental Results

We conducted experiments for several transformation models and measured the background prediction quality between adjacent frames. This was achieved by warping every image of a sequence into the coordinate system of the neighbor using the estimated transformation parameter. The quality is assessed by computing the luminance background PSNR (*peak signal to noise ratio*) for only the rigid background of a scene. In order to do so, an already generated binary foreground mask is used. The background PSNR is calculated as follows:

$$bPSNR = 10 \log \frac{255^2}{\frac{1}{N_R} \sum_{(x,y)\in R} (I_n(x,y) - I_{n+1}(x',y'))^2 \cdot \alpha_n(x,y) \cdot \alpha_{n+1}(x',y')} \tag{3.34}$$

The value of term $I_{n+1}(x', y')$ is interpolated using B-splines of order 9. The SSD is masked out by the binary alpha mask α_n of the nth frame and the warped binary alpha mask α_{n+1} of the $n+1$th frame.

For sequence "Stefan" the short-term background PSNR was measured over 300 frames. For comparison we also computed the short-term parameters of the sequence while masking out the foreground object before the motion estimation. Hence, less outliers should have impact to the estimation process and the registration result should increase. The difference between motion parameter estimation with and without masked foreground objects is an indicator for the robustness of our approach. In Table 3.1 we give the average background PSNR for affine, perspective and parabolic transformation. Additionally, the values from the approach in [131] are given. As one can see, our hierarchical short-term estimation for perspective image transformation yields the best registration results, even 0.2 dB higher than the parabolic motion model. We will later show that for long-term estimation the parabolic model will improve the background re-registration. Compared to the approach proposed by Smolic et al. we exceed the perspective background PSNR by almost 2 dB for the unsegmented test sequence.

Figures 3.17 and 3.18 depict the registration quality assessment over the whole sequence. The motion models of higher degree, perspective and parabolic, meet more ore less the same registration results, independent of the foreground masking process. The affine motion model always yields worse results up to 4.5 dB for certain frames. In Figure 3.19 we compare the results of our registration method with the results of [131] on a frame-by-frame basis. It is noticeable that the curve differences are not that big in the peaks, when only slow

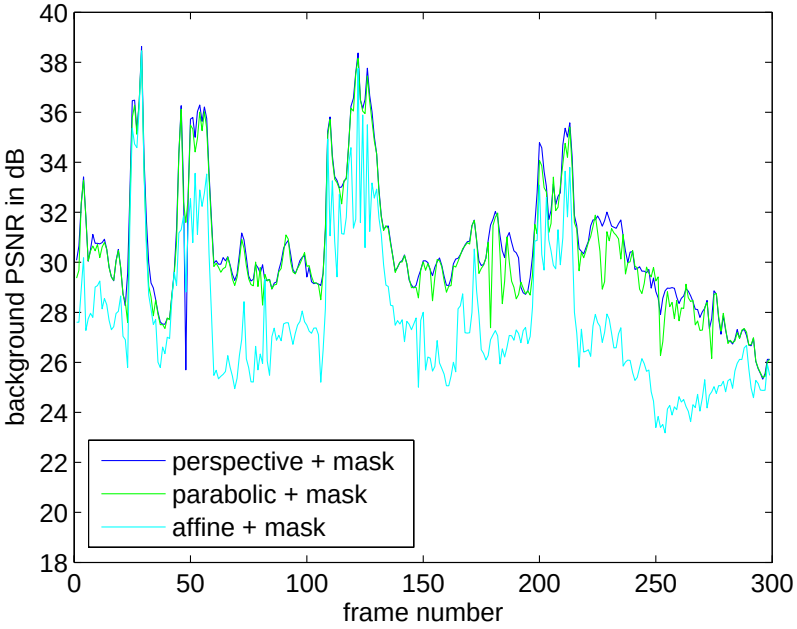


Figure 3.17: Comparison of registration results using the background PSNR for Sequence "Stefan" while masking out the foreground objects.

camera motion occurs. On the other hand, in scene parts of low registration quality, e.g., fast camera motions, the quality gain is up to 4 dB.

3.3 Long-term Registration (Frame-to-mosaic)

The construction of planar background sprites requires the registration of a multitude of frames of one shot into the coordinate system of one image, the final sprite. There exist several ways to obtain this registration based on previously estimated short-term registration parameters $\mathbf{k}_{n-1,n}$. The simplest approach would be the concatenation of those parameters. The problems that arise from this *close loop* approach will be addressed in the next section.

3.3.1 Short-term Concatenation and Error Propagation

Concatenating 2D transformations obtained by global motion estimation (GME) implies two fundamental problems to be tackled. The first one is only valid for nonlinear transformation models. Assuming we estimated two nonlinear transformations of the same type $\mathcal{T}_{n-1,n}$ and $\mathcal{T}_{n,n+1}$ for global camera motion between frames I_{n-1}, I_n, and I_{n+1}. Then the overall transformation obtained by chaining both transformations $\mathcal{T}_{n-1,n+1} = \mathcal{T}_{n,n+1}(\mathcal{T}_{n-1,n})$ would generally not be the same type of transformation anymore. Thus, the order of transformation for the registration of any frame I_i into the plane of the reference frame I_r would be dependent on the temporal distance $|i - r|$ between those frames. That means that the size of parameter vector $\mathbf{k}_{r,i}$ is not constant any longer. For applications that require the transformation parameters in order to re-project the sprite content, e.g., object segmentation or

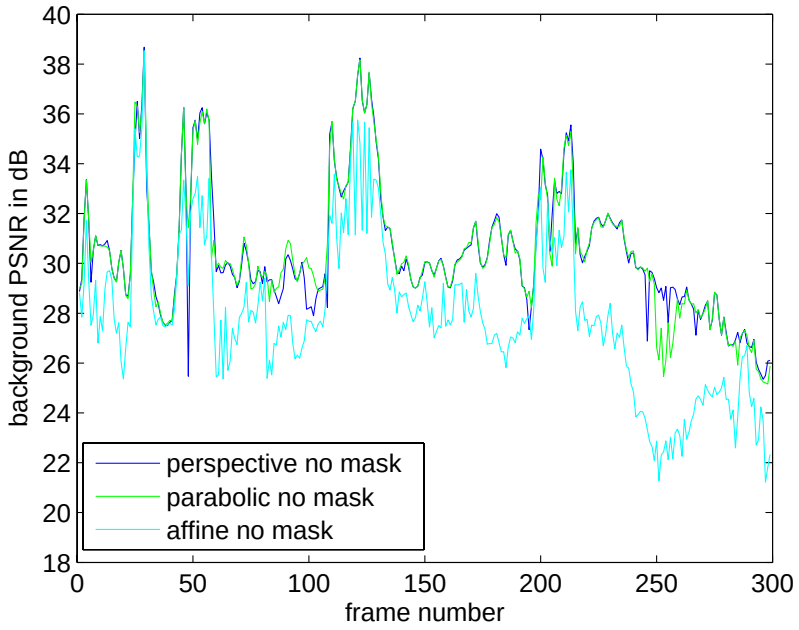


Figure 3.18: Comparison of registration results using the background PSNR for Sequence "Stefan" without masking out of foreground objects.

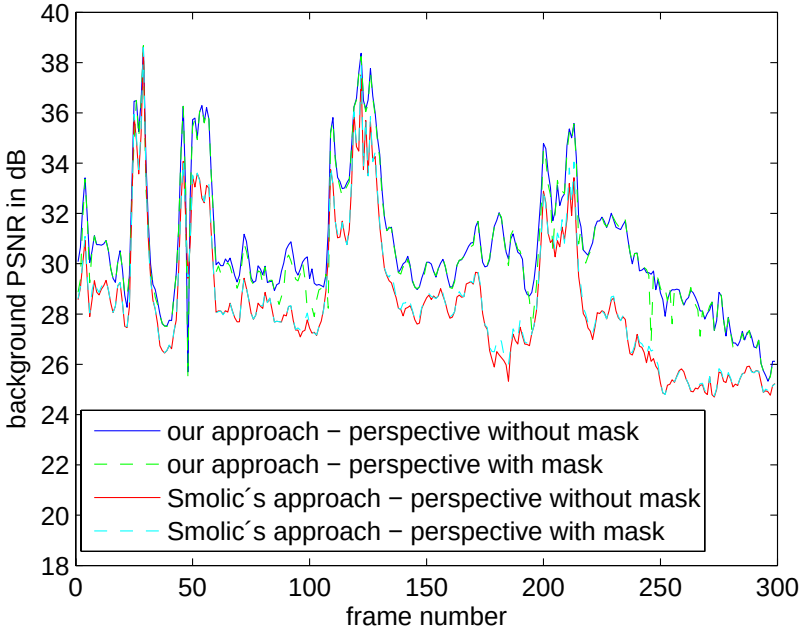


Figure 3.19: Comparison of registration results using the background PSNR for Sequence "Stefan" for our short-term registration approach and the one presented in [131].

sprite coding, this property is not desirable. Therefore, in case of nonlinear basis motion models the long-term frame-to-mosaic registration has to be computed in a different way.

The second problem, which is even more important, is the way of error propagation. Due to image noise and natural derivations from the adopted camera motion model, registration will always be erroneous. If we concatenate the transformations over the sequence the error will grow due to error propagation rules. Assuming the short-term transformation parameters $\mathbf{k}_{n-1,n}$ are already estimated and the estimation error to be independent for each dimension of the parameter vector and each estimation between image I_{n-1} and I_n with $n = 2, \ldots, N$. In each short-term transformation the transformation result x_n is linearly dependent of each parameter $\mathbf{k}_{n-1,n} \pm \Delta\mathbf{k}_{n-1,n}$, where $[\mathbf{k}_{n-1,n} - \Delta\mathbf{k}_{n-1,n}, \mathbf{k}_{n-1,n} + \Delta\mathbf{k}_{n-1,n}]$ is the error interval, e.g., the 2σ neighborhood of the distribution of $\mathbf{k}_{n-1,n}$. In first approximation we can assume the absolute error $\Delta\mathbf{k}$ to be constant for all adjacent image pairs. It can be shown that for a certain number of transformations during concatenation the overall absolute error $\Delta\mathbf{k}_{n-a,n}$ will be bigger than $\Delta\mathbf{k}$ since the the overall transformation parameters is always a sum of products of the single parameters $k^i_{n-1,n}$.

$$\Delta\mathbf{k}_{n-a,n} > \Delta\mathbf{k} \quad \text{for} \quad a > a_\epsilon. \tag{3.35}$$

Therefore, it is essential for a good long-term registration approach to correct this error accumulation. In some papers it was proposed to do the short-term registration between image I_n and an estimated image $\hat{I}_{n-1}$, obtained by re-registration of the last registered image I_n in the sprite S_{n-1} (closed loop approach). We follow a different approach, where the actual image I_n is directly registered into the sprite image S_n.

3.3.2 Direct Frame-to-Sprite Registration

The principle of direct frame-to-sprite registration is depicted in Figure 3.20. The algorithm is performed in an iterative order. Starting from the reference frame I_r, which can be chosen arbitrarily the frames of one shot are registered in order of temporal distance from the reference frame. Of course, this works in both directions of time. Without loss of generality we only address the case of positive time direction. Let's suppose frame $n-1$ is already registered represented by the transformation parameter set $\mathbf{k}_{r,n-1}$ we first perform a short-term frame-to-frame registration following the hierarchical approach of Section 3.2.1, which yields in short-term parameter set $\mathbf{k}_{n-1,n}$. By combining transformation $\mathcal{T}_{r,n-1}$ with transformation $\mathcal{T}_{n-1,n}$ we obtain an initial guess for the long-term parameter set $\mathbf{k}^0_{r,n}$. In parallel, we build up an preliminary sprite $\tilde{S}_{n-1}$ that saves all newly discovered image content. Since rigid background objects and moving foreground objects are not classified, it is possible that artifacts of the foreground objects are observable. Figure 3.21 shows a preliminary sprite over 450 frames of sequence "Charlottenburg".

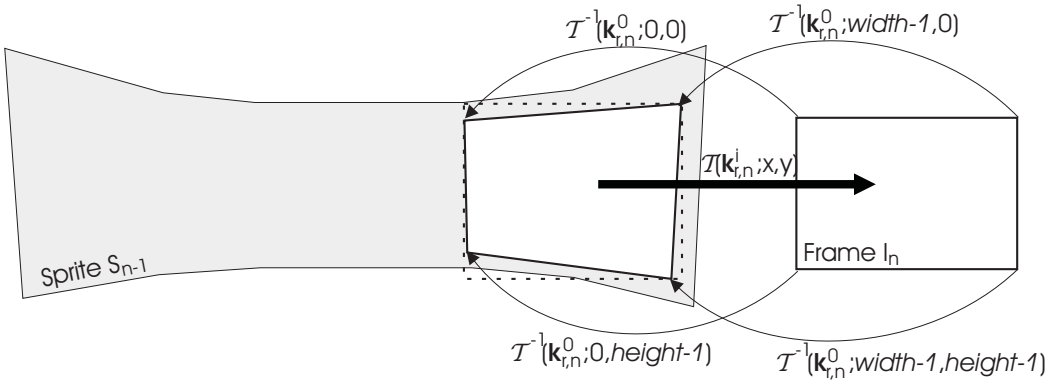


Figure 3.20: Direct frame-to-sprite registration: Due to inversion of the concatenated image transformation, the corners of the warped actual frame I_n in sprite S_{n-1} can be estimated. The surrounding rectangle (dashed box) is then used as image for performing gradient-based direct registration. It shall be understood that this procedure is necessary to roughly estimate the region in the sprite where the image content of frame I_n can be found. Of course, the gradient descent is only performed on pixels that lie within the frame I_n after transformation. Thus the number of pixels that have impact to the registration may vary over the iteration process.

In order to eventually estimate the long-term parameter $\mathbf{k}_{r,n}$ using the Levenberg- Marquardt photometric energy minimization method (Section 3.1.2.2) with minimal computational load, we have to find the region in sprite $\hat{S}_{n-1}$ where actual image I_n will fit in. This is achieved by inverse transforming the corners of I_n and utilizing the surrounding rectangle. The rectangle corner points can be computed by

$$(x_r^c, y_r^c)^T = \mathcal{T}^{-1}(\mathbf{k}^0_{r,n}; x_n^c, y_n^c)^T, \tag{3.36}$$

where $(x_r^c, y_r^c)^T$ $(c = 1, \ldots, 4)$ are the images of the corners $(x_n^c, y_n^c)^T$ of frame I_n. The whole method for long-term registration is shown in the flowchart of Figure 3.22.

3.3.3 Nonlinear Transformation Models

In the previous section we utilized transformation concatenation or transformation inversion at certain points of the registration method. For linear motion models like translative, affine, or perspective this can be done in a closed mathematical form. However, for the nonlinear parabolic model combination and inversion have to be estimated numerically.

Parabolic Motion Concatenation

Knowing the short-term transformation parameters $\mathbf{k}_{i-1,i} = (a_{0;i-1,i}, \ldots, b_{5;i-1,i})$ with $i = a+1, \ldots, b$ for the parabolic model, we can compute the image of a subset of pixels $\mathbf{x}_a$ of frame I_a in frame I_b by iteratively calculating

$$\mathbf{x}_i = \mathcal{T}(\mathbf{k}_{i-1,i}; \mathbf{x}_{i-1}) \quad \text{for} \quad i = a+1, \ldots, b. \tag{3.37}$$

Figure 3.21: Preliminary sprite of sequence "Charlottenburg" (360x288, 450 frames) with frame 0 as reference. Artifacts that arise from pasting parts of the foreground objects are marked with white circles.

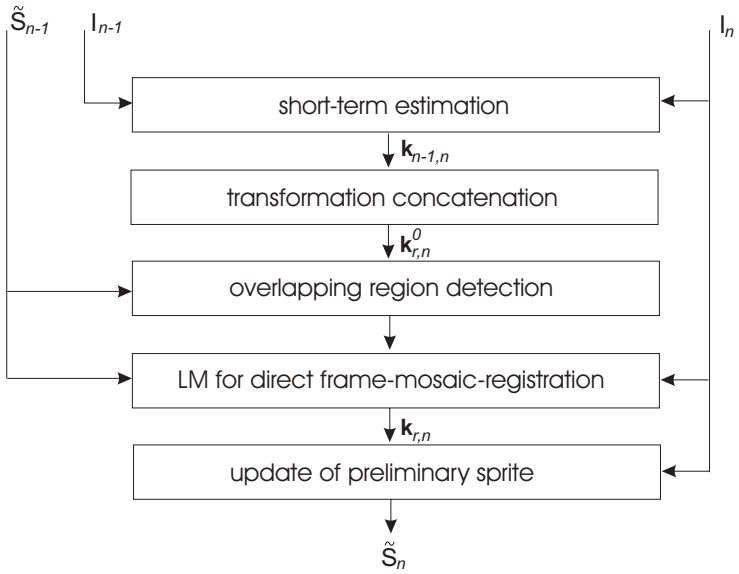


Figure 3.22: Flowchart of the final long-term parameter estimation for sprite generation.

For this subset we obtain the point correspondences ($\mathbf{x}_a \leftrightarrow \mathbf{x}_b$) between both images I_a and I_b. Because of being free of outliers, we can directly estimate the overall parabolic transformation following the linear feature-based estimation using Equation 3.6. It is clear that the exact transformation between the two images would be of order $2(b-a)$ and thus, the estimated transformation of order 2 does not necessarily be the optimal solution. In praxis we found out that as a initial parameters set for further refinement this estimation is absolute satisfying.

Parabolic Motion Inversion

In contrary to the concatenation case the estimation of the inverse transformation has to be very exact. Since one of our goals is to re-project the sprite content into all registered

frame planes of a shot, we want to introduce no additional error. Starting point here is the transformation vector $\mathbf{k}_{par} = (a_0, \ldots, a_5, b_0, \ldots, b_5)^T$ with

$$\mathbf{x}' = \mathcal{T}(\mathbf{k}_{par}; \mathbf{x}).$$

Motion inversion is defined by the following problem: For every pixel coordinate $\mathbf{x}' = (x', y')^T$ in image I' find the estimation $\hat{\mathbf{x}} = (\hat{x}, \hat{y})^T$ in the coordinate system of the corresponding image I, for which holds

$$f(\hat{\mathbf{x}}) = \mathbf{x}' - \mathcal{T}(\mathbf{k}_{par}; \hat{\mathbf{x}}) = 0. \tag{3.38}$$

Equation 3.38 can be solved by applying again Newton-Raphson for functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$. Thus, we can compute $\hat{\mathbf{x}} = (\hat{x}, \hat{y})^T$ iteratively following the rule

$$\hat{\mathbf{x}}_{n+1} = \hat{\mathbf{x}}_n - J_f^{-1}(\hat{\mathbf{x}}_n) f(\hat{\mathbf{x}}_n). \tag{3.39}$$

$J_f^{-1}(\hat{\mathbf{x}}_n)$ is the 2×2 Jacobian of $f(\hat{\mathbf{x}})$ with

$$J_f^{-1}(\hat{\mathbf{x}}_n) = \begin{bmatrix} \frac{\partial f_x(\hat{\mathbf{x}})}{\partial \hat{x}} & \frac{\partial f_x(\hat{\mathbf{x}})}{\partial \hat{y}} \\ \frac{\partial f_y(\hat{\mathbf{x}})}{\partial \hat{x}} & \frac{\partial f_y(\hat{\mathbf{x}})}{\partial \hat{y}} \end{bmatrix}_{\hat{\mathbf{x}}=\hat{\mathbf{x}}_n} = - \begin{bmatrix} a_1 + 2a_3\hat{x}_n + a_5\hat{y}_n & a_2 + 2a_4\hat{y}_n + a_5\hat{x}_n \\ b_1 + 2b_3\hat{x}_n + b_5\hat{y}_n & b_2 + 2b_4\hat{y}_n + b_5\hat{x}_n \end{bmatrix} \tag{3.40}$$

As starting value $\hat{\mathbf{x}}_0$ we use an estimation resulting from inverting only the linear part of the transformation, i.e., parameter a_3, a_4, a_5, b_3, b_4, and b_5 are set to zero.

3.4 Blending and Color Pixel Assignment

After the estimation of all long-term registration parameters $\mathbf{k}_{r,i}$, $i = 0, \ldots, N_f - 1$ the final sprite generation step is to assign the best fitting pixel values to the sprite image. This process is also known as blending. It is done by warping all registered images into the reference coordinate system and selecting the optimal pixel value from all temporal candidates. Blending aims two main objectives. First, for the rigid background content the pixel value should minimize the derivation in a *maximum likelihood* sense. Those derivations are caused by illumination changes over the sequence, image noise, or registration errors. Second, it is of highest importance to filter those pixel candidates out that belong to foreground objects moving independently from the camera motion.

3.4.1 Pixel Candidates

For the estimation of pixel values that represent the background of a sequence for a certain sprite we want to use as many information as possible. Therefore, all available frames I_i are warped into the coordinate system of the reference frame I_r. Displaying the warped images over time yields a deformed image cylinder. The principle of blending is depicted in Figure 3.23. All pixels that are located at the same spatial position $(x_r, y_r)^T$ are used for the estimation of sprite pixel value at this position.

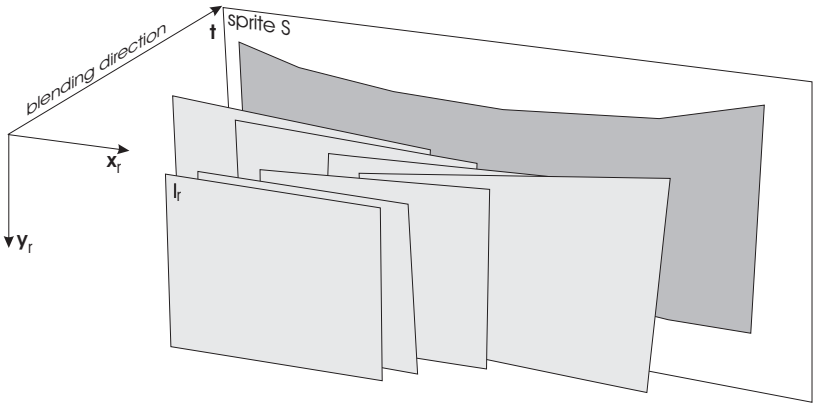


Figure 3.23: Principle of sprite blending: The warped images do spatially overlap and build up a image cylinder. Here the first frame is used as reference frame. The pixel assignment of the final sprite is performed into the direction of time.

To demonstrate the principle we generated horizontal cross sections through the computed image cylinders, i.e., the image of all pixel candidates for a horizontal line in the final sprite S. Figure 3.24 shows the pixel candidates over time for sequence "Stephan" for different horizontal lines of the sprite. One can easily identify the pixels of the background region (vertical structure) and pixels belonging to the independently moving foreground objects (non-vertical image cluster).

3.4.2 Robust Median Filtering

Since for many scene shots with rotating camera the foreground objects are moving quick enough to cover different parts of the background, it is likely that the majority of sprite pixel candidates for one certain position $(x_r, y_r)^T$ represents the background. If this is the case, the median value represented by

$$S(x_r, y_r) = \underset{\forall\, i}{\text{median}} \left(I_i(\mathcal{T}(\mathbf{k}_{r,i}; x_r, y_r)) \right) \tag{3.41}$$

would be a proper choice for pixel assignment. It is clear that only those image values $I_i(\mathbf{k}_{r,i}; x_r, y_r)$ are considered that have pixel positions inside the image I_i.

Amongst the background pixel candidates the median value may not be the pixel value that minimizes the squared error difference $E(x_r, y_r)$

$$E(x_r, y_r) = \sum_{i \in T_{b(x_r, y_r)}} \left(S(x_r, y_r) - I_i(\mathcal{T}(\mathbf{k}_{r,i}; x_r, y_r)) \right)^2 , \tag{3.42}$$

where $T_{b(x_r,y_r)}$ represents the set of all temporal samples for a certain pixel position $(x_r, y_r)^T$, for which the background is not covered by any other object. We assume the value $I_i(\mathcal{T}(\mathbf{k}_{r,i}))$ for all background pixels to be Gaussian distributed. $E(x_r, y_r)$ is a measure for the expected value of $(S(x_r, y_r) - I_i(\mathcal{T}(\mathbf{k}_{r,i}; x_r, y_r)))^2$, which is $\chi^2_1(\mu^2)$ distributed, where μ is the expected value of the underlying Gaussian distribution of $S(x_r, y_r) - I_i(\mathcal{T}(\mathbf{k}_{r,i}; x_r, y_r))$. The expected

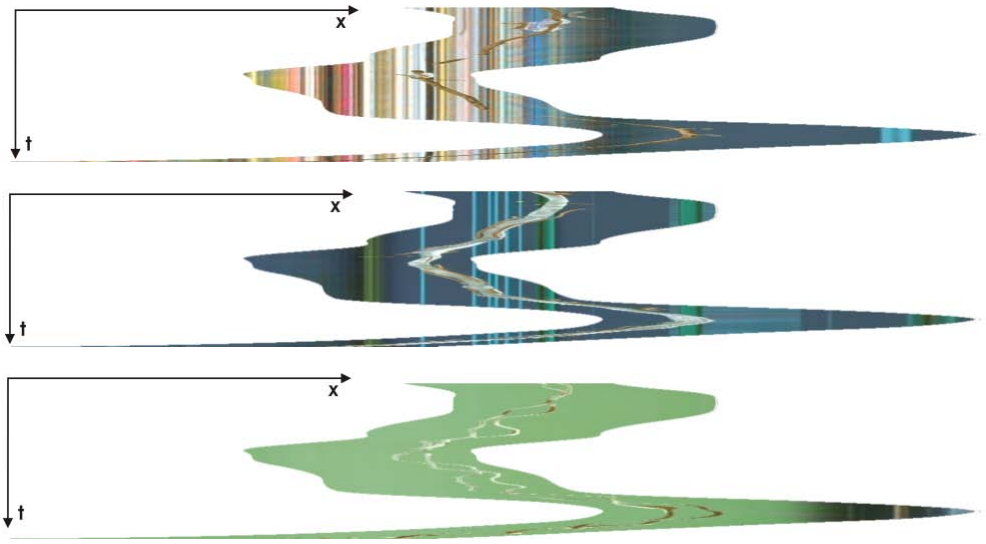


Figure 3.24: Color pixel candidates for sprite of first 250 frames of sequence "Stefan" for one horizontal line: $y_r = 75$ (upper), $y_r = 120$ (middle), $y_r = 190$ (lower). The rigid background content has a vertical structure after motion compensation. Irregularities of this structure are mainly caused by foreground pixels.

value of $\chi^2_1(\mu^2)$ becomes minimal if $\mu = 0$. Hence, it follows that the squared error difference $E(x_r, y_r)$ becomes minimal if

$$S(x_r, y_r) = \underset{i \in T_{b(x_r, y_r)}}{\text{mean}} \left(I_i(\mathcal{T}(\mathbf{k}_{r,i}; x_r, y_r)) \right). \tag{3.43}$$

We estimate a range of reliability around the median value calculated in Equation 3.42. All pixel values that fall into this range will be considered for mean calculation. An image pixel $I_n\left(\mathcal{T}(\mathbf{k}_{r,n}; x_r, y_r)\right)$ is chosen if it holds:

$$\begin{aligned} \left| I_n\left(\mathcal{T}(\mathbf{k}_{r,n}; x_r, y_r)\right) - m \right| &\leq A \cdot \underset{\forall \tau}{median} \left| I_\tau\left(\mathcal{T}(\mathbf{k}_{r,\tau}; x_r, y_r)\right) - m \right| \\ \text{with} \quad m &= \underset{\forall \upsilon}{median} \left(I_\upsilon\left(\mathcal{T}(\mathbf{k}_{r,\upsilon}; x_r, y_r)\right) \right). \end{aligned} \tag{3.44}$$

3.4.3 Correlation Analysis

If the background content for a certain sprite image region is covered by the foreground objects median-based robust filtering will not lead to the desired results. In that case, correlation-based analysis can be utilized to generate proper background images. The main idea driving this analysis is the fact that the background of a region appearing in different compensated images is spatially highly correlated, whereas, due to the non-rigidity of the foreground objects, these object's images are less correlated for any sprite region.

In [28] it was proposed to generate a block similarity matrix for every 8×8 block of the sprite image. It is achieved by spatially correlating blocks at the same sprite position over time. For lower computational load it can be approximated by determining the normalized

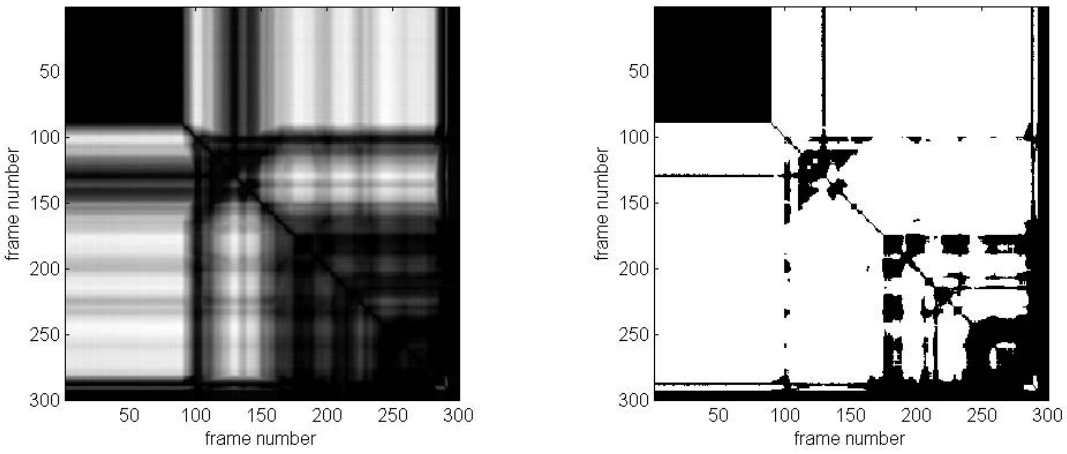


Figure 3.25: Similarity matrix for one block of sequence "Hall and monitor" (300 frames): (a) Correlation levels following Equation 3.46, (b) Binarized version of similarity matrix. Dark values represent high spatial correlation for the respective pair of blocks.

sum of absolute differences (SAD) between two blocks $\mathbf{B}^{t_1}(a,b)$ and $\mathbf{B}^{t_2}(a,b)$ of different time marks, where $\mathbf{B}^t(a,b)$ is a 8×8 matrix with

$$\left(\mathbf{B}^t(a,b)\right)_{k,l} = I_t\left(\mathcal{T}(\mathbf{k}_{r,t}, 8a+k, 8b+l)\right) \quad k,l = 0,\ldots,7 \tag{3.45}$$

Indexes a and b represent the single blocks of the sprite and can also have negative values. The similarity value is calculated by

$$\mathbf{M}_{ab}(t_1,t_2) = \frac{1}{64}\sum_{l=0}^{7}\sum_{k=0}^{7}\left|\left(\mathbf{B}^{t_1}(a,b)\right)_{k,l} - \left(\mathbf{B}^{t_2}(a,b)\right)_{k,l}\right|. \tag{3.46}$$

Highly correlated blocks result in low values for $\mathbf{M}_{ab}(t_1,t_2)$. Binarizing the symmetric block similarity matrix by applying a threshold yields only the highly correlated blocks. An example of the similarity matrix and its binarization is shown in Figure 3.25. Black regions show the temporal values t_1 and t_2, which have high correlation. Since the block similarity matrix is symmetric, one only has to concentrate on rows or columns. If we assume that the correlation of the background does not change over time, the column with the broadest distribution of high correlated blocks, i.e., the index t_2 of $M_{ab}(t_1,t_2)$ having a multitude of broad distributed zeros is chosen as prediction index. In other words, if the correlation peaks are equally distributed over time it is likely that they are caused by rigid background objects. As measure for the distribution of high correlation peaks we simply use the variance of temporal indexes. Finally, all highly correlated blocks with the prediction index block are used for robust median filtering described in Section 3.4.2.

In Figures 3.26 and 3.27 we present the results for two different sequences. Sequence "Hall and monitor" is captured over 300 frames with a fixed camera so that all transformation parameters represent the identity transformation. However, this does not affect the blending process. Moreover, the resulting sprite image has the same dimensions as the single frames

and therefore can be compared with frames that do not contain any foreground objects. This is the case for the first five frames. As one can easy realize, the correlation analysis is subjectively the most robust blending method. For objective comparison we also calculated the PSNR value, where the first frame of the shot was taken as reference. We measured PSNR values of 36.35 dB for simple mean filtering, 42.16 dB for robust median filtering, and 45.45 dB for the correlation-based method.

Also for sprite generation with rotating and zooming camera the described temporal filter methods work very well. In sequence "Vectra" (142 frames) we generated the sprite for the last 67 frames and applied the median-based filter method as well as the correlation-based method. The sequence has great portions of foreground content, which unfortunately is very homogeneous, e.g., the spatial correlation of the foreground object – a white car – is very high itself. One can observe that the size of artifacts caused by foreground objects is smaller for the correlation analysis approach. However, due to its block-based processing, the wrongly estimated blocks subjectively seem to disturb the perceived image quality in a very noticeable way.

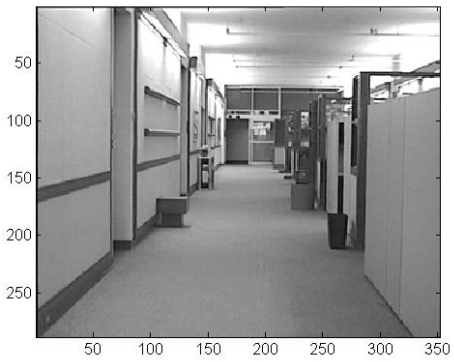
3.4.4 Experimental Results for Blending

In this section we will present the results for sprite generation after the blending process for several sequences. In most cases the foreground objects are small enough that the robust median method can be chosen as temporal filtering. The advantage of this method is the lower computational complexity.

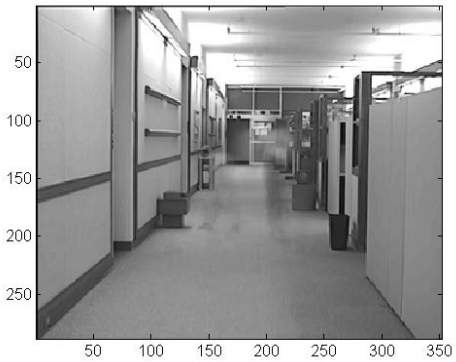
In Figures 3.28, 3.29, and 3.30 the resulting sprites for sequences "Mountain" (129 frames, 720×400), "Desert" (240 frames, 720×400), and "Charlottenburg" (450 frames, 360×288) respectively, are shown. In the particular cutouts one can see that the foreground object as part of the preliminary sprite vanishes. Additionally, effects caused by motion blur in the original sequence are significantly reduced. The underlying motion model is the 8 parameter perspective motion model. Finally, we depict the result of sprite generation for sequence "Stefan" in Figure 3.31 using the perspective motion model as well.

3.5 Comparison for Different 2D Transformations

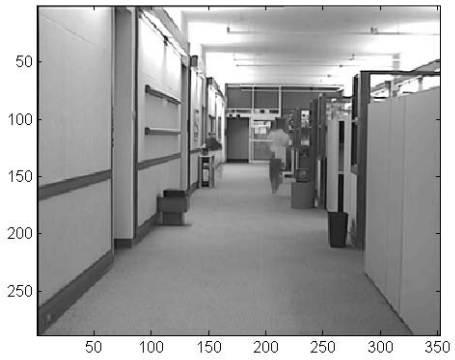
In this section we show the accuracy of the long-term estimation and sprite blending approach while comparing different transformation models. As shown in Section 3.2.3 perspective transformation-based image mapping yields best results for short-term registration. As test sequences for long-term registration and mosaic generation two different scenarios where applied. In the first sequence "Stefan" fast camera movements and great portions of foreground regions appear. In the second sequence "Monitor" (150 frames) no foreground appears and the speed of camera movement is rather slow while the panning angle is kept



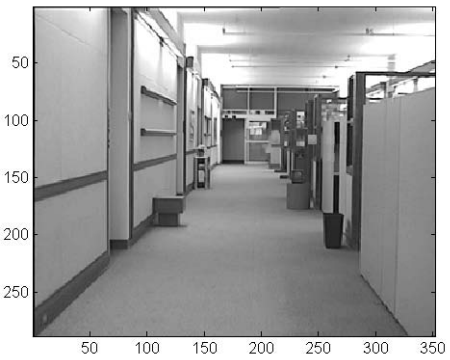
(a) Original frame 1



(b) Mean filter



(c) Robust median filter



(d) Correlation analysis

Figure 3.26: Blending results for sequence "Hall and monitor" (352 × 288, 300 frames).

(a) Robust median filter

(b) Correlation analysis

Figure 3.27: Blending results for sprite of sequence "Vectra" (352 × 288, 67 frames (76 to 142)).

(a) Preliminary sprite

(b) Final sprite after blending

(c) Preleminary sprite – cutout

(d) Final sprite after blending – cutout

Figure 3.28: Sprite generation result for sequence "Mountain".

(a) Preliminary sprite

(b) Final sprite after blending

(c) Preleminary sprite – cutout

(d) Final sprite after blending – cutout

Figure 3.29: Sprite generation result for sequence "Desert".

(a) Preliminary sprite

(b) Final sprite after blending

(c) Preleminary sprite – cutout

(d) Final sprite after blending – cutout

Figure 3.30: Sprite generation result for sequence "Charlottenburg".

(a) Preliminary sprite

(b) Final sprite after blending

Figure 3.31: Sprite generation result for sequence "Stefan" (300 frames). The reference frame is frame 254.

small. For both sequences the re-projection background quality was measured for different camera motion models. The average background PSNR values for both test videos are given in Table 3.2. In order to strengthen the obtained results we additionally show the approximated average values for sequence "Stefan" as presented in [129] and in [79]. For each single motion model we exceed the background PSNR values.

Transformation model	"Stefan"	"Stefan" in [129]	"Stefan" in [79]	"Monitor"
affine	21.59	≈ 21.0	-	28.84
perspective	25.51	≈ 25.5	24.11 (100 frames)	32.93
parabolic	26.27	≈ 26.0	-	33.05

Table 3.2: Average background PSNR in dB for re-projected frames from the long-term mosaic of sequences "Stefan" and "Monitor". The results for "Stefan" are compared to the ones presented in [129] and [79].

The most important fact is, that in contrast to the short-term compensated frames the parabolic model yields much better objective results. The difference to the perspective model increases with faster and wider camera motion. This can be motivated by a higher degree of freedom for the nonlinear parabolic model, which partially covers deviations from the theoretical camera motion model primarily restricted to rotation and zoom. Since those effects have higher impact for larger camera angles, they can only be measured for long-term compensated frames.

The curves of background PSNR for each frame are given in Figures 3.32 and 3.34. For the "Stefan" sequence the parabolic motion-based sprite yields re-projection results up to 3 dB higher than the perspective motion-based sprite. Note that it is impossible to generate a single sprite for the whole sequence using the parabolic motion model. Thus, we only compared results for the first 250 frames of the sequence using frame 1 as reference frame. In Figure 3.33 the final sprite results for sequence "Monitor" using perspective and parabolic motion are shown. Due to small and slow camera motion, the differences are very small, which is reflected in the bPSNR curves of Figure 3.34. Here, frame 76 is chosen as reference frame.

3.6 Chapter Summary

In this chapter we described our approach for single sprite generation in detail. Starting with an in-depth discussion of the two main 2D image registration methods, feature-based and photometric consistency-based, we developed an hierarchical algorithm for short-term image registration that fulfills our needs of an very exact estimation of global camera parameters while being very robust in case of image ambiguities and independently moving foreground objects.

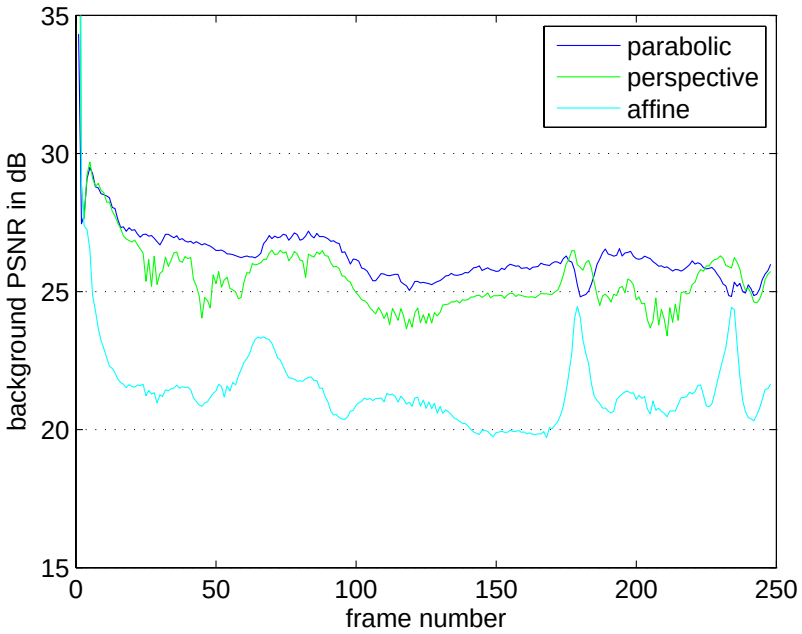


Figure 3.32: Comparison of sprite re-projection results for different motion models using the background PSNR for sequence "Stefan".

(a) Perspective sprite (b) Parabolic sprite

Figure 3.33: Sprite generation result for sequence "Monitor" (150 frames).

The hierarchical approach (Section 3.2) is build up on approaches proposed by Smolic [133] and Dufaux [18] but achieves major improvements while concentrating on exact nonlinear energy minimization and high order image interpolation aspects. Experimental results strongly proof the success of this strategy.

In order to generate one oversized sprite as summarization of all rigid background objects long-term registration into the coordinate system of one reference frame has to be performed. In Section 3.3 we propose a direct estimation method that corrects error accumulation obtained by concatenating short-term frame-to-frame transformations. Therefore, a preliminary sprite is being built by adding newly discovered content during the registration of each frame. Also aspects of inversion and concatenating of nonlinear transformations have been discussed and incorporated into this approach.

Since the preliminary sprite is likely to contain foreground artifacts and does not nec-

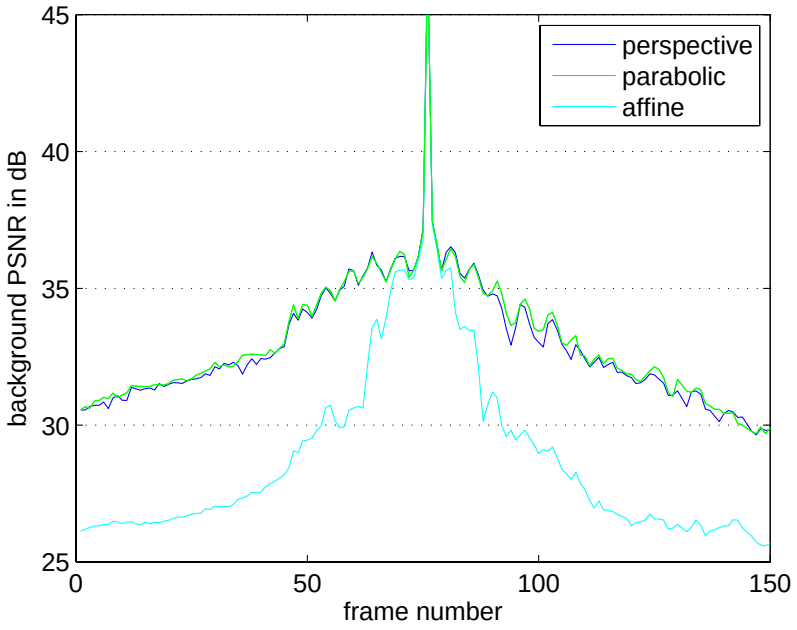


Figure 3.34: Comparison of sprite re-projection results for different motion models using the background PSNR for sequence "Monitor".

essarily represent the background content in an maximum likelihood sense, we presented a robust blending method for final sprite pixel value assignment in section 3.4. To achieve a robust and exact blending two methods of temporal filtering of motion compensated image values where discussed: mean filtering and correlation analysis exploiting spatial similarity of the background images. We found out that correlation analysis objectively yields better results but can subjectively be perceived in a similar manner due to the block character of estimation errors. Additionally the complexity of the blending process is much higher.

Finally we compared our sprite generation approach for different assumed camera motion models. As expected, models of higher degree achieve better sprite quality in terms of background re-projection errors. This is in contrast to short-term compensation results – also referred to as global motion estimation – where the perspective model achieves the best background estimation.

The use of these sprites for further applications like segmentation and video coding is limited due to objective limits of the re-projection quality. One should keep in mind that the representation of any video background content using sprites and the set of transformation parameters is limited by the above given background PSNR values. Subjectively this limitation is not that serious. Methods of improving the re-projection quality will be presented in the next chapters.

Chapter 4

Super-resolution (SR) and Sprites

The lack of mathematical education could not appear more evidently than in an excessive accurate calculation.

Carl Friedrich Gauß (1777-1855)

In this chapter we especially focus on the fact that in scenes recorded with a fixed camera – mainly pan, tilt, zoom, and roll occurs – we observe multiple samples of the background objects without changing the position of view. Since due to arbitrary camera rotations and perspective or parabolic image warps the pixel motions between temporal neighboring frames are of non-integer values, we are enabled to reconstruct super-resolution background sprites (SR sprites).

In the next sections the fundamentals and main paradigms of image super-resolution will be discussed briefly. We will introduce the observation model that relates high resolution images (HR images) with low resolution camera images (LR images) and mention several approaches for its inversion. Starting from one general description for SR image generation, we will derive a method, which is a simplified but very effective way of SR computation and can easily be incorporated into the sprite generation framework described in chapter 3.

4.1 Fundamentals of Super-resolution

Spatial image super-resolution is a very intensively studied topic for over 20 years because it promises the improvement of the inherent resolution limitation of captured low resolution images [99], [6]. The main objective is to construct one or more high resolution images by processing several LR images captured by different cameras or at different points in time. This can be achieved by estimating the inverse of the observation model, which relates LR images to HR images [99], and usually consists of three stages: registration, interpolation, and restoration. Several methods for image super-resolution have been developed. In the following section we will give a short introduction to the problem addressed by super-resolution and explain the common attempts for a solution.

(a) One camera - multiple images in time

(b) Multiple camera - one image in time

(c) Moving video camera

Figure 4.1: Basic setups for SR image reconstruction. Source: [2]

4.1.1 Introduction to Image Super-Resolution

Today's photo and video cameras are equipped with common CCD or CMOS image sensors to capture digital images. In approximation, the sensor can be regarded as an array of sensor elements covering a rectangular base area. Since the sensors are produced by typical semiconductor surface processing, the transfer elements, which read out the sensor element's charge, occupy a part of the sensor's surface. Thus, a fill-factor of 100% is still impossible [103]. Modern sensors increase the fill-factor by applying an array of microlenses on top of the sensor. On the other side, the size of the elements is limited towards a minimum since the amount of light, which has main impact to the elements charge, decreases with the elements surface. Therefore, the spatial resolution of the resulting image is limited and goes along with a loss of information during the image capture process. Especially for video capturing devices the amount of picture elements is rather low. Thus, a post-processing step using signal processing techniques promises an increase of resolution. This can be achieved by combining observed multiple LR images to obtain a HR image or a sequence. The basic premise for this process is that one has several images of the same content captured with non-integer shifts of the sensor grid. Figure 4.1 depicts three main setups for SR image reconstruction.

As a simple example for the effects of super-resolution imaging algorithms we can assume to capture two pictures with a camera using an image sensor with a regular sensor array – the sensor elements are approximated by points on a grid without spatial expansion – having a spatial sampling distance of S_{LR} in horizontal and vertical direction. In order to increase the sampling point density we can capture the second image by only shifting the sensor plane with a shift of $(S_{LR}/2, S_{LR}/2)^T$. Combining both images we yield an image of double the number of sampling points for the overlapping area. The sample distance of the HR image then becomes $S_{HR} = S_{LR}/\sqrt{2}$ into both diagonal directions. Figure 4.2 shows the principle of simple image super-resolution. Regarding the presentation of the resulting image in the frequency domain the lower sample distance has especially impact on the maximally presentable spatial frequency. Therefore, super-resolution is very useful to overcome limitations caused by spatial aliasing in high frequency image areas like edges and corners.

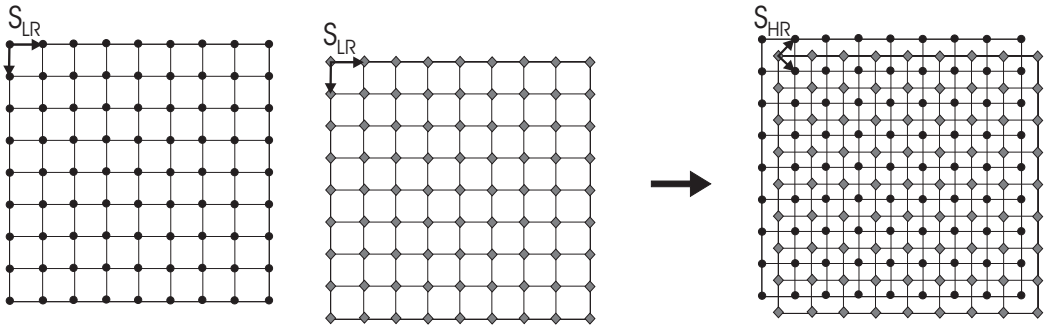


Figure 4.2: Ideal case for super-resolution image reconstruction with an in-plane sensor shift of exact half the spatial sampling distance of the sensor. The basis vectors of the sampling grid are marked with black arrows.

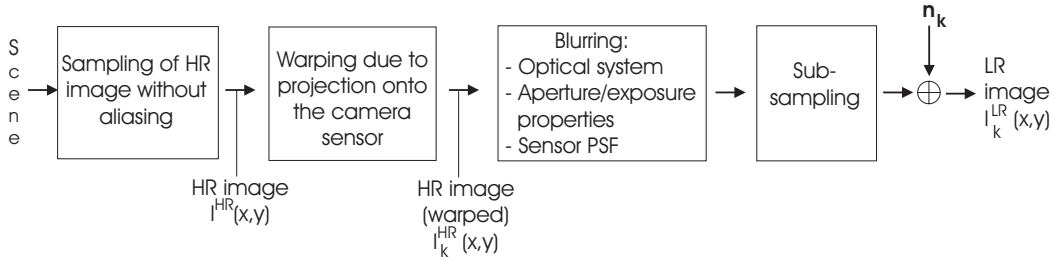


Figure 4.3: Observation model for SR imaging. [99]

Unfortunately, the positions of sensors for different frames of a video shot do not relate to each other by simple in-plane shifts. Additionally, the single sensor elements cannot be represented by a two-dimensional delta comb and, due to properties of the optics and the sensor itself, the impulse response of the camera system is spatially spread. Hence, for the creation of HR images applying super-resolution computational complex algorithms have to be utilized. Their main objective is to invert the *observation model* relating LR images to HR images.

4.1.2 The Observation Model

Figure 4.3 shows the model, which generally describes the mapping of a continuous real scene onto a number of discrete LR image. As intermediate result the discrete HR image without any spatial aliasing effects is assumed to exist. Mathematically we can formulate the observation model as follows:

$$I_k^{LR}(x', y') = D\left(B_k\left(I^{HR}(x, y)\right)\right) + \mathbf{n}_k \qquad \text{with} \quad (x', y') = \mathcal{T}_k\left((x, y)\right), \tag{4.1}$$

where $\mathcal{T}_k^{-1}(\cdot)$ represents the inverse of the kth image transformation of the HR image pixels, $B_k(\cdot)$ represents the blurring operator and $D(\cdot)$ represents the decimation operator. The resulting image I_k^{LR} is corrupted by additive noise $\mathbf{n}_k$. Note that in contrast to the conventional description with matrices [99], [163] we prefer to describe the single operations of the observation model using function notation, which is more precise.

4.2 Super-resolution Approaches, Constraints, and Feasibility

Following the observation model of Equation 4.1 the HR image construction process is not only dependent on the observed LR images. Moreover, additional noise has to be removed and the impact of the blurring operator has to be estimated. In several approaches the blur handling is divided into motion blur removal, caused by fast motion and low shutter speed, and a deconvolution with an estimated *point spread function* (PSF) being the impulse response of the whole optical system. In most cases the PSF is assumed to be Gaussian. There exist many approaches of estimating the PSF directly from an image [111], from stellar images [105], or by simply using laser beams.

Most of the SR reconstruction algorithms proposed in the literature conduct the generation of the HR image in three stages. First, the LR image are registered, second the pixel values of the HR image are assigned by blending and interpolation to fit the HR image grid, and third, the HR image is deblurred in a restoration process. Like in [165] the three stages can be processed iteratively in order to jointly optimize registration and restoration.

4.2.1 Image Registration

The registration of images in order to obtain the exact position of any pixel in relation to the HR image grid is the most critical part of the super-resolution process. Since the registration accuracy has to be verified in sub-pixel values, the registration must be very exact. Most common registration methods are parametric motion estimation techniques, as we are using for our sprite generation scheme and *optical flow*-based motion estimation techniques, where every pixel is assigned with a motion vector pointing toward the position of the pixel in the registered image.

For exact registration we use the combined short- and long-term motion estimation process described in Chapter 3. In order to obtain the desired super-resolution the preliminary sprite is constructed in an up-sampled domain, depending on the image magnification we want to achieve. Ye [165] proposed to up-sample as well the LR input images, which is only necessary if low quality interpolation techniques (bilinear, bicubic) are applied in the nonlinear energy minimization process. As we utilize high order B-spline interpolation the up-sampling step would not improve the registration quality.

4.2.2 Image Interpolation

The registered and warped LR images I_k^{LR} do generally not fit onto the regular HR image pixel grid. Thus, assigning the pixel values for the HR image using correctly warped pixels of the observed LR images always causes degradation due to interpolation techniques. See Figure 4.4 for a schematic illustration of the interpolation problem. Some authors have

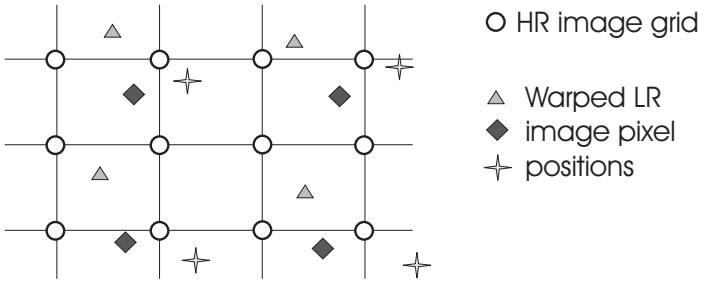


Figure 4.4: On the assignment of pixel value for SR image reconstruction.

proposed not to do any interpolation causing blur. In [136] only those pixel values are used that have very small distance to the HR image grid. This yields good SR results for small camera movements so that the number of LR images containing a certain region of the original scene is high enough. Otherwise, the resulting super-resolution image will not be densely filled. Additionally, this technique complicates a joint approach of robust image blending with foreground removal and super-resolution.

4.2.3 Image Restoration

For many SR reconstruction approaches, the blurring process is assumed to be known [99]. Thus, the restoration process for the removal of image blur caused by the sensor integration and the optical apparatus can be done using deconvolution methods with a priori knowledge. The overall blur is represented by the *point spread function* $\mathrm{PSF}_k(x, y)$ [4]. It can be decomposed into two components

$$\mathrm{PSF}_k(x, y) = (a_k * w_k)\,(x, y), \tag{4.2}$$

where a_k represents the sensor properties and w_k represents the optical apparatus (lens, motion blur due to temporal integration, defocus).

Several techniques have been proposed in order to identify the blur, i.e., the $\mathrm{PSF}_k(x, y)$, without any prior knowledge. Those approaches are also known as *blind SR image reconstruction*. Because we are mainly dealing with video sequences without any additional information, we would have to apply these techniques. Since the identification of $\mathrm{PSF}_k(x, y)$ is a very complex topic itself, this process goes beyond the scope of this thesis. Instead, we assume a $\mathrm{PSF}_k(x, y)$ being close to a 2D Dirac function $\delta(x, y)$ and examine what resolution enhancement can be achieved by only processing high quality registration and interpolation approaches.

4.2.4 Observation Model Inversion Approaches

In the following sections we briefly describe the different approaches of constructing HR-images using super-resolution.

4.2.4.1 Nonuniform Interpolation

This approach concentrates on the final pixel assignment of the HR image pixels. As shown in Figure 4.4 there may be several pixel candidates after warping the registered images into the coordinate system of the HR image. Then the techniques compute a uniformly spaced sampling grid using nonuniform interpolation techniques. Finally, the blur is removed by restoration process using deconvolution methods, e.g., Wiener filtering.

4.2.4.2 Frequency Domain

If we assume bandlimited continuous HR images in the observation model super-resolution construction can also be performed in the frequency domain. An additional constraint to this type of SR generation is the shifting property of the *Fourier transform*. Therefore, only images obtained by mainly translational camera motion are useful to be processed. Every shifted version of the continuous HR image I_{cont}^{HR} can then be written as

$$\mathcal{I}_k^c(\omega_1, \omega_2) = \mathcal{F}(I_{k,cont}^{HR}) = \mathcal{F}(I_{cont}^{HR}) e^{j\omega_1 r_k + j\omega_2 s_k}, \tag{4.3}$$

where $\mathcal{F}(\cdot)$ denotes the continuous Fourier transform, ω_1 and ω_1 represent the angular frequencies in horizontal and vertical direction, and the vector $(r_k, s_k)^T$ represents the already determined image shift.

The frequency domain approach then exploits the simple relation between continuous and discrete Fourier transformation for bandlimited signals causing the aliasing effect

$$\mathcal{I}_k^d(\Omega_1, \Omega_2) = \mathcal{DFT}(I_k^{HR}) = \frac{1}{T_1 T_2} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \mathcal{I}_k^c \left(\frac{2\pi}{T_1} \left(\frac{\Omega_1}{M} + p \right), \frac{2\pi}{T_2} \left(\frac{\Omega_2}{N} + q \right) \right). \tag{4.4}$$

In Equation 4.4 Ω_1 and Ω_2 represent the normalized angular frequencies representing both directions of the discreet image, T_1 and T_2 represent the spatial sampling periods of the observed LR images, while M and N represent the dimensions of the discrete spectrum of the LR images. Assuming a bandlimited signal $I_{k,cont}^{HR}$ Equations 4.3 and 4.4 can be combined. We obtain an equation for every observed image I_k^{LR}, where the discrete spectrum of the HR image is related to the continuous one. Inverting the system of equations and sampling of the continuous spectrum yields in the construction of the super-resolved HR image.

Due to the use of FFT as DFT, this approach is of low computational complexity. The disadvantage is that only translational motion can be handled properly.

4.2.4.3 Stochastic Approaches

A very popular method of SR image reconstruction is the stochastic approach. It provides a powerful technique to define certain energy terms that can be optimized applying nonlinear gradient-based methods. In most cases an a posteriori probability is maximized. Therefore,

these methods are called *maximum a posteriori* (MAP) estimators. For super-resolution imaging, we define the problem as follows. Given a set of LR images I_k^{LR}, find (or estimate) a HR image I^{HR} for that the a posteriori *probability density function* (PDF) is maximized

$$I^{HR} = \arg \max_{\forall I^{HR}} P\left(I^{HR} | I_1^{LR}, \ldots, I_N^{LR}\right). \tag{4.5}$$

Using Bayes' theorem we can write

$$P\left(I^{HR} | I_1^{LR}, \ldots, I_N^{LR}\right) = \frac{P\left(I_1^{LR}, \ldots, I_N^{LR} | I^{HR}\right) P\left(I^{HR}\right)}{P\left(I_1^{LR}, \ldots, I_N^{LR}\right)}. \tag{4.6}$$

Assuming the denominator of Equation 4.6 to be a constant, one simply has to maximize the numerator. Taking the logarithm of that term – note that the logarithm function is strictly monotonic increasing – Equation 4.5 can be led over to

$$I^{HR} = \arg \max_{\forall I^{HR}} \left\{\ln P\left(I_1^{LR}, \ldots, I_N^{LR} | I^{HR}\right) + \ln P\left(I^{HR}\right)\right\}. \tag{4.7}$$

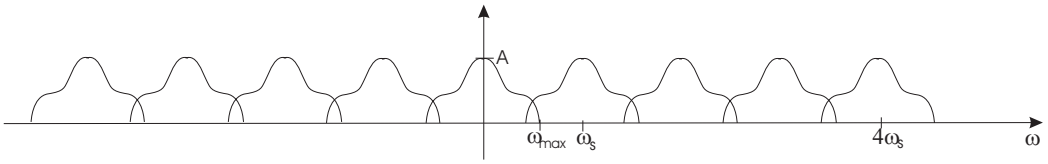
Both probabilities of Equation 4.7 are defined using a priori knowledge. The first part of the maximization term can be regarded as the log-probability of the registration error while the second term – $\ln P\left(I^{HR}\right)$ – represents an abstract formulation of universal paradigms for natural images, e.g., limited signal energy. In many cases it is replaced by a regularization term, which can be used for stabilization and deblurring of the HR image. If we assume all probabilities to be Gaussian and independent we finally obtain

$$I^{HR} = \arg \min_{\forall I^{HR}} \left\{\sum_{k=1}^{N} \left\|I_k^{LR} - D\left(B_k\left(I^{HR}\right)\right)\right\|^2 + \lambda \left\|L\left(I^{HR}\right)\right\|^2\right\}. \tag{4.8}$$

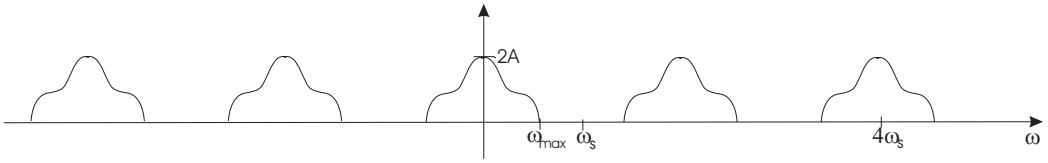
Here, the regularization term is denoted by an operator $L(\cdot)$ that usually has a high-pass characteristic, e.g., a derivative or the Laplacian operator [163]. Thus, $L\left(I^{HR}\right)$ represents a measure for the smoothness of a region in the reconstructed HR image. The Regularization parameter λ works as Lagrangian multiplier and controls the balance between the best estimation of I^{HR} in *mean squared error* (MSE) sense and the smoothness of this image.

4.2.4.4 Regularization

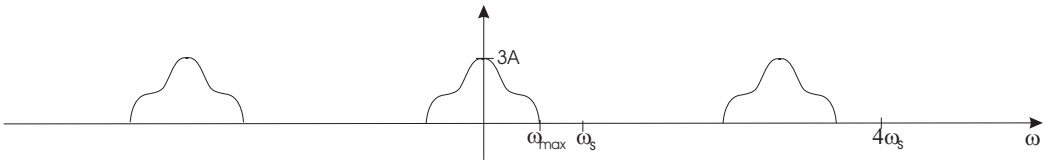
In several publications ([99], [165]) it was observed that for large numbers of LR images and a low amount of noise the influence of the regularization term should be reduced by choosing small values for λ. In fact, undue penalization of sharp edges may prevent the super-resolved estimation from overcoming aliasing limits of the LR image domain. In order to keep the complexity low and the SR algorithm as simple as possible, we do not incorporate the regularization term into the SR sprite generation algorithm.



(a) Originally sampled signal



(b) SR result with $M = 2$



(c) SR result with $M = 3$

Figure 4.5: Amplitude-frequency characteristic of original signal (a) and super-resolved signals with different linear magnification M=2 (b) and M=3 (c).

4.2.5 Spatial Aliasing and SR

Increasing the sampling rate of the resulting image using SR reconstruction methods will have an observable effect on the spatial aliasing. Especially in video sequences spatial aliasing is very disturbing since it causes strong flicker on moving object and texture edges as well as on high frequency structures. For this reason professional TV anchormen never wear very patterned clothes. In the next paragraphs we will perform a 1D - frequency analysis of the super-resolution technique and its impact on spatial aliasing.

Frequency Analysis

For real images and signals it is useful to assume the signals to be bandlimited with a maximum frequency $\omega_{max} = 2\pi f_{max}$. The next assumption we make is that the sampling frequency ω_s for a certain signal is below the minimal frequency of the *Nyquist-Shannon sampling theorem* $\omega_s < 2\omega_{max}$. As an example for the one-dimensional Fourier transformation see figure 4.5-(a). For simplicity of the analysis we consider a number M of LR signals or images $s_m^{LR}(kX)$ with $m = 0, \ldots, M-1$ that have the same regular sampling grid with equal sampling frequency $\omega_s = 2\pi/X$ and a fractional shift Δx_m, which is already estimated by the registration process. Without loss of generality the shift Δx_0 for the first LR signal can be set to zero. All discrete signals are sampled versions of the real image $s(x)$, where x represents one dimension in the spatial domain. The HR signal s^{HR} can be regarded as the

sum of all registered LR signals

$$\begin{aligned} s^{HR} &= \sum_{m=0}^{M-1} s_m^{LR}(kX + \Delta x_m) \\ &= s(x)\left[\sum_{m=0}^{M-1} \delta_X(x - \Delta x_m)\right] \\ &= s(x)\left[\sum_{m=0}^{M-1}\sum_{k=-\infty}^{\infty} \delta(x - kX - \Delta x_m)\right], \end{aligned} \tag{4.9}$$

where $\delta_X(x - \Delta x_m)$ represents the delta comb $\delta_X(x)$ shifted about $\Delta x_m < X$. Performing the Fourier transformation on Equation 4.9 we obtain

$$\begin{aligned} S^{HR}(\omega) &= \mathcal{F}(s^{HR}) = \frac{1}{2\pi} S(\omega) * \left[\omega_X \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_X) + \omega_X \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_X) e^{-jk\omega_X \Delta x_1}\right. \\ &\qquad \left. + \ldots + \omega_x \sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_x) e^{-jk\omega_X \Delta x_{M-1}}\right] \\ &= \frac{1}{X} S(\omega) * \left[\sum_{k=-\infty}^{\infty} \delta(\omega - k\omega_x)\left(1 + e^{-jk\omega_X \Delta x_1} + \ldots + e^{-jk\omega_X \Delta x_{M-1}}\right)\right] \end{aligned} \tag{4.10}$$

In reality the shifted and sampled versions of the continuous signal $s(x)$ can only be combined if the shift Δx_m is an integer multiple of X/L with $L \in \mathbb{N}$. In the ideal case we utilize only M different LR signals to achieve a magnification of M. The single shifts become then

$$\Delta x_m = m\frac{X}{M}, \qquad m = 0, \ldots, M-1. \tag{4.11}$$

The right-handed sum of Equation 4.11 can be written as

$$\left(1 + e^{jk\omega_X \Delta x_1} + \ldots + e^{-jk\omega_X \Delta x_{M-1}}\right) = \sum_{m=0}^{M-1} e^{-jk\omega_X m \frac{X}{M}} = \sum_{m=0}^{M-1} e^{-j2\pi k \frac{m}{M}}. \tag{4.12}$$

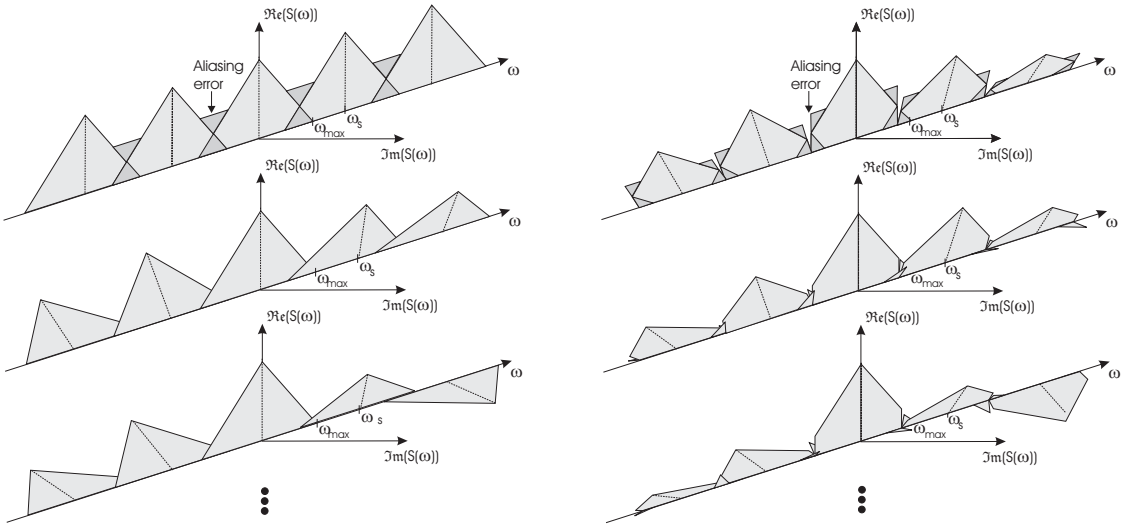
The result of Equation 4.12 can be interpreted as sum of all complex solutions of $\left(\sqrt[M]{1}\right)^k$, which either equals zero or has the result M.

$$\sum_{m=0}^{M-1} e^{-j2\pi k \frac{m}{M}} = \begin{cases} M & k = zM \\ 0 & k \neq zM \end{cases} \qquad z \in \{0, \pm 1, \pm 2, \ldots\} \tag{4.13}$$

In other words, only every Mth spectrum of the original aliased spectrum of $s^{LR}(kX)$ remains. Combining Equation 4.10 and 4.13 yields

$$\begin{aligned} S^{HR}(\omega) &= \frac{M}{X} S(\omega) * \left[\sum_{z=-\infty}^{\infty} \delta(\omega - zM\omega_x)\right] \\ &= \frac{M}{X} \sum_{z=-\infty}^{\infty} S(\omega - zM\omega_x) \end{aligned} \tag{4.14}$$

The results for $M = 2$ and $M = 3$ of Equation 4.14 are exemplarily shown in Figure 4.5. In real applications the shift Δx_m will never be exact like in Equation 4.11. On the other hand



(a) LR signals without interpolation (b) LR signals after interpolation and resampling

Figure 4.6: Spectra of the sampled versions of an exemplary signal with real spectrum (triangle). (a) Undistorted, aliased LR spectra of shifted and sampled original signal that satisfy Equations 4.10 to 4.14. (b) Spectra of interpolated and resampled versions of the same aliased LR signals.

the number of images that can be used for exact registration, e.g., estimation of the shift, is usually much higher than the magnification M. Since we cannot easily interpolate the images in order to approximate the preferred shift – Equation 4.9 would be infringed due to different lowpass filtering of the spectra –, we have to derive a method that can achieve both, finding samples with the preselected shift and keeping the spectra of all LR images as close as possible to the original ones.

Another point we did not mention in this analysis is the existence of nonuniform sampling caused by perspective geometric distortions of the image content. To overcome this problem *projective Fourier analysis* can be applied instead of the above used Euclidean Fourier analysis [152].

4.3 Super-resolution as Extended Blending for Sprites

The generation of video background mosaics already provides a technique of robust and exact image registration. Therefore, it is obvious to apply this method in order to create super-resolution background sprites.

4.3.1 Interpolation and SR

As stated in section 4.2.5 the interpolation of LR image pixels is not desirable for any SR image construction application. Since our proposed sprite generation technique uses intensive interpolation in the blending stage for the motion compensation of all images, we

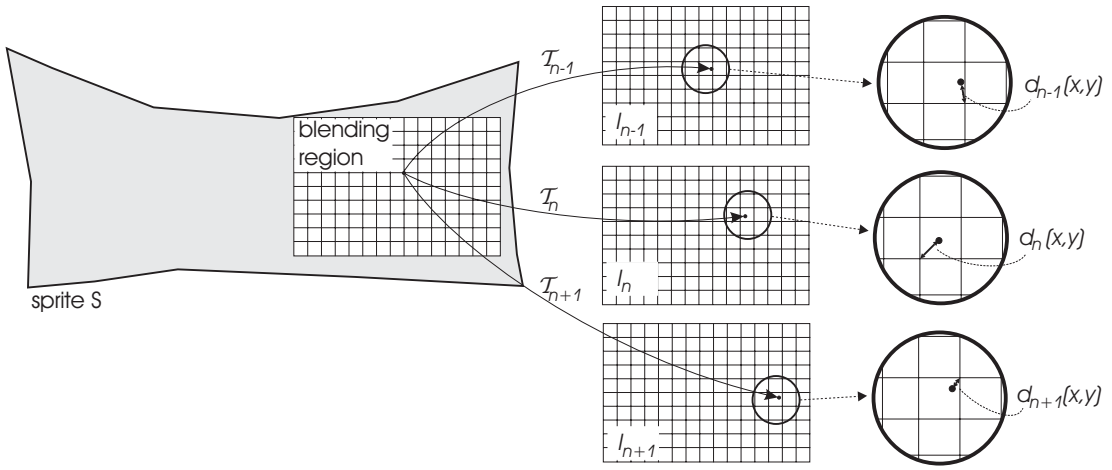


Figure 4.7: On the computation of distances $d_k(x, y)$ for the Super-resolution sprite blending approach: Every warped pixel coordinate $(x, y)^T$ of HR image I^{HR} will generally fall onto a non-integer position $(x'_k, y'_k)^T$ in the LR images I_k^{LR}. We consider only those pixels with the smallest distance $d_k(x, y)$ to the sampling grid of the LR images. In a preprocessing step outliers have to be robustly eliminated.

have to redesign that process. In approximation we can regard the interpolation process as an ideal lowpass filtering with the cut-off frequency $\omega_c = \omega_s/2$ and a subsequent resampling of the continuous signal with a different sample shift. In general, the new sample shift will not be regularly distributed for all LR signals. Thus, the interpolated images will not satisfy Equation 4.10. See Figure 4.6 as an example for necessary 1D signals to achieve SR compared to signals obtained by interpolation. Note, that amplitude and phase of the signal spectra are shown while drawing a three-dimensional version. From the depiction it can be concluded that only signals resampled at the same positions in continuous space are useful to generate anti-aliased SR images and sprites. This fact is exploited for a distance-based method of pixel assignment for sprite SR image generation.

4.3.2 A Distance-based Approach for SR Sprite Construction

Constructing a SR background sprite for a video shot mainly captured by a rotating and zooming camera is in most stages similar to the construction of the common sprite, which was explained in Chapter 3. The SR sprite generation differs from the LR sprite generation process in two stages. First, the preliminary sprite (Section 3.3) is generated in the magnified image domain, which means that the selected reference frame I_R for the long term registration is registered with a linear magnification of M_x in horizontal direction and M_y in vertical direction respectively. For the upsampling we use the already introduced B-spline interpolation filter of 9th order. Second, the blending process is modified towards a minimal utilization of pixel interpolation. The principle of the modified blending can be seen in Figure 4.7.

Additionally to the interpolated pixel value for any LR pixel during the global motion compensation process we store the Euclidean distance of that pixel to the original LR sample

grid. The distance refers always to the original pixel in a nearest neighbor sense. Thus, the distance $d_k(x, y)$, where k is the subscript for the referred LR image, is always in the range

$$0 \leq d_k(x, y) \leq \frac{1}{\sqrt{2}} \text{ pxl.} \tag{4.15}$$

In order to remove the outlier caused by inaccurate registration or independently moving foreground objects we apply the same technique as described in Section 3.4.2, where of all possible pixel candidates for any SR sprite pixel point only those are selected that hold

$$\begin{aligned} |I_n\left(\mathcal{T}(\mathbf{k}_n; x, y)\right) - m| \; &\leq A \cdot \underset{\forall \tau}{median} \, |I_\tau\left(\mathcal{T}(\mathbf{k}_\tau; x, y)\right) - m| \\ \text{with} \quad m &= \underset{\forall \upsilon}{median} \left(I_\upsilon\left(\mathcal{T}(\mathbf{k}_\upsilon; x, y)\right)\right). \end{aligned} \tag{4.16}$$

Unlike in LR sprite generation process we do not assign the mean value for the robust inliers. Finally, we assign only one pixel value of that pixel interpolated $I_n(x', y')$ with the smallest distance value $d_n(x, y)$. We can formulate

$$\hat{I}^{HR}(x, y) = I_n(x', y') \qquad \text{with} \quad n = \arg \min_{\forall k} d_k(x, y), \tag{4.17}$$

where $k = 0 \ldots, K - 1$ and K is the number of frames that provide robust content values for pixel position $(x, y)^T$ of the sprite.

From a certain number of overlapping low resolution images the selected pixel value will have a very small distance $d_n(x, y)$ of below 0.01 pixels. Hence, for large regions of the SR sprite the interpolation and resampling step will have no impact in such way that the super-resolution approach enhances the image quality and reduces the aliasing significantly.

4.4 Resolution Gain and SR Limits

In this section we want to examine how much image enhancement we can gain applying super-resolution at all and with our low complexity approach in particular. It was reported in several articles that even with the most sophisticated SR techniques the maximal resolution enhancement reaches the factor 4 [81]. Moreover, in latest publications the reachable resolution one can achieve was identified with 1.6 [75]. This fact does not imply that there is no research progress but rather new tremendous changes in sensor development that have a significant impact on the LR data characteristics. Modern sensors reach a very high fill factor due to the application of microlenses in front of the sensor array. Also the density of sensor elements has increased and thus the size of elements becomes smaller with every new generation of optical systems. That means, the imaging hardware is nowadays almost reaching image resolutions that the human optical perception system is able to detect. Therefore, it will be difficult to significantly enhance the resolution for natural images. However, in theory an enhancement can always be achieved. There are many applications where the high density sensors cannot be used for, e.g., ultra high speed captures or low cost video cameras. Also one may think of applications where modern sensors are not dense enough, e.g., satellite or space photography or microscopy.

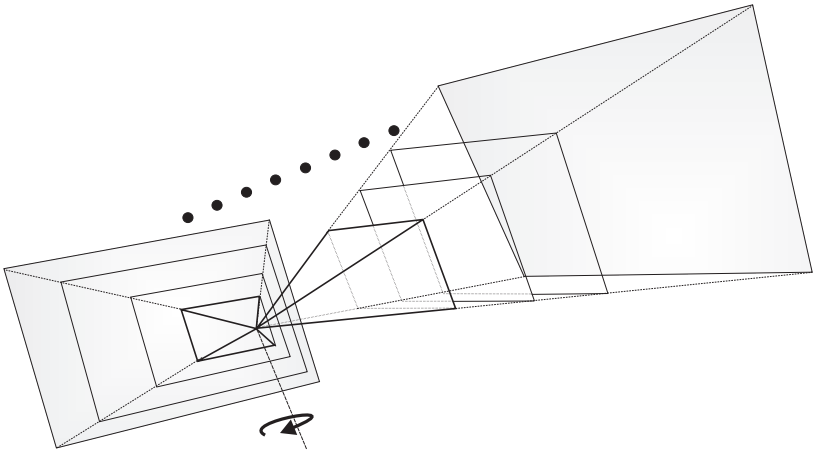


Figure 4.8: Schema of artificial scene rendering for the exploration of the image overlap impact to SR sprite quality.

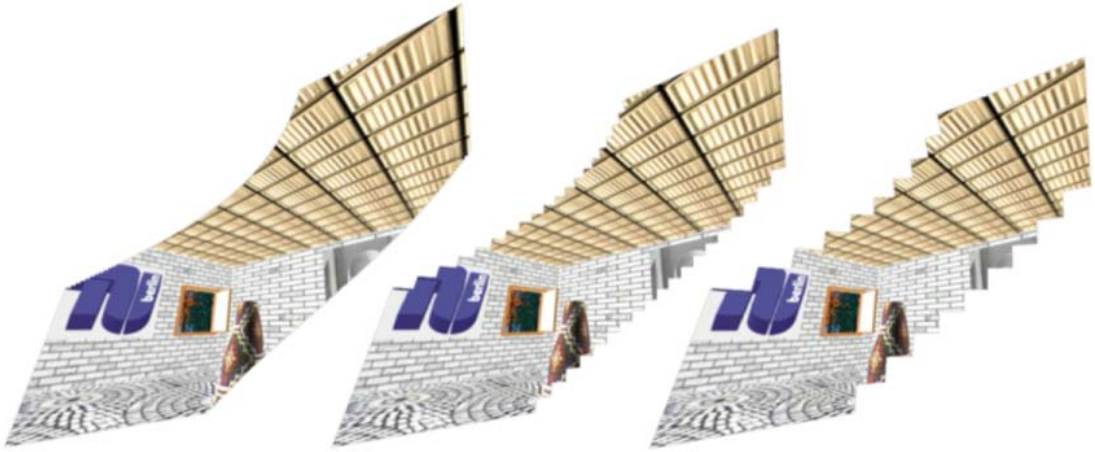


Figure 4.9: SR sprites of diagonal pan for sequence "TUB-roomII" for different image overlap of 96% (left), 81% (middle), and 64% (right). The sprites are down-sampled for illustration, the SR effect can be better evaluated in cutouts.

4.4.1 The Image Overlap Impact to SR

In order to determine the impact of the number of LR images on the SR sprite quality, we generated an artificial sequence using the rendering functionalities of Autodesk's 3ds max software. It is able to render the same content with different camera parameters, e.g., different focal length. To ensure the overlap varying in both spatial directions we apply a diagonal camera pan on a static scenery. See figure 4.8 for a schematic depiction of the scene rendering process. We rendered the scene "TUB-roomII" with different focal length, i.e., different spatial solutions. Also the temporal sampling was varied that we could reach an approximated image overlap of 64%, 81%, and 96% between adjacent video frames. This was achieved by processing only every nth frame of the same sequence. Examples of the resulting SR-sprites are shown in Figure 4.9.

For quality assessment of the constructed SR sprites applying linear magnifications between 1.5 and 3 into both directions we re-project the sprite content into the original frame

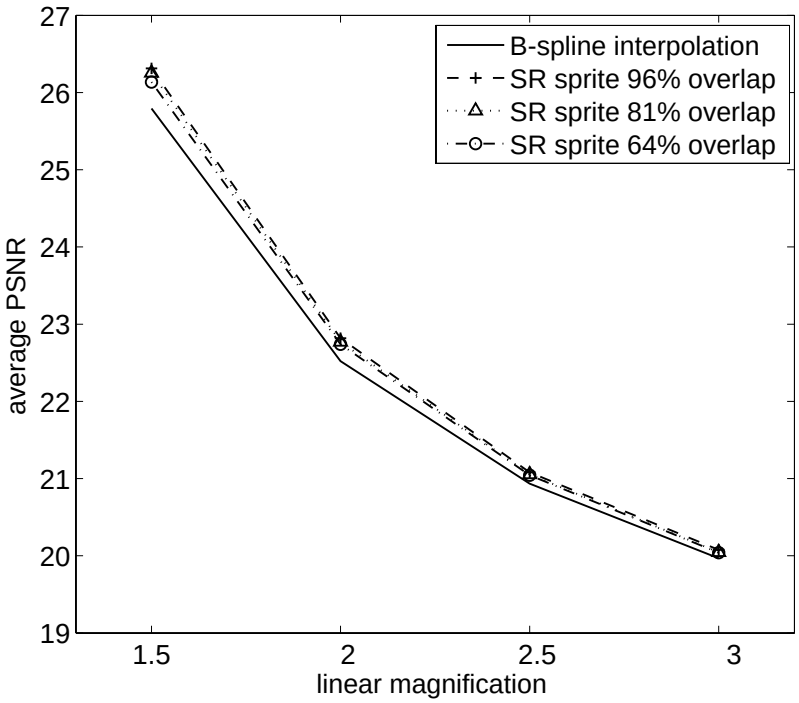


Figure 4.10: Reconstruction results for sequence "TUB-roomII" using LR sprites, SR sprites of sequence with 64%, 81%, 96% image overlap.

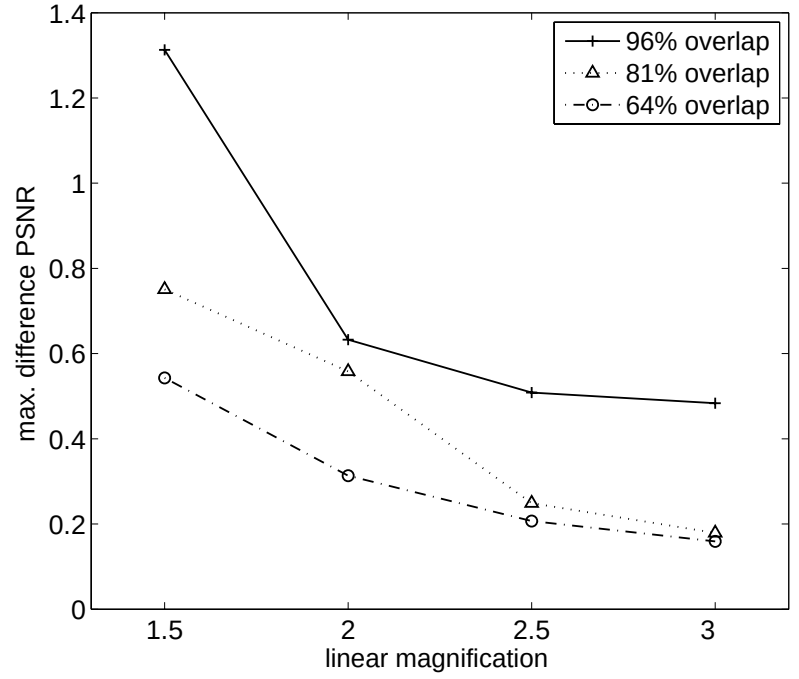


Figure 4.11: Maximum difference PSNR for reconstructed sequence "TUB-roomII" between SR sprite approaches and LR sprite technique.

(a) B-spline (b) 64% overlap (c) 81% overlap (d) 96% overlap

Figure 4.12: Subjective comparison of SR image enhancement: (a) Cropped region after resizing the LR sprite using high order B-spline interpolation; (b)-(d) Cropped region of the SR sprite of linear magnification $M = 2$ with different image overlap.

coordinate systems keeping the enhanced resolution. The resulting sequences with different frame dimensions are than compared with the rendered versions of the same size. The results for different overlap and different magnifications are shown in Figures 4.10 and 4.11. While the former diagram shows the average PSNR curves for different amounts of image overlap compared to simple high order B-spline interpolation, the latter shows the maximum difference between the SR-based reconstruction and the interpolation.

It can be seen in Figure 4.10 that for increasing linear image magnification the gain over conservative interpolation techniques decreases. The maximum gain over B-spline interpolation can be achieved by only small magnification of 1.5 in every direction reaching maximally 0.4 dB for 96% overlap. Since the temporal border frames of a pan shot do not have as many frames to overlap as the middle frames, the average PSNR does not give an exact measure of the gain that can be achieved. Thus, in Figure 4.11 we give the maximal value of difference PSNR between SR generated and interpolation-based image magnification. Here, the gain is much more significant as well as the differences between the results for varying content overlap. In order to give a subjective impression of the image overlap impact to the quality of the resulting sprites in Figure 4.12 a cropped region of the sprites is shown. One can notify that sharpness increases and the number of artifacts decreases for larger overlap between adjacent frames.

4.5 Results and Comparison

In this section we will present the results applying the low complexity SR sprite method described above for several real scene shots. Further, we will compare these results with the proposed LR approach and with other techniques. As stated in the last section a linear magnification over 2 for the resulting sprite image does not lead to a significant improvement over conventional image upsampling methods. Therefore, in our experiments we keep the value $M = 2$ for all applied sequences.

In Table 4.1 the average background PSNR values for three different sequences is given ("Stefan, "Horse", and "Monitor"). Additionally, for the "Stefan" sequence we compare the SR reconstruction results for the perspective and parabolic camera motion model. For this

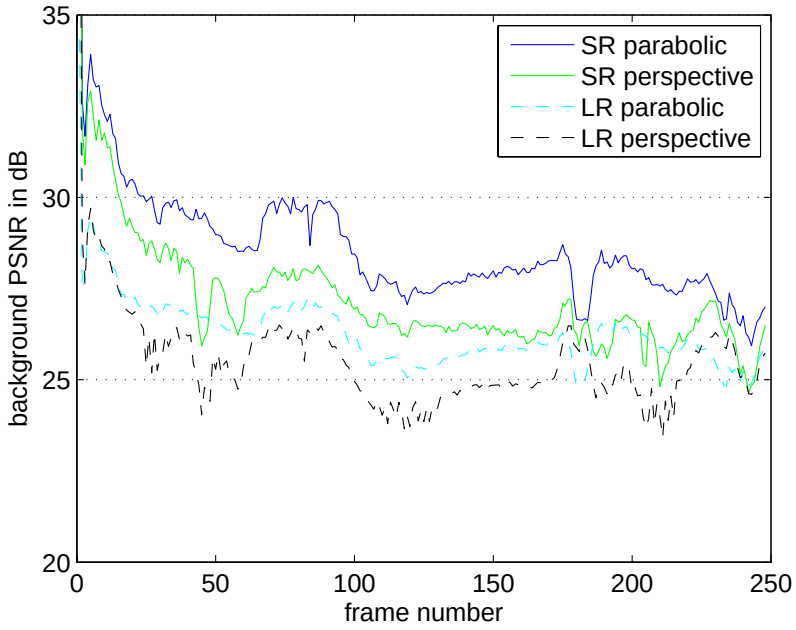


Figure 4.13: Background PSNR over 250 frames of sequence "Stefan" for SR and LR sprite-based reconstruction.

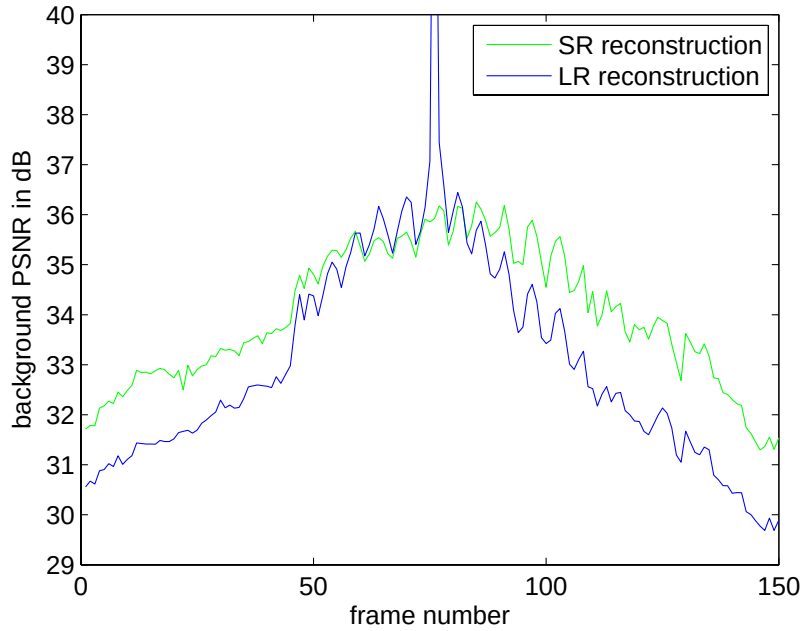


Figure 4.14: Background PSNR over 150 frames of sequence "Monitor" for SR and LR sprite-based reconstruction.

Figure 4.15: SR sprite ($M = 2$) of sequence "Horse".

experiment the sprites (LR and SR) were generated and the background was reconstructed by re-projecting the sprite content into every single frame. The reconstructed scene shot has the same dimensions as the original one. Applying our SR technique we can achieve an averaged improvement of up to 2.3 dB. For most sequences it is more than 1 dB.

Sequence	"Stefan"				"Horse"		"Monitor"	
Trans. model	persp.		parab.		parab.		parab.	
Resolution	LR	SR	LR	SR	LR	SR	LR	SR
bPSNR in dB	25.51	27.18	26.27	28.63	28.04	29.58	33.05	34.00

Table 4.1: Average background PSNR in dB for re-projected frames from the LR and SR sprites of sequences "Stefan" (250 frames), "Horse" (150 frames), and "Monitor" (150 frames).

The curves of bPSNR values for every re-projected frame are given in Figures 4.13 and 4.14. Applying the parabolic image transformation model we obtain for sequence "Stefan" an average of 30.9 dB over the first 100 frames, which is just 0.74 dB below the value given in [165]. Note that the author applies an iterative SR approach of much higher complexity.

In order to give a better subjective impression several cropped regions of the LR sprite and SR sprite respectively are shown. In Figure 4.17 the suppression of aliasing along diagonal edges can be observed. The overcoming of the aliasing effect is even more impressive for high frequency textures like in 4.18. A disadvantage of the proposed approach for super-resolution sprite construction is the occurring *zipper effect*. Due to missing regularization and small registration noise, the pixels in the HR image may vary in their position. However, this effect does not strongly disturb the perceived image quality. The impression of higher sharpness outweighs the distortions (see sprite details in Figures 4.19 and 4.20).

Figure 4.16: SR sprite ($M = 2$) of sequence "Stefan".

4.6 Conclusions and Chapter Summary

After executing the experiment for several amounts of content overlap between adjacent frames and analyzing different video re-projection sizes for artificial and real digital video shots we come to the following conclusions:

- We confirm the conclusions drawn in [3] that super-resolution results decrease rapidly with the image magnification if no constraints are added. From a linear magnification of the video scaling every dimension by 3 (9 times of the image area) there is no gain in applying our SR technique. With our SR technique we propose a maximal linear magnification of 2.
- The objective gain is only an indicating measure for the video quality. Subjectively, aliasing errors and blurs are identified with much higher attention. Thus, in subjective evaluations SR techniques are often rated with a higher score.
- The more images overlap the better the SR result is. Especially for frames with enough spatial overlap into all panning directions we can gain more than 1 dB over LR sprite-based methods.

Unfortunately the comparison to other methods of super-resolution mosaicing is very difficult since many authors do not give any measurement figures [172]. Comparing "nice" images may be just a starting point but cannot be a proof for any proposed technique. As already stated, we nearly achieve the SR performance proposed in [165] by applying a SR technique of less computational complexity. Thus, SR methods are very helpful to improve the reconstruction quality of the static scene background. Applying the proposed SR technique the inherent quality limit of sprite-based re-projected image frames can be pushed towards higher levels in subjective and objective assessment.

In this chapter we proposed a new technique to create super-resolution sprites by mainly modifying the blending process of the LR sprite generation algorithm of Chapter 3. The described process is of low complexity and therefore simply applicable to any sequence. First, we introduced into the fundamentals of super-resolution image processing by depicting the

(a) LR sprite

(b) SR sprite $M = 2$

Figure 4.17: Cropped region of LR (a) and SR (b) sprite of sequence "Horse". (The image contrast is amplified for better visual comparison.)

(a) LR sprite

(b) SR sprite $M = 2$

Figure 4.18: Cropped region of LR (a) and SR (b) sprite of sequence "Stefan".

(a) LR sprite

(b) SR sprite $M = 2$

Figure 4.19: Cropped region of LR (a) and SR (b) sprite of sequence "Monitor".

(a) LR sprite

(b) SR sprite $M = 2$

Figure 4.20: Cropped region of LR (a) and SR (b) sprite of sequence "Monitor".

observation model and specifying the constraints for its inversion. Then we briefly gave an overview of today state-of-the-art super-resolution approaches, which aim to invert the observation model. After presenting an in-depth analysis on the main property to overcome spatial aliasing we introduced a technique for SR sprite construction based on minimal interpolation necessities. Finally, we gave experimental results utilizing both, synthetic and real image data.

Chapter 5

Multiple Sprites for Minimal Coding Costs

Few things are harder to put up with
than the annoyance of a good example.

Mark Twain (1835 - 1910), from 'Pudd'nhead Wilson'

The use of planar sprites for the summarization of video background objects has many advantages over other background representations. Mainly the simple image-based transformation model makes the utilization very attractive. In order to achieve a global invertible registration no knowledge of the internal camera parameters is necessary. In contrast to techniques generating cylindrical or spherical background panoramas we do not need to know the focal length or the position of the principal point, e.g., the intersection of the optical axis with the camera sensor plane. Thus, the derived transformation parameters (see Chapter 2) can easily and robustly be derived. However, we have already shown that planar background sprite techniques do not support the projection of 360 degree pans and tilts into one mosaic (Section 2.4). If the normal of the reference image plane and the direction of any pixel in an image to register exceeds 90 degree we will obtain a *degenerated transformation* [23]. But even if the degeneration does not appear, the geometrical distortions caused by the projection can increase the resulting sprite size tremendously. For many applications extreme large images are of no use since they require an essential part of the system's memory.

In this chapter we will discuss how the limitation of single sprite generation can be overcome by applying a multiple sprite construction strategy, which ensures an all-directions registration and keeps the memory usage as small as possible.

5.1 Geometric Distortions and Coding Costs

Due to projection properties of the perspective and parabolic transformation, parts of the mosaic content that are not close enough to the principal point of the sprite reference frame will be geometrically distorted. The distortion is highly dependent on the panning or tilting

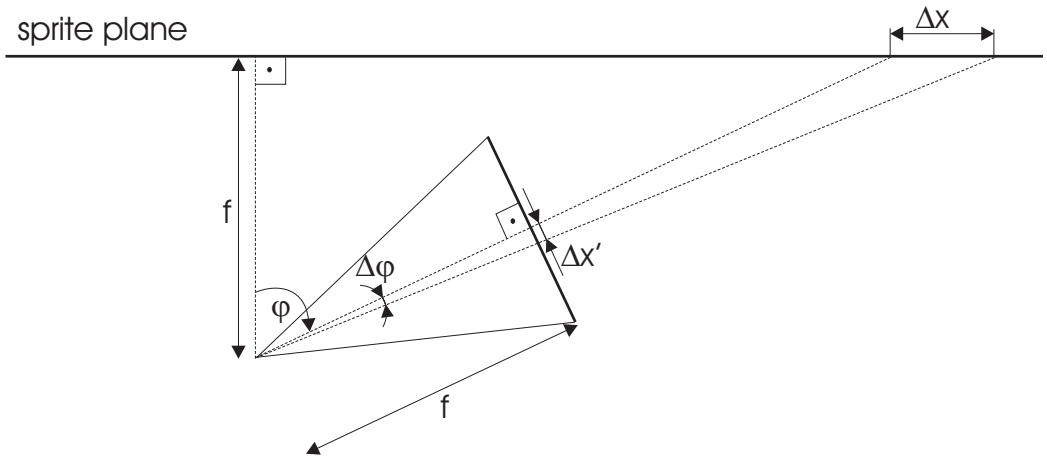


Figure 5.1: Principle of geometry-based image distortions caused by plane-to-plane projections with wide angles: The ratio $\frac{\Delta x}{\Delta x'}$ strongly depends on the rotation angle ϕ.

angle φ_y or φ_x. Thus, with increasing rotation angle of the capturing camera the projected background sprite dimensions will increase rapidly. In the next section we present an exact analysis of the introduced distortion for the one-dimensional projective space $\mathbb{P}^1$.

5.1.1 Limitations of Planar Image Mappings

For simplification of a mathematical analysis of the introduced geometrical distortion we just regard a horizontal camera pan about angle $\varphi = \varphi_y$. Using the intersection with the plane $y = 0$ we can proceed the analysis in projective space $\mathbb{P}^1$. The geometrical distortion introduced by the projective camera can now be described by the derivative dx/dx', where dx represents a differential distance in the sprite image while dx' represents a differential distance in the frame to register. In Figure 5.1 the principle of distortion is shown. We assume the focal length f for one pan to be constant. Approximating the derivative with the difference quotient we obtain

$$\frac{dx}{dx'} = \lim_{\Delta x' \to 0} \frac{\Delta x}{\Delta x'}. \tag{5.1}$$

Writing for Δx and $\Delta x'$

$$\Delta x = f\left[\tan(\varphi + \Delta\varphi) - \tan(\varphi)\right] \tag{5.2}$$

$$\Delta x' = f\tan(\Delta\varphi) \tag{5.3}$$

and replacing trigonometric identities we obtain

$$\begin{aligned} \frac{\Delta x}{\Delta x'} &= \frac{1}{\tan(\Delta\varphi)}\left[\frac{\tan(\varphi) + \tan(\Delta\varphi)}{1 - \tan(\varphi)\tan(\Delta\varphi)} - \tan(\varphi)\right] \\ &= \frac{1 + \tan^2(\varphi)}{1 - \tan(\varphi)\tan(\Delta\varphi)}. \end{aligned} \tag{5.4}$$

Forming the limit $\Delta x' \to 0$ can be represented by $\Delta\varphi \to 0$ and thus

$$\frac{dx}{dx'} = \lim_{\Delta\varphi \to 0} \frac{\Delta x}{\Delta x'} = 1 + \tan^2(\varphi). \tag{5.5}$$

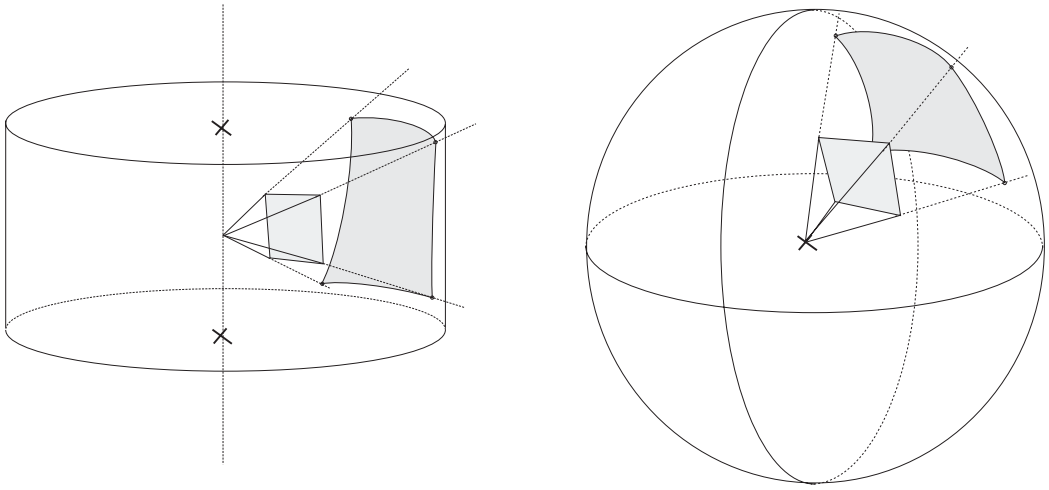


Figure 5.2: Projection of captured images onto a cylindrical (left) and spherical (right) surface. This type of sprite generation would overcome the angle-based registration limit, but can only be achieved conducting an exact camera calibration.

Since φ expresses the angle between reference frame and the actual frame to register, the distortion dx/dx' will increase towards infinity when $\varphi \rightarrow \pm 90°$. In that case we obtain so-called *degenerated* image transformations, which are of no use for sprite construction and its further applications. Choosing the optimal reference frame one sprite can only contain images captured by camera pans of *less* than 180°. As an alternative one could generate cylindrical or spherical (pan and tilt) sprites, as shown in Figure 5.2. Unfortunately, this process requires a quite exact estimation of the focal lengths of all frames in a shot. Hence, the sprite generation would be highly dependent on a good and reliable camera calibration technique.

However, applying a rough and low complex camera parameter estimation technique as proposed in Chapter 2 would be sufficient enough to estimate the panning and tilting angle φ_y and φ_x. Since this parameters are estimated using only the short-term homographies $\mathbf{H}_{n-1,n}$ for $n = 2, \ldots, N$, we can prevent the sprite construction algorithm from estimating *degenerated* frame-to-mosaic registration parameters.

5.1.2 Sprite Sizes Related to Memory Costs

The aim, we want to achieve, is not only the prevention of the estimation of degenerated image registration but also a minimization of the memory costs for sprite storage. Especially for super-resolution sprite processing the mosaics become very huge – e.g., 50 Mega-Pixels or more – for short shots in Standard Definition TV (SDTV) resolution. Hence, it is desirable to minimize the memory cost for sprite storage while keeping the reconstruction quality on the same level.

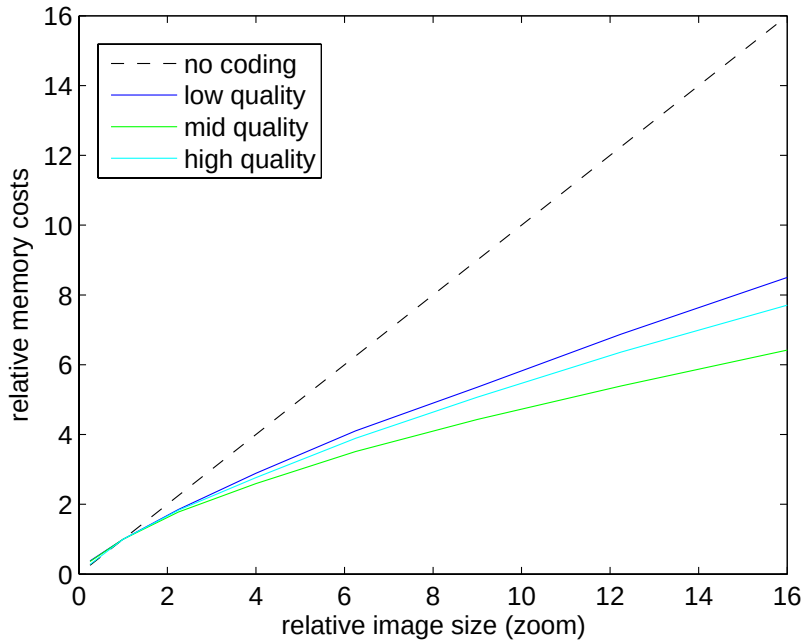


Figure 5.3: Relation between sprite image sizes and coding costs for sequence "Stefan" using different levels (QL=100, QL=60, QL=30) of JPEG coding. With this experiment we compare only the coding properties for standard zoom. Actually, for different reference frames the regions in the sprites undergo a full perspective distortion.

5.1.2.1 Still Image Coding - JPEG

In Equation 5.5 we see that the magnification Δx of the resulting sprite image rapidly grows with increasing panning angle. The dependence is highly nonlinear.

$$1 + \tan^2(\varphi) > 2 \qquad \text{for} \quad \varphi > 45^\circ$$

Thus, with growing angle the sprite image size will increase towards infinity. For uncoded sprite images the memory cost are linearly dependent on the image size. Lets assume the memory cost for any uncoded image I (dimensions $M \times N$) to be C_{Mem}. Then, the memory costs of any sprite image of size $sM \times N$ would become

$$C_{Mem,s} = s \cdot C_{Mem}. \tag{5.6}$$

Typically, images are stored and transmitted applying source coding to reduce spatial redundancies. Therefore, it would be interesting, what the dependence looks like in the coded domain. In order to compare the relative memory costs we compressed an sprite image from sequence "Stefan" at different resolutions using JPEG compression with different quality levels [153]. Figure 5.3 shows the results for $s = 0.5, \cdots, 16$. From the experiment the relation of memory cost $C_{Mem,s}$ and relative sprite size s can be expressed by

$$C_{Mem,s} = s^{\frac{\alpha}{2}} \cdot C_{Mem} \qquad \text{with} \quad 1.35 \leq \alpha \leq 1.55. \tag{5.7}$$

In the coded domain the memory cost does not increase linearly with the sprite size. This fact has to be taken into account when deriving a rule for the partition of sprites.

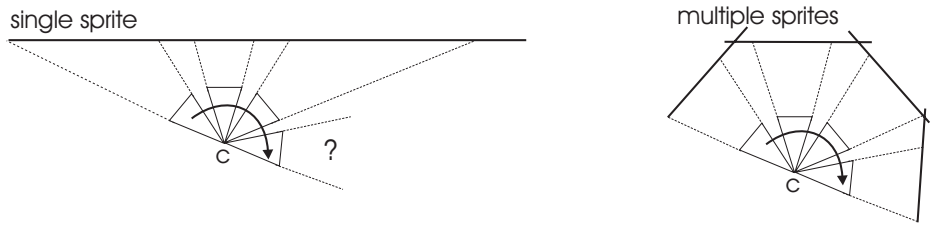


Figure 5.4: Principles of single sprite construction (left) and multiple sprite construction (right).

However, since projective distortions are more complex than a simple scaling of both image dimensions, like performed in the experiment, the actual curve for a certain image region will be closer to the linear dependency defined by Equation 5.7. Furthermore the curves will be dependent on the applied coding strategy. Thus, for further examination we will consider the linear dependency as basis, well knowing that this can only be an approximation. The merit of this approximation is the coding independent processing of the sprite partitioning.

5.1.2.2 Sprite Partition

The principle of multiple sprite generation is shown in Figure 5.4. If the sprite dimensions become to large the sprite will be subdivided into several sub-sprites. Those multiple sprites will then represent the background of a scene shot in an optimal way. Hereby, it is important not to allow image subsampling during sprite generation process. Hence, for the several sub-sprites we have to magnify the selected reference frame if necessary, so that no under-sampling occurs. It was shown that under-sampling can introduce significant aliasing errors, which can corrupt the registration accuracy. In the next section we will present a new and fast method for optimal multiple sprite generation and reference frame selection.

5.2 Multiple Sprite Construction and Reference Frame Selection

In [29] the authors propose a technique for the construction of multiple sprites with minimal coding costs based on the estimation of the size of the surrounding rectangle. In order to find the best solution, they determine the bounding box size for every pair of frames i, k being first and last frame of a sub-sprite while testing every reference frame I_r with $(i \leq r \leq k)$. Tracing back the cost values from the last frame one obtains the optimal sequence partitioning for sprite construction. However, the algorithm still works sequentially, i.e., all frames that compose a sub-sprite are always adjacent frames of the original shot. Additionally, the algorithm itself remains structurally complex even if the cost computation can be simplified.

In our approach we rely on a very comprehensible physical camera parameter-based sequence partitioning. If the camera angle between two views of a scene becomes too wide, i.e., the perspective distortion causes huge sprite image sizes, the overall angle (pan and/or

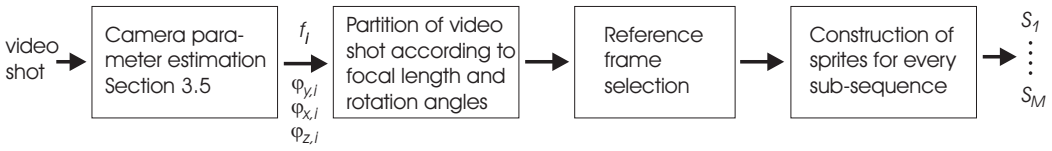


Figure 5.5: Flowchart of the proposed camera parameter-based multiple sprite generation technique.

tilt) will be parted into M angles of equal size. Thus, the partition algorithm yields in the calculation of M_{opt} minimizing a simple cost function depending on the camera parameter set. One advantage of our approach is the possible reordering of the frames. Therefore, unlike other methods, frames of different temporal positions can compose one sub-sprite.

A flowchart of the proposed technique for the generation of multiple sprites is given in Figure 5.5. First, we estimate the external and internal camera parameter as proposed in Section 2.5. Second, according to the panning and tilting angle the shot is divided in several sub-shots containing all images that build up one sprite. Third, for every sub-shot the reference frame is selected and a magnification is performed if necessary. Finally, each sprite is constructed using direct frame-to-mosaic registration as proposed in Section 3.3 and robust blending techniques as given in Section 3.4. The construction of super-resolution multiple sprites can be achieved as well as for the single sprite technique.

5.2.1 Shot Division

In order 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 $\Delta\varphi_{\max} = \max(\varphi) - \min(\varphi)$. 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.

$$C(\Delta\varphi_{\max}, M) = \sum_{i=0}^{M-1} f_{\max,i} \cdot 2\tan\left(\frac{\Delta\varphi_{\max}}{2M} + \frac{FOV}{2}\right) \tag{5.8}$$

Factor $f_{\max,i}$ describes the maximum focal length within the set of frames classified by a certain angle range

$$i \cdot \frac{\Delta\varphi_{\max}}{M} \leq \varphi - \min(\varphi) \leq (i+1) \cdot \frac{\Delta\varphi_{\max}}{M}, \quad \text{for} \quad i \in \{0, \ldots, M-1\}.$$

The field of view (FOV) is the average aperture angle for the same subset of frames. It can be computed from the focal length using

$$FOV = 2 \cdot \arctan\frac{W}{2f}, \tag{5.9}$$

where W is the width of image for panning angles in pixels. For camera tilt W has to be replaced by image height H. An example for subdivision along one dimension is given in Figure 5.6. After subdividing the sequence in one rotation plane (pan or tilt), we apply

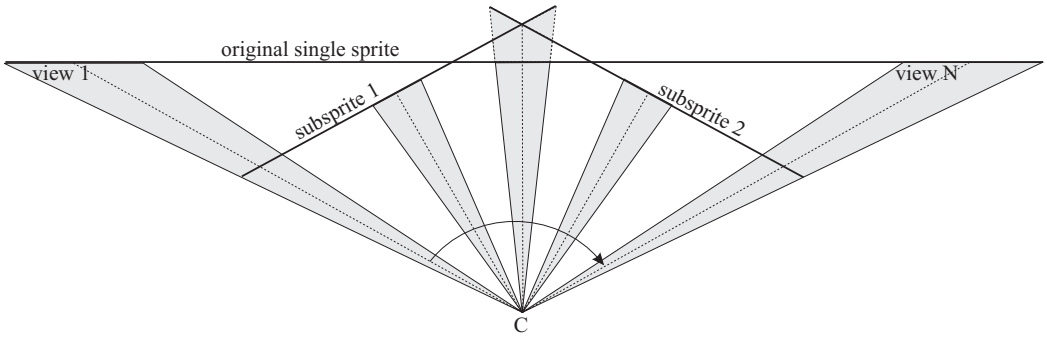


Figure 5.6: Partition of one sprite into two sub-sprites for constant focal length (constant FOV) along horizontal dimension. See that the overall size of the two sub-sprites is smaller than the original sprite due to geometrical distortions. The reference frame is chosen as the middle frame with respect to the view's absolute angle. The field of view (FOV) is marked as gray.

Equation 5.8 for the perpendicular plane as well. Thus, we handle the sequence division as a separable problem, which is only an approximation for the non-separable perspective mapping. However, for small angles $\Delta\varphi_{\max}/M$ the introduced error is negligible. Beside its low computational complexity this approach has the advantage that frames located relatively close in space are classified into on sub-sprite. Thus, the single sequences do not necessarily contain temporal consistent videos.

5.2.2 Reference Frame Selection and Spatial Magnification

As reference image we select the frame I_r with an rotation angle φ closest to the center angle of the shot. Since this frame is not the frame with the biggest zoom, the reference frame coordinate system is scaled by

$$s = \frac{f_{\max,i}}{f_r}, \tag{5.10}$$

where $f_{\max,i}$ represents the maximum focal length within a subsequence. Thus, we prevent resolution degradation and aliasing errors due to subsampling of images during the warping process.

5.2.3 Advantages over other Approaches

The two main advantages of the proposed multiple sprite approach are the temporal non-linear subdivision of a video shot and the low complexity of the whole algorithm. In cases where a camera pans forth and back within one shot it is possible that the first and last frames of this shot are classified in order to be registered into one sprite, while the other frames construct another sprite. Other approaches as in [14] and [29] allow only a sequential classification of frames into a background sprite.

Since the assignment of images to different sprites is based on the fast camera estimation algorithm, the complexity of this multiple sprite generation approach is very low. Actually,

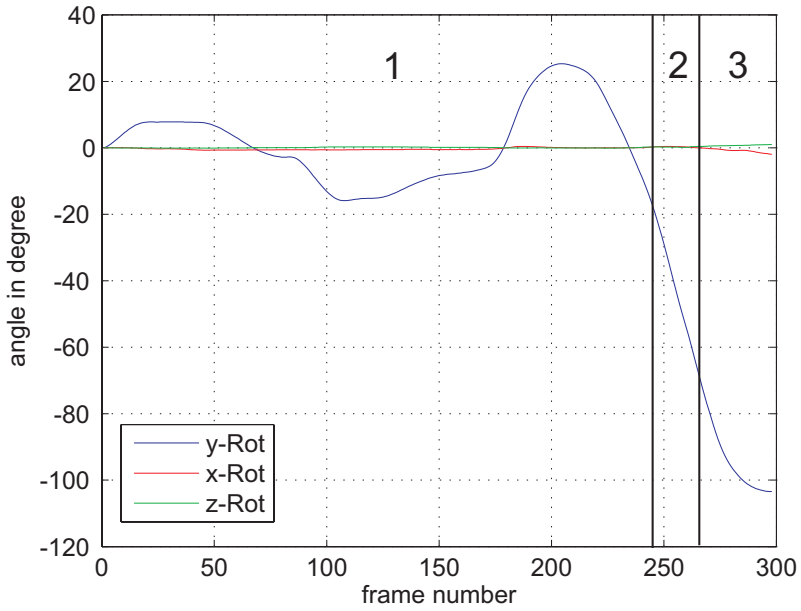


Figure 5.7: Plot of absolute angles for pan, tilt, and roll of sequence "Stefan". A subdivision of the sequence is executed along the panning angle φ_y (y-rotation). The boundaries of the sub-shots are drawn in the diagram with vertical lines. The resulting three sprites are: Sprite 1 - frame 1 to frame 245; Sprite 2 - frame 246 to 262; Sprite 3 - frame 263 to 300.

the computational load for camera calibration and sprite division is only a fractional amount of the load for exact registration and blending.

5.3 Experimental Results: Quality and Coding Costs

We examined the multiple sprite approach for several real world recordings. In order to assess the results, we will show that the reconstruction quality of the background improves and/or the consumed memory for the multiple sprites significantly reduces compared to the single sprite approach. Since this optimization also applies for coded sequences, we will finally present the coding results for the background using H.264/AVC - intra coding for the sprite image.

5.3.1 Multiple Sprites and Reconstruction Quality

For the background sequence "Stefan" (300 frames, *standard intermediate format* SIF, 30 fps) and sequence "Biathlon" (200 frames, *common intermediate format* CIF, 25 fps) we applied the presented multiple sprite approach. For "Stefan" the optimal number of sprites is 3 following Equation 5.8. Only the panning angle φ_y changed significantly. For natural scenes it is very common that mainly pan occurs because this is the usual way humans observe their surroundings. Thus, we obtain a horizontal arrangement of the resulting sub-sprites. Figure 5.7 shows the estimated Euler angles for the whole shot. The curve for angle φ_y induces the subdivision of the shot. The boundaries of the sub-shots are marked

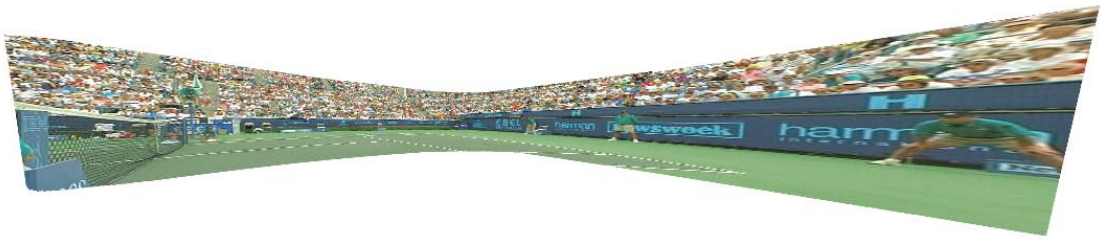


Figure 5.8: Single sprite approach - "Stefan" (original size: 4339x953 pel, frame 1 to 300, reference frame number: 254).

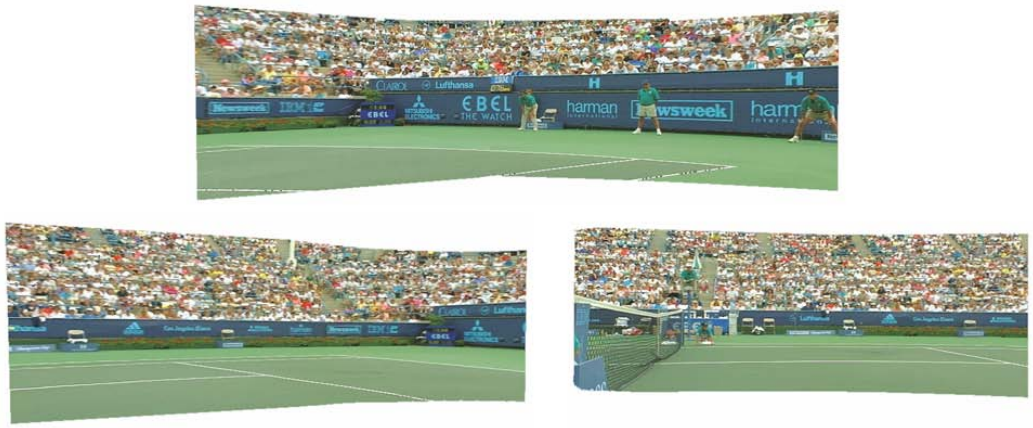


Figure 5.9: Multiple sprite approach - "Stefan" (of original size: 1010x317 pel, frame 1 to 245 - upper; 799x317 pel, frame 246 to 262 - lower left; 679x261 pel, frame 263 to 299 - lower right).

with solid vertical lines. For visual comparison we show the resulting sprites for the single sprite approach in Figure 5.8 and the multiple sprite approach in Figure 5.9. In order to blend out the foreground object one should use as many redundant information as possible. Hence, for blending the final multiple sprites we register also temporal neighboring frames that normally belong to another sub-sprite. The analogous curves and sprites for sequence "Biathlon" are presented in Figures 5.10, 5.11, 5.12, and 5.13. Here, the number of sprites to minimize the coding costs is estimated to 4.

In Table 5.1 we compare the reduction of the memory costs for the uncoded sprites using the single and the multiple sprite approach. Additionally, the reconstruction quality for the video background was measured. The curves for both sequences are shown in Figures 5.14 and 5.15. We found out that for all sequences we can reduce the memory size significantly while even improving the reconstruction quality. There are two reasons for the improvement of the background PSNR. First, due to the generation of new sprites slight, changes in the background are covered. Therefore, small motions of those objects can be reproduced closer to reality switching from one sub-shot to another. Second, derivations in the camera capture setup from the ideal case, e.g., only rotational camera motion plus zoom, can lead to registration errors. These errors increase with growing panning or tilting angle. Since

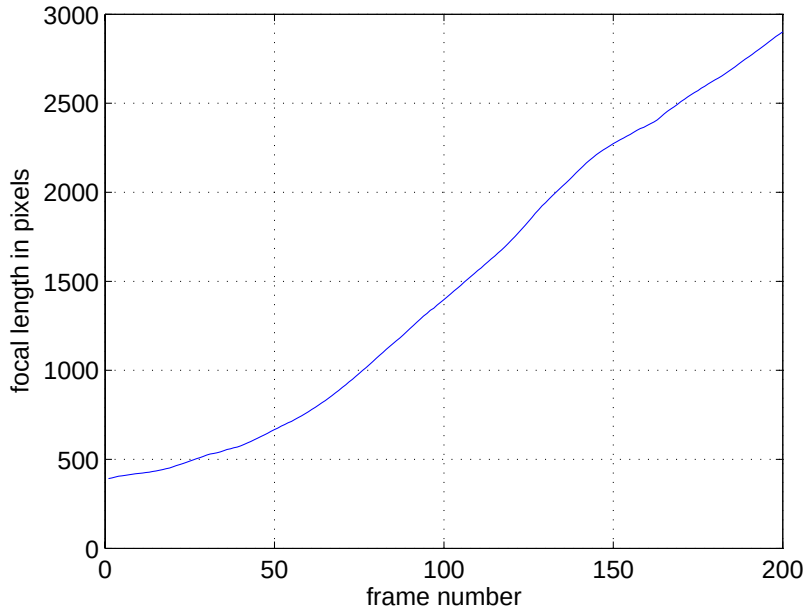


Figure 5.10: Focal length estimation for sequence "Biathlon". While panning left, the camera follows the sportsman and changes from wide angle to huge zoom.

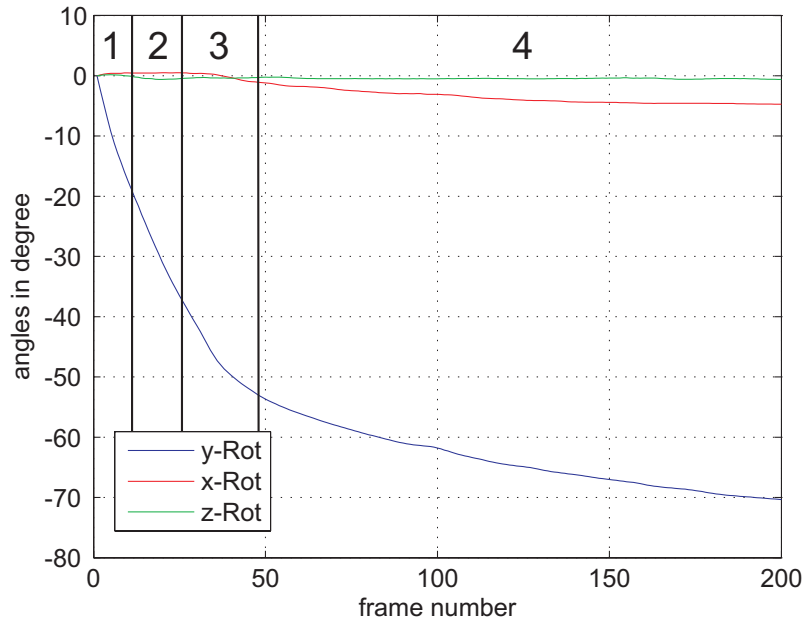


Figure 5.11: Plot of absolute angles for pan, tilt, and roll of sequence "Biathlon" (200 frames). A subdivision of the sequence is executed along the panning angle φ_y (y-rotation). The boundaries of the sub-shots are drawn into the diagram with vertical lines. The resulting four sprites are: Sprite 1 - frame 1 to frame 10; Sprite 2 - frame 11 to 23; Sprite 3 - frame 24 to 47; Sprite 4 - frame 48 to 200.

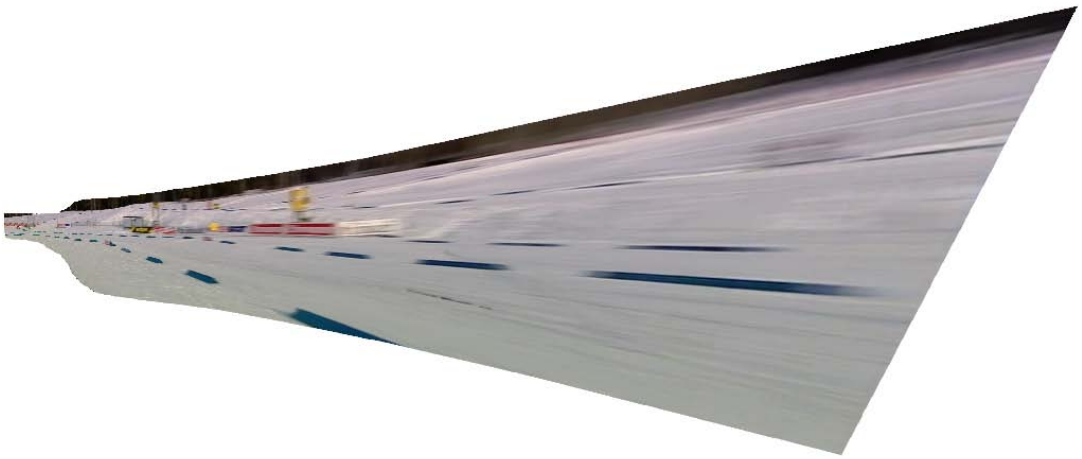


Figure 5.12: Single sprite approach - "Biathlon" (original size: 3246x1401 pel, frame 1 to 200, reference frame number: 36).

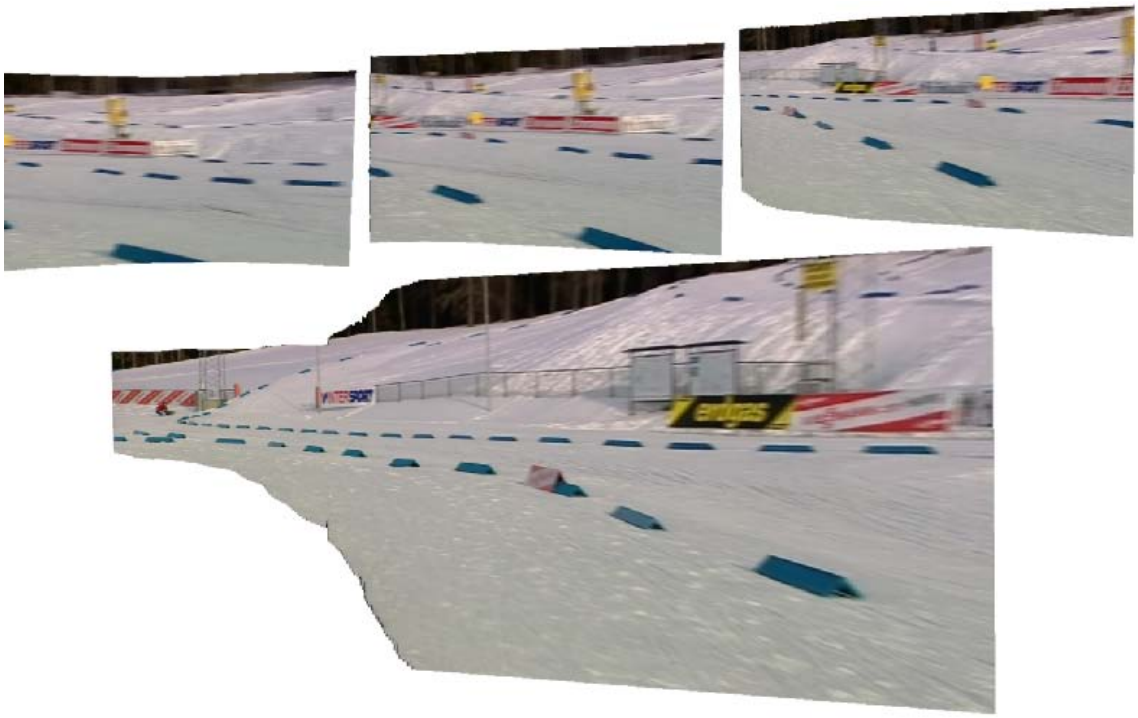


Figure 5.13: Multiple sprite approach - "Biathlon" (of original size: 529x310 pel, frame 1 to 10 - upper left; 539x328 pel, frame 11 to 23 - upper middle; 600x357 pel, frame 24 to 47 - upper right; 1291x727 pel, frame 48 to 200 - lower).

multiple sprites do not allow huge pans, the registration error will be kept small. Especially for sequence "Biathlon", which is a left pan plus heavy zoom, the memory saving is about 66.8% while increasing the background reconstruction quality by 4.22 dB.

Sequence	"Stefan"		"Biathlon"	
Sprite approach	single	multiple	single	multiple
Memory size in kByte	6064.98	1104.11	6666.35	2215.64
Memory savings using multiple sprites	81.8%		66.8%	
Average reconstruction quality	25.51 dB	25.77 dB	30.21 dB	34.43 dB
Quality gain using multiple sprites	0.26 dB		4.22 dB	

Table 5.1: Comparison of memory sizes and reconstruction quality for uncoded single and multiple sprites of sequences "Stefan" and "Biathlon". The images are stored in YCbCr (4:2:0) format, where the color components are subsampled by 2 in every dimension.

5.3.2 Background Video Coding

The results of memory reduction for uncoded video scenes are not compelling since the spatial redundancy in the sprite image was not reduced at all. One could argue that due to non-rectangular sprite images, all regions containing no information (border regions) can be coded very efficiently. Thus, the gain by applying almost rectangular multiple sprites could be neglected. We will show that this is not the case. Even by compressing the images strongly, we obtain both, memory or bit rate savings and reconstruction improvement compared to single sprite coding.

We coded the sprite images with different quantization parameters (QP) applying H.264/AVC intra video coding [160]. It is more efficient than usual JPEG or JPEG2000 still image codecs. The QPs were varied from 28 to 40. Additionally, the perspective transformation parameter set was transmitted using 3 bytes per parameter. In [17] the authors estimate the number of bytes per parameter 1.5 using sophisticated compression methods. Since the portion of the parameter set in the overall bit rate is rather small, we apply no compression. After the reconstruction of the video background using the compressed images, we measured the objective background quality using an auxiliary foreground mask. The rate-distortion measures for different QPs are given in Figures 5.16 and 5.17.

Finally, we exemplarily present subjective results for both sequences at equal bit rates. See Figures 5.18 and 5.19 for visual comparison. Especially for sequence "Stefan" the subjective quality improvement is very demonstrative since the objective improvement of 0.46 dB does not suggest such a big difference.

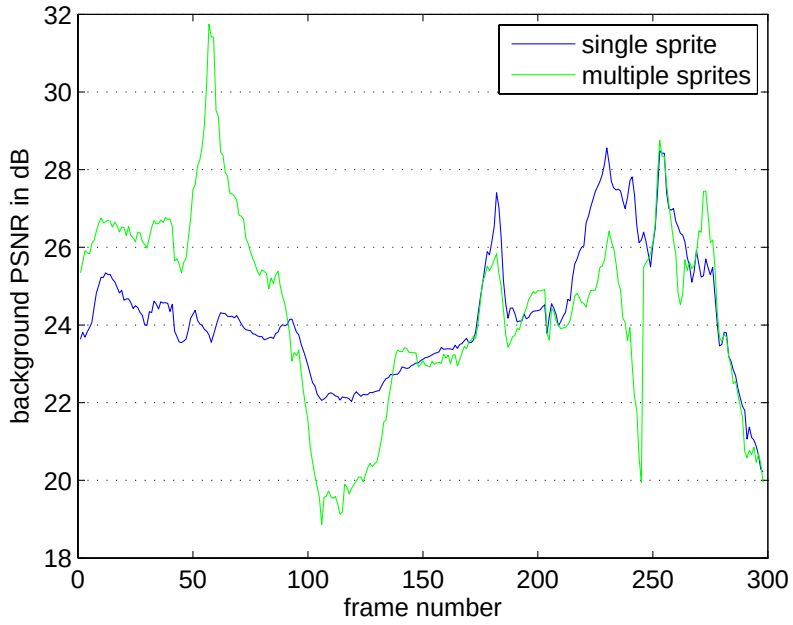


Figure 5.14: Per frame reconstruction quality for uncoded sprites of sequence "Stefan".

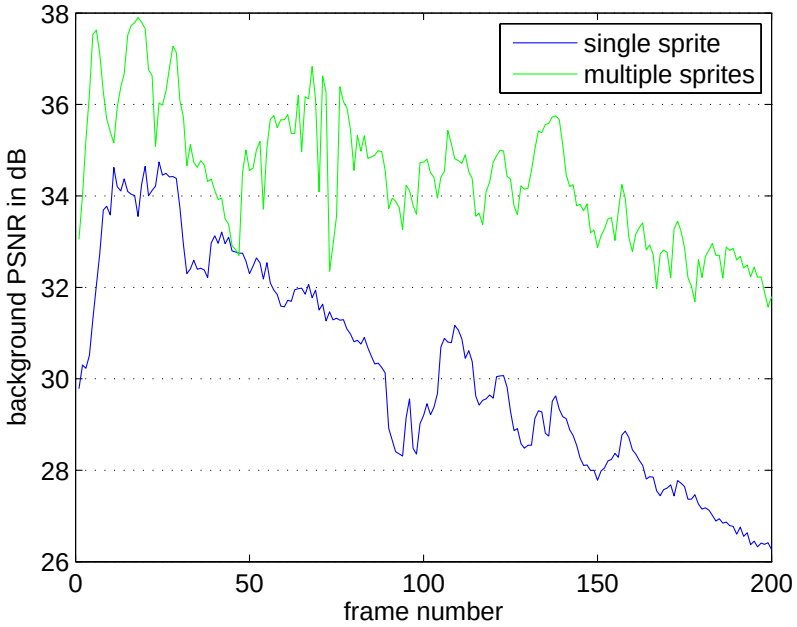


Figure 5.15: Per frame reconstruction quality for uncoded sprites of sequence "Biathlon".

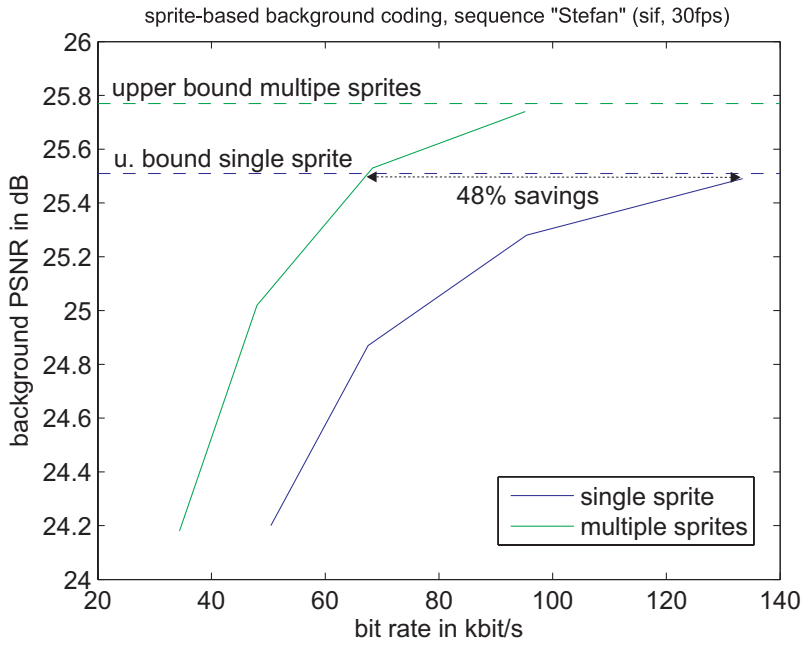


Figure 5.16: Background coding results for sequence "Stefan". The upper limits for single and multiple sprite approaches are marked with the dashed lines. They can be achieved by applying lossless coding. The bit rate savings are up to 48% at the same quality level.

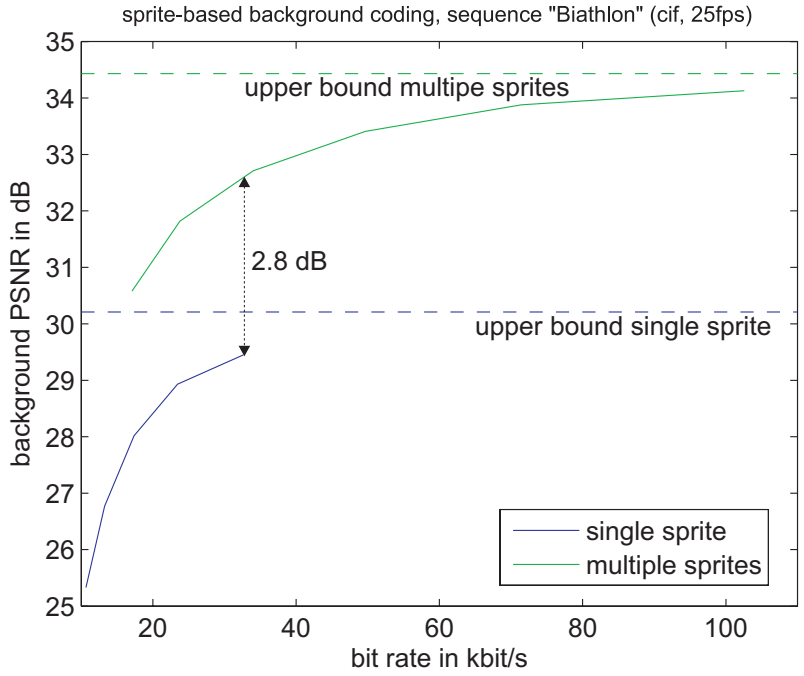


Figure 5.17: Background coding results for sequence "Biathlon". The upper limits for single and multiple sprite approaches are marked with the dashed lines. They can be achieved by applying lossless coding. The quality gain using multiple sprites is up to 2.8 dB at the same bit rates, which is strongly noticeable at this absolute quality level.

(a) single sprite - 95.42 kbit/s (b) multiple sprites - 95.19 kbit/s

Figure 5.18: Subjective comparison for frame 120 of sequence "Stefan" at equal bit rates.

(a) single sprite - 17.39 kbit/s (b) multiple sprites - 17.11 kbit/s

Figure 5.19: Subjective comparison for frame 66 of sequence "Biathlon" at equal bit rates.

5.4 Chapter Summary

In this chapter we have presented a new approach to overcome one of the main limitations of planar background sprite generation. Due to the projective properties of the perspective image transformation, the sprite image sizes increase nonlinearly with growing panning or tilting angle during the scene capture. This causes inappropriate memory costs for storing the sprite image. Instead of applying more complex mappings like projections onto cylinders and spheres we suggest to split the shot into several sub-shots. Each of them will be used to generate a separate sprite.

The multiple sprites approach combines two advantages. First, the memory cost for uncoded sprite images will be minimized. Second, the technique selects an useful reference frame for every sub-sprite. Moreover, due to possible background refresh when switching between the sprites for background reconstruction, we can even improve the re-projection quality. Also the number of globally registered frames will be kept small and in this way registration errors can be limited.

In the first part of this chapter we introduced the problem of nonlinearly increasing sprite dimension and derived an approximated formula for its computation. Then, we proposed a method for angle-based shot division in order to minimize the memory costs for an uncoded sprite. The technique mainly relies on the roughly estimated camera parameters of Section 2.5. Finally, we examined the multiple sprite technique for recorded real world scenes and showed that this method significantly reduces the coding costs for storage and transmission of the scene background.

Chapter 6

Sprites for Motion-based Object Segmentation

Never mistake motion for action.

Ernest Hemingway (1899-1961)

In this chapter we concentrate on the automatic segmentation of moving objects in a video shot. We will show that sprites are very helpful to perform the segmentation since they inherently represent a stabilized long-term background estimation containing all rigid objects, i.e., all objects that do not move camera-independently. The segmentation of moving objects is very important in many aspects of multimedia processing. On the one hand, it can be seen as application itself. For many tasks in image and video editing it is necessary to extract meaningful objects of a scene, which can be manipulated and combined arbitrarily. On the other hand, object segmentation is a very essential preprocessing step of a wide range of video processing and communication techniques. In second-generation video coding [148] as standardized in MPEG-4 [101] the generation of video object layers and video object planes (VOP) is a premise for successful coding. But also more recent applications and video analysis tools, such as content-based video retrieval, automatic video annotation, video object recognition (e.g., person detection), and video genre classification require the presence of segmented objects. Despite of the huge efforts that have been made to investigate segmentation methods it is still an error-prone task. Therefore, results of different approaches vary significantly depending on the video content and additional constraints.

In the following section we will give a short survey on methods and techniques for moving object segmentation. Then we will introduce two approaches for object segmentation using global motion compensated change detection and background sprites. The first approach is based on compensated long-term frame differences applying *markov random fields* (MRF) for estimation stabilization. The second approach uses a logical combination of long-term and short-term frame differences in order to extract the moving foreground objects. Both approaches use change detection algorithms as base techniques. Finally, we will present segmentation results and compare it to other motion-based segmentation techniques.

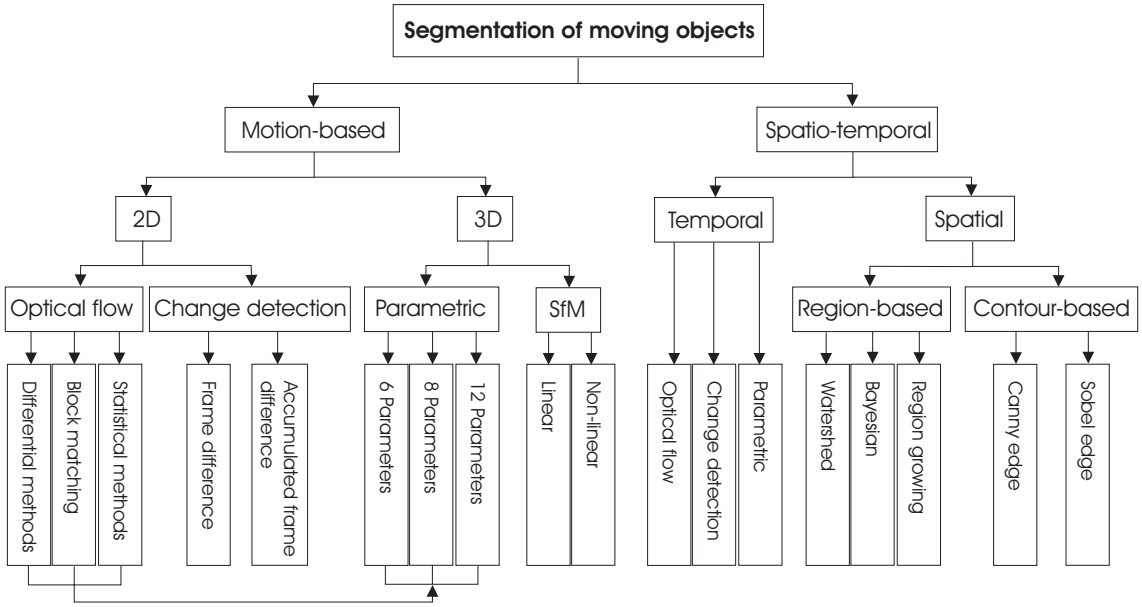


Figure 6.1: Classification of moving object segmentation approaches. [166]

6.1 Moving Object Segmentation: An Overview

The classification of segmentation techniques for moving objects in video scenes varies significantly in the literature, e.g., in [86] or [147]. We follow the classification scheme proposed by [166]. It aims to give a complete survey of motion-based techniques. A schematic overview of main segmentation approaches for moving objects is shown in Figure 6.1. In the schema the authors divide the object segmentation into motion-based segmentation and spatio-temporal segmentation. While the former class of segmentation techniques relies only on the temporal changes, the latter one combines both spatial image information, e.g., texture, and temporal motion information. The segmentation methods applying only motion-based techniques are subdivided into two groups: 2D motion, referring mainly to translational motion of objects or object parts and 3D motion, describing objects utilizing higher order motion models or *structure from motion* (SfM) 3D scene analysis.

Using only motion-based segmentation tends to result in insufficient spatial segmentation, i.e., parts of an object may not be detected. Over-segmentation, i.e., the separate classification of image regions belonging to the same object, may also occur. However, the computational load is smaller since no additional spatial analysis is applied. A simplified block scheme of motion-based segmentation is presented in Figure 6.2. One common method to achieve 2D motion-based segmentation is the calculation of the *optical flow* [69], [118]. In this type of segmentation the *dispacement* vector for every pixel or preselected image region between two frames is determined. The objects are finally segmented due to a classification of the displacement vectors.

Another often used type of segmentation is the *change detection*. It is especially useful if only a bi-segmentation, i.e., a segmentation in two classes, foreground and background,

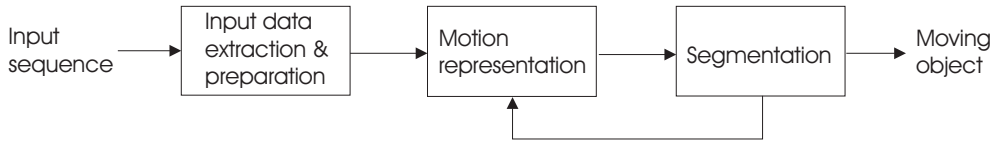


Figure 6.2: Simplified scheme of motion-based segmentation. [166]

is necessary. It is mainly based on a robust estimation of the background and is applied on the *frame differences* FD between the background estimation and any original frame.

$$FD(x,y) = I_k(x,y) - \hat{I}_{back}(x,y) \tag{6.1}$$

In many cases $FD(x,y)$ is applied on a supporting region, e.g., a window, rather than on the single pixel. For segmentation it is compared to a threshold T_{ch}. Usually, the threshold is based on statistical measures and assumptions [30]. Since it is one of the critical parameters of such segmentation methods, it has to be chosen very carefully. It is usually computed adaptively by including additional information of the sequence. Aach et al. [1] formulated change detection as a Bayesian problem and presented an adaptive estimation of the thresholds while incorporating pixel neighborhood assumption using *Markov random fields* (MRF). We will present two segmentation approaches which mainly use change detection fundamentals in order to extract the moving objects. A block diagram of common change detector is depicted in Figure 6.3. Using prior mask information and additional knowledge the threshold is calculated. Finally, holes are filled and small regions are removed applying morphological operations on the binary mask.

The segmentation based on higher order motion models and SfM techniques is a good method to differentiate several rigid objects moving independently. A motion of the camera, however, is useful to classify all regions that are fixed in the scene. Segmenting every object according to the parametric description of its motion is the fundamental idea for 3D motion-based segmentation. Especially for SfM methods one has to recover the 3D geometry of the scene [147]. Unfortunately, the robust estimation of motion parameters and their discrimination is computationally quite complex. It can often only be achieved

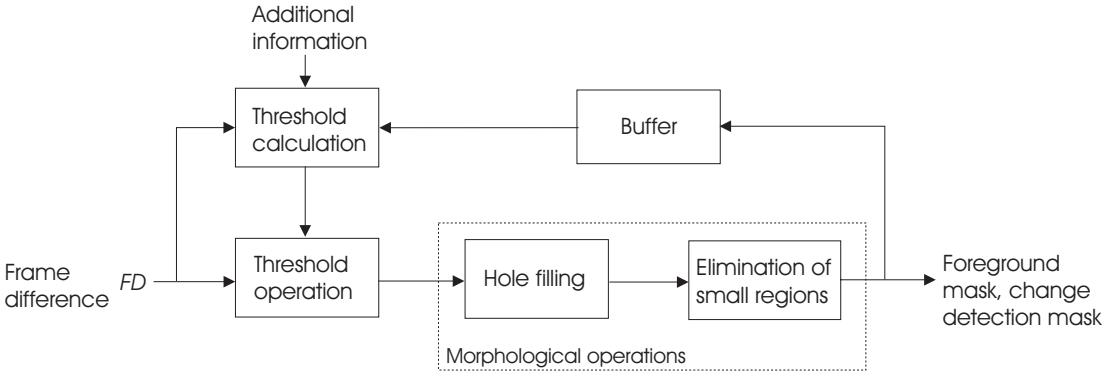


Figure 6.3: Block diagram of a typical change detector.

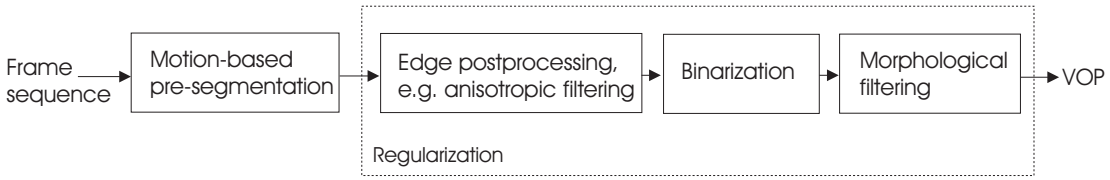


Figure 6.4: Block diagram of spatio-temporal segmentation methods.

for feature points and thus makes a complete object segmentation without using spatial information very difficult.

Applying spatial-temporal segmentation methods promises an improvement of the segmentation quality [166]. Since it marks a fusion of available temporal and spatial information, the segmentation process is performed more robustly and object borders are determined more exactly. In scenes where the foreground objects do not move for while, e.g., sports recordings, some motion-based segmentation approaches fail in discriminating the objects from the background. Here, the utilization of spatial segmentation techniques can reduce the errors. A schematic flow chart of typical spatial-temporal segmentation methods is given in Figure 6.4.

Other methods for reliable video object segmentation that usually rely on previously generated foreground masks are referred to as *tracking methods*. A consistency check between already segmented object planes in previous frames and the VOPs of the actual frame can decrease the probability of false positives and false negatives detection. Thus, the segmentation quality can furthermore be improved. For tracking of objects and regions Kalman filters [12] or particle filters [53] are often used. But also Active Contour models and feature-based trackers can be found prevalently in the literature.

The advantage of using background sprites for video object segmentation is easy to see. In parts of typical scenes where the foreground objects are motionless or moving only slightly simple motion-based techniques cannot detect these objects due to small frame differences between adjacent frames. Costly spatio-temporal methods or long-term tracking filter have to be applied. In order to keep this additional processing step simple, sprites can be used to estimate a background image for every frame free of arbitrarily moving objects. Since sprites provide a long-term video memory, we use all possible frame contents to estimate the background. In the next sections we will present two methods for moving foreground segmentation. Here, we are only interested in bi-segmentation, i.e., the separation of the foreground from the background. Both techniques are based on change detection. The first method additionally applies a MRF-based tracking method using only the previously generated change mask, while the second applies spatio-temporal refinement.

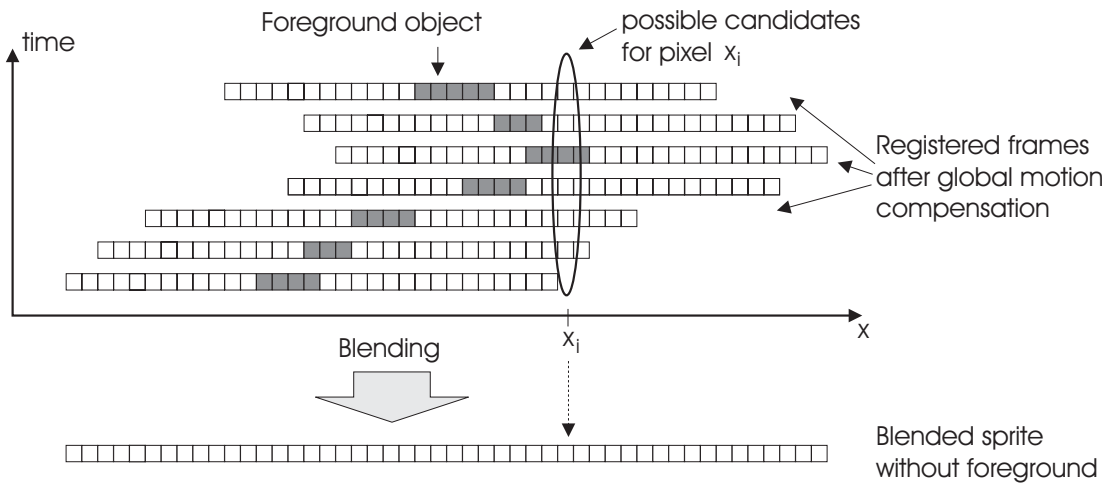


Figure 6.5: Principle of blending global motion compensated frames after registration. The blending process is depicted for one horizontal line $y = y_k$ of the mosaic. In many cases the occluded regions will be revealed in several frames. Using robust median filtering or correlation analysis reliable pixel candidates can be found.

6.2 Foreground Removal and Video Background Estimation using Sprites

The main approaches for robust blending of frame content into the sprite were already presented in Section 3.4. At this point, we will briefly discuss the problems and drawbacks we are confronted with. The principle of robust blending for on horizontal line of the sprite is shown in Figure 6.5.

Estimating the frame background for a scene using sprite technology carries along two major problems. First, small motions of semantically classified background elements, e.g., leaves, clouds, audiences, etc., are represented in a "frozen" form in the mosaic. We referred to that phenomenon as irrelevancy reduction. Since for the calculation of frame difference these movements are not considered, those elements can be classified as foreground. It is very unlikely that the differences will be modeled as image noise. Hence, the segmentation tends to classify additional parts of a frame into the foreground mask. Because of the small sizes, they can mainly be removed by morphological operations.

The second type of problems is connected with the registration accuracy. Re-projecting the sprite content into the frame coordinate system can result in small background displacements. Non-robust change detectors can mark pixels at edges and high frequency textures as changed. Morphological filtering and region-based change detection can help to remove those detection errors. High accuracy registration achieved by higher order image transformation models and the utilization of multiple sprites are also helpful to keep the percentage of such detection errors low.

(a) original (b) sprite re-projection (c) luminance difference

Figure 6.6: Frame difference for frame 42 between the original image and the re-projected sprite content of sequence "Race01" (544x336, 25 fps). The difference image is the base for the long-term change detection technique.

(a) original (b) short-term compensation (c) luminance difference

Figure 6.7: Frame difference for frame 42 between the original image and the short-term compensated (using frame 43) of sequence "Race01" (544x336, 25 fps). The difference image is the base for the short-term change detection technique.

6.3 Approach 1: MRF-based Change Detection for Moving Object Segmentation

Figures 6.6 and 6.7 depict the base operation for change detection algorithms. In Figure 6.6 the luminance difference for sprite-based background estimation is shown. On the one hand, one can see that in regions of the foreground objects the energy of the difference image is significantly increased. This is a benefit of using long-term change detection over short-term change detection techniques of Figure 6.7, where the difference image contains double artifacts of the foreground images. On the other hand, the difference energy for strong edges in the background is much higher for the sprite-based compensation due to registration errors.

In our first segmentation approach we will only use long-term memory-based change detection. In order to stabilize the detected foreground mask we will incorporate a temporal consistency constraint using the previous mask as a state map. Thus, the state of any pixel in the actual map – marked as changed or unchanged – is dependent on the states of neighboring pixels in the previous map plus the frame difference energy. We can regard the change mask as Markov random field (MRF) where the probability of the mask's pixel state depends on the state of the influencing entities and its probabilities.

6.3.1 Change Detection Approach

The change detection approach is a simplified version for Bayesian change detection related to the techniques proposed in [1] and [30]. Since it is not the scope of this thesis to do in-depth research of segmentation methods, we renounce to provide statistical fundamentals for this approach. We consider significance-based change detection for a block of pixels centered at every image pixel of $I_n(x, y)$. A pixel is recognized as changed if the block pixel difference energy $E_D(x, y)$ exceeds a certain threshold τ_c

$$E_D(x,y) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \left(I_n(x+u, y+v) - \hat{I}_{n,back}(x+u, y+v) \right)^2 > \tau_c. \tag{6.2}$$

In Equation 6.2 the block size is $(2k+1) \times (2k+1)$. $I_n(x, y)$ represents the actual frame while $\hat{I}_{n,back}(x+u, y+v)$ represents the background estimation from the sprite image. Using a block difference is very robust against noise in the difference image, first notably spike noise. The threshold τ_c is estimated automatically. We write

$$\tau_c = \tau_{static} - \tau_{consistency}. \tag{6.3}$$

τ_{static} represents the threshold derived from the pure frame differences while $\tau_{consistency}$ is a value dependent on the state of the previous change mask. The first value is estimated from the standard derivation of the absolute difference image noise for regions were only background objects appear. Here, we assume again the majority of the image not being covered by moving foreground objects. Thus, the standard derivation can be computed by

$$\begin{aligned} s_D &= \underset{\forall\, x,y}{\text{median}} \left(D(x,y) \right) \\ \text{with} \quad D(x,y) &= \left| I_n(x,y) - \hat{I}_{n,back}(x,y) \right|. \end{aligned} \tag{6.4}$$

τ_{static} is then set to

$$\tau_{static} = (2k+1)^2 (\alpha s_D)^2. \tag{6.5}$$

Coefficient α represents the inlier for absolute difference noise. Assuming a 4σ-neighborhood α is set to 4. The second threshold $\tau_{consistency}$ is computed using the previous mask as state map. It is determined according to the number N_c of pixels in a $(2k+1) \times (2k+1)$ spatial neighborhood that were marked as *changed*. A large number of *changed* pixels in the previous mask lowers the overall threshold τ_c. A schematic picture of the influence of the states is given in Figure 6.8. $\tau_{consistency}$ can be denoted as

$$\tau_{consistency} = \frac{1}{2} N_c (\alpha s_D)^2. \quad N_c \in \{0, 1, \ldots, (2k+1)^2\} \tag{6.6}$$

Therefore, if the neighboring pixels in the previous mask mark a foreground object the probability for the actual pixel to be a foreground pixel is much higher.

For color videos the above described classification in foreground and background pixels can be processed independently for every of the three color channels in the RGB color space.

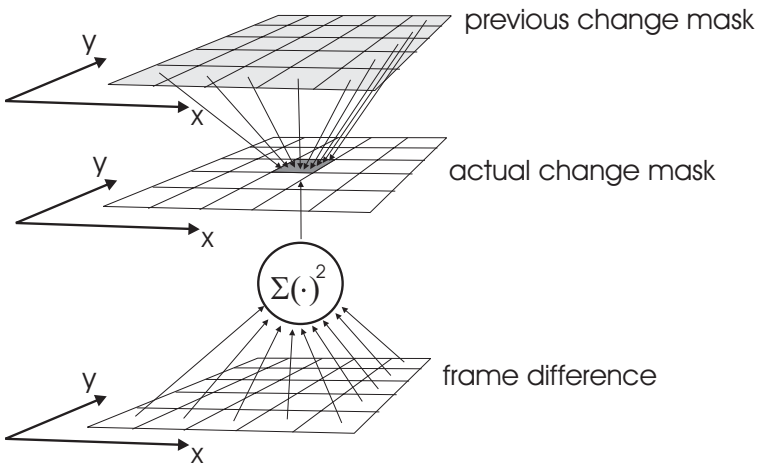


Figure 6.8: Influences of previous states of the change mask and the frame difference on the decision for the actual change mask. The decision process can be represented by a Bayesian network or Markov random field.

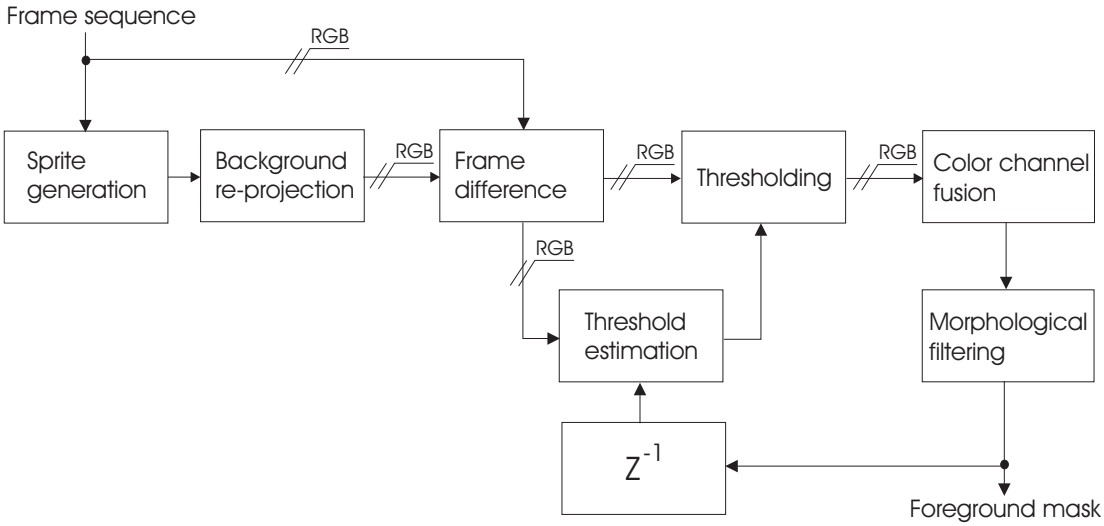


Figure 6.9: Flow chart of the MRF-based segmentation technique using change detection.

Fusing the results by applying a *majority decision*, i.e., a pixel is marked as foreground if it is a foreground pixel in at least two color spaces, yields the final change mask. Subsequently, the object boundaries are straightened by median filtering and small objects are removed using morphological filter technique. A block diagram of the overall MRF-based segmentation process is shown in Figure 6.9.

6.3.2 Segmentation Results

Assessing the quality of segmentation algorithms usually requires the presence of *ground truth* data. Recently, quality assessment schemes without any ground truth have been proposed [36] that also reflect a perceptual measure. But these metrics are not widely used among researchers. Therefore, we use only well-known pixel measure. Unfortunately, ground truth data for panning or tilting camera sequences are rare. In order to obtain meaningful

results manually generated ground truth has to be used. To keep the operating expense as low as possible only for sample frames the foreground object masks was generated. For instance, for sequence "Biathlon" (200 frames) the segmentation mask was generated for frame 1 to 10 and frame 101 to 110. Thus, we approximate the overall segmentation result by evaluating 20 frames (10%) of the whole sequence.

We classify four types of pixels to characterize the segmentation quality. The true positives (TP) give the number of correctly detected foreground pixels. The false negatives (FN) represent the number of pixels that are falsely marked as background and false positives (FP) are pixels falsely detected as foreground. The rest of the image is marked as true negatives (TN), pixels that are correctly detected as background. Based on the above described pixel numbers, different evaluation measures can be defined. To describe the accuracy of the segmentation, we utilize three measures: *precision* P, *recall* R, and the *F-measure* F. The precision P is the ratio between the correctly foreground detected pixels and all pixels detected as foreground.

$$P = \frac{TP}{TP + FP} \tag{6.7}$$

The recall R is the ratio between the correctly detected foreground pixels and all relevant foreground pixels.

$$R = \frac{TP}{TP + FN} \tag{6.8}$$

Since precision and recall can be weight out against each other, i.e., one measure can increase by decreasing the other using dilatation or erosion filter, the F-measure was introduced to equally combine precision and recall. It can be regarded as the *harmonic mean* between both measures.

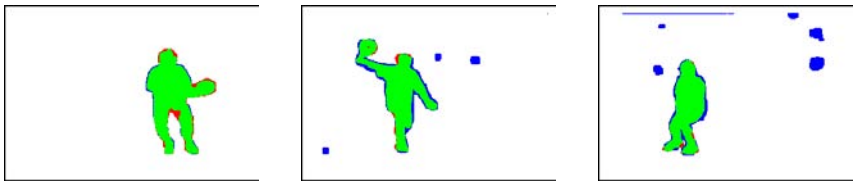
$$\begin{aligned} \frac{1}{F} &= \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right) \\ F &= \frac{2 \cdot P \cdot R}{P + R} \end{aligned} \tag{6.9}$$

Results for several frames of pixel classification, i.e., TP, FP, FN, and TN, are given in Figures 6.11 and 6.12 for sequence "Stefan" and sequence "Biathlon", respectively. The applied color scheme for the classification is explained by Figure 6.10.

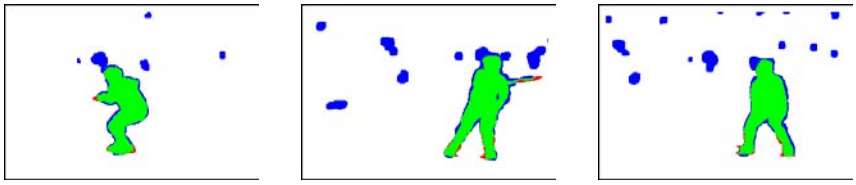
One can see that the algorithm tends to have a high recall and a lower precision. This can be explained by the errors that result from differencing the long-term sprite background

		ground truth	
		foreground	background
foreground segmentation	positive	TP	FP
	negative	FN	TN

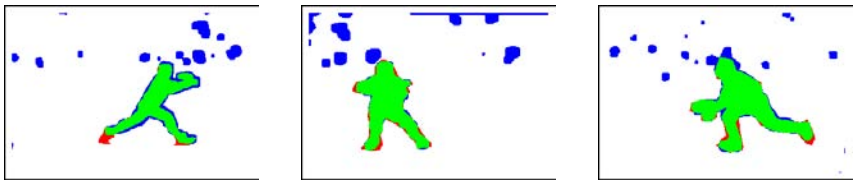
Figure 6.10: Pixel classification for segmentation quality assessment. [57]



(a) frame 1 (b) frame 31 (c) frame 61

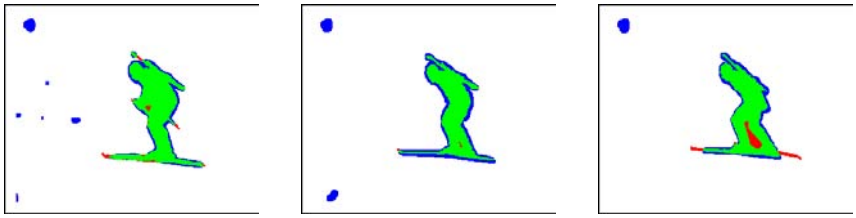


(d) frame 91 (e) frame 121 (f) frame 151

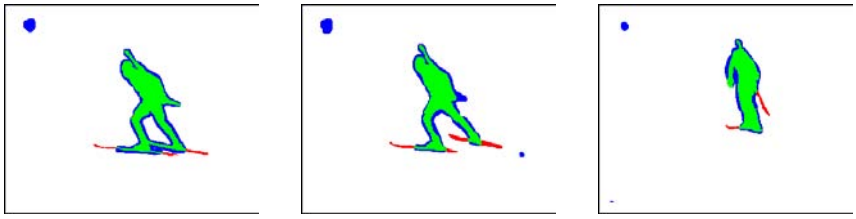


(g) frame 181 (h) frame 211 (i) frame 241

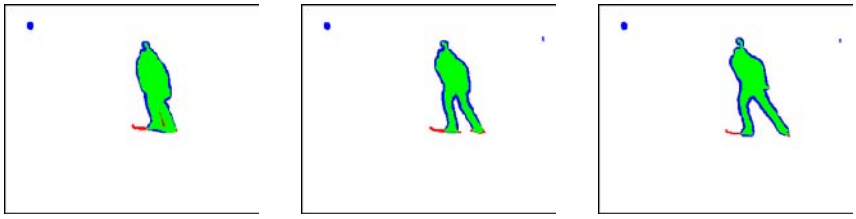
Figure 6.11: Segmentation results using MRF-approach (approach 1) for sequence "Stefan": green - true positives; blue - false positives; red - false negatives.



(a) frame 1 (b) frame 3 (c) frame 5



(d) frame 7 (e) frame 9 (f) frame 101



(g) frame 103 (h) frame 105 (i) frame 107

Figure 6.12: Segmentation results using MRF-approach (approach 1) for sequence "Biathlon": green - true positives; blue - false positives; red - false negatives.

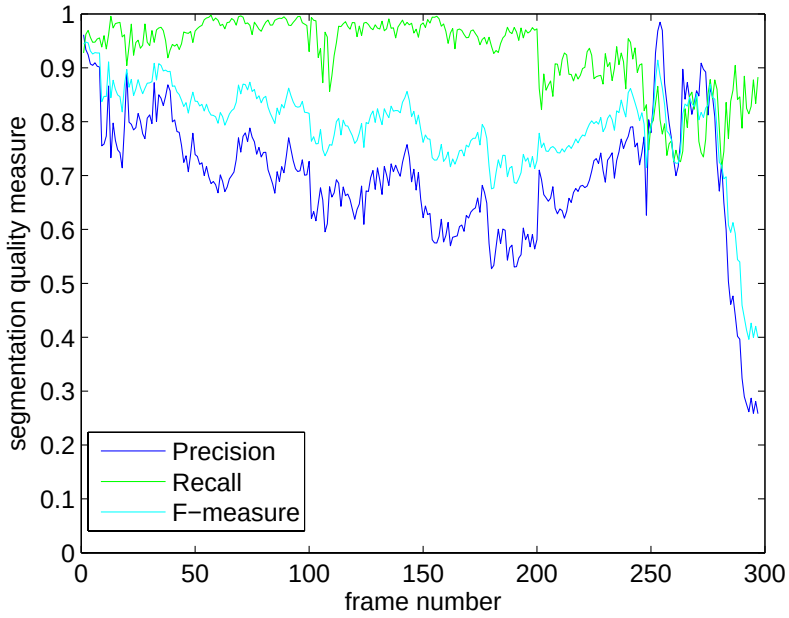


Figure 6.13: Segmentation quality measures for MRF-based segmentation of sequence "Stefan".

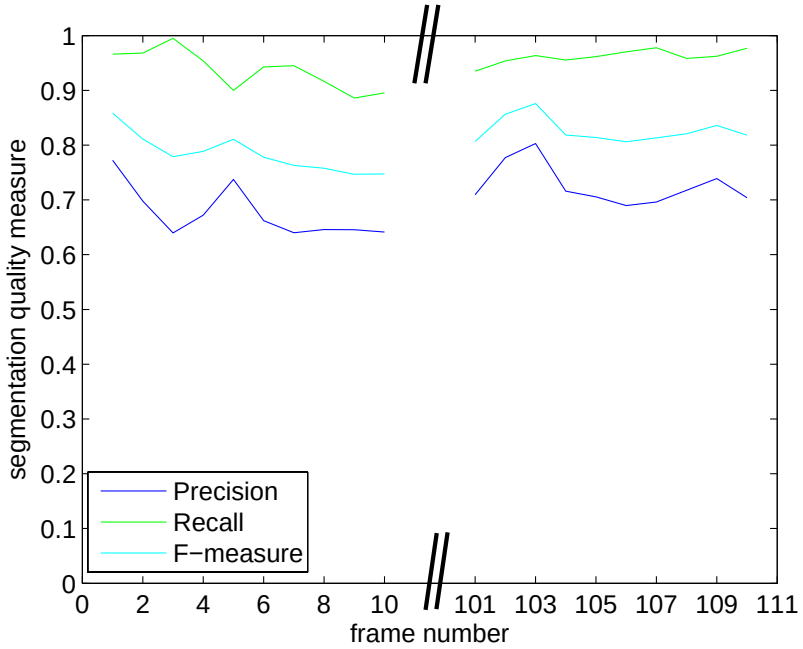


Figure 6.14: Segmentation quality measures for MRF-based segmentation of sequence "Biathlon".

with the actual frame. Background objects that move slightly cause wrong detection and thus too many false positives.

The per-frame results for the segmentation measures P, R, and F are shown in Figures 6.13 and 6.14 for both sequences. The average values are $\bar{P} = 0.70$, $\bar{R} = 0.93$, and $\bar{F} = 0.79$ for "Stefan" and $\bar{P} = 0.70$, $\bar{R} = 0.95$, and $\bar{F} = 0.81$ for "Biathlon". An average F-measure of about 80% is a very good value since it is a midrange accuracy compared to segmentation approaches for still cameras [57]. In contrast to the methods presented in [57] the evaluated scenes have strong camera motion (see comparison in Section 6.5).

6.4 Approach 2: Combined Short-term and Long-term Compensation for Moving Object Segmentation

In the last section we have seen that change detection based on the frame difference between sprite-based background re-projection and any video frame leads to a detection of many false positives. In order to minimize this effect we introduce a combined long-term and short-term compensated change detection technique. It has been proposed in [68]. While the long-term compensated frame difference represents the foreground objects very exactly, the short-term compensated frame differences have no significant energy in the background regions. Additionally, spatial characteristics of the difference images are exploited. This is achieved by a regularization of the resulting difference images (motion compensated) applying anisotropic spatial filters.

6.4.1 Segmentation Approach

The flow chart of the proposed segmentation approach is presented in Figure 6.15. We generate two binary object masks in parallel. Both masks are constructed using change detection techniques together with anisotropic diffusion. For the upper path in Figure 6.15 the compensated frame-to-frame difference is used, which is computed by the subtraction of the luminance values of any frame I_n with the camera motion compensated consecutive frame I_{n+1}. The compensation is achieved applying the short-term image transformation presented in Section 3.2. It is clear that both images contain the foreground elements. Thus the difference image will not precisely represent those objects. However, the error for short-term registration is very small. Thus, artifacts for background regions in the difference frame will be rather unfrequent.

For the lower path in Figure 6.15 the difference between the actual frame I_n and the background re-projection from the sprite S is utilized. The re-projection is achieved by the inversion of the underlying transformation $\mathcal{T}(\mathbf{k}_{r,n}; x, y)$ between reference frame I_r and the actual frame I_n where $\mathbf{k}_{r,n}$ represents the estimated vector of transformation coefficients. The technique for change detection using anisotropic filtering, which is applied on both difference images is specified in the following paragraph.

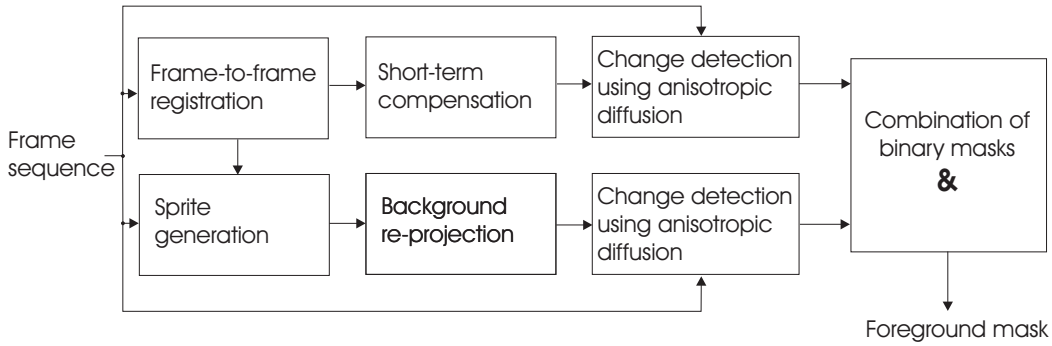


Figure 6.15: Flow chart of the combined short-term and long-term compensation-based segmentation technique applying change detection with anisotropic diffusion.

Objective

Computing the binary change mask from an absolute luminance difference input frame $D(x,y) = |I(x,y) - \hat{I}(x,y)|$ by exploiting spatial conditions applying anisotropic diffusion.

Algorithm

(i) Anisotropic lowpass filtering using diffusion.
(ii) Estimation of threshold τ_c.
(iii) Image binarization using τ_c.
(iv) Removement of small objects and hole filling using morphological operations.

Algorithm 6.1: Change detection applied for combined short-term and long-term compensated segmentation.

Change detection

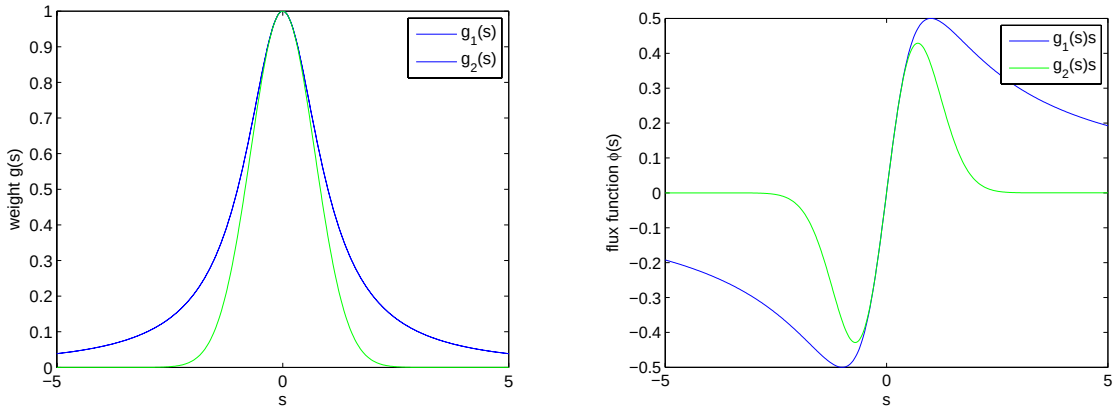
An overview of the change detection method is presented in Algorithm 6.1. First, the absolute image difference $D(x,y)$ is filtered with an anisotropic diffusion filter. This filter iteratively solves the diffusion equation

$$D_t = \frac{dD}{dt} = \mathrm{div}(g(|\nabla D|)\nabla D), \tag{6.10}$$

proposed in [102]. The term $\frac{dD}{dt}$ represents the temporal derivation while ∇D represents the spatial gradient of the absolute difference frame. Note that the time t describes the iterative character of the diffusion process and has nothing to do with real time instances n of the video frame. Equation 6.10 can be regarded as the divergence of the weighted gradient vector field. In Cartesian coordinates it can be noted as

$$D_t = \frac{\partial}{\partial x}\left[g\left(\sqrt{\left(\frac{\partial D}{\partial x}\right)^2 + \left(\frac{\partial D}{\partial y}\right)^2}\right)\frac{\partial D}{\partial x}\right] + \frac{\partial}{\partial y}\left[g\left(\sqrt{\left(\frac{\partial D}{\partial x}\right)^2 + \left(\frac{\partial D}{\partial y}\right)^2}\right)\frac{\partial D}{\partial y}\right].$$

The weighting function $g(|\nabla D|)$ is dependent on the local position and thus, the diffusion process is anisotropic. Function $g(s)$ is also called *edge stopping function* since it is designed



(a) edge stopping function $g(s)$ (b) flux $\phi(s)$

Figure 6.16: Examples of $g(s)$ and $\phi(s)$ with coefficient $K = 1$.

to suppress the influence at strong edges, i.e., regions with a high value for $|\nabla D|$. Typical functionalities for $g(s)$ are

$$g_1(s) = \frac{1}{1 + \left(\frac{s}{K}\right)^2} \tag{6.11}$$

$$g_2(s) = e^{-\left(\frac{s}{K}\right)^2}. \tag{6.12}$$

The term $\phi(\nabla D) = g(|\nabla D|)\nabla D$ can be rewritten as $\phi(s) = g(|s|)s$ and denotes the *flux*. For the one-dimensional case $g(s)$ and $\phi(s)$ are shown in Figure 6.16. The flux is a measure for the strength of the diffusion activity. For both examples of $g(s)$ the flux converges toward zero for high values of s. Tuning the coefficient K we can control the convergence behavior. Thus, if the gradient exceeds a certain value no diffusion occurs. On the other hand for absolute gradient values close to zero the diffusion process approximates a Gaussian lowpass. The overall effect of the iterative diffusion process is a lowpass filter, which operates only inside homogeneous regions bounded by strong edges. With this technique we can effectively smooth the difference images without blurring the object edges. The result is an smoothed absolute difference image D_s

After the diffusion process the threshold for the binarization is estimated. In order to find an appropriate value we assume foreground objects to be present in every frame. Thus, every value $D_s(x, y)$ that significantly exceeds the average difference is marked as foreground object.

$$\tau_c = \underset{\forall x,y}{\text{mean}}(D_s(x, y)) + 0.1 \left(\max_{\forall x,y}(D_s(x, y)) - \underset{\forall x,y}{\text{mean}}(D_s(x, y)) \right) \tag{6.13}$$

Finally, the binarized mask is filtered using morphological operators to remove small artifacts and to close small holes.

To generate the object mask short-term mask and long-term mask are combined applying a logical AND operation. Figure 6.17 exemplifies the whole change detection chain plus the logical mask combination for frame 39 of sequence "Race01".

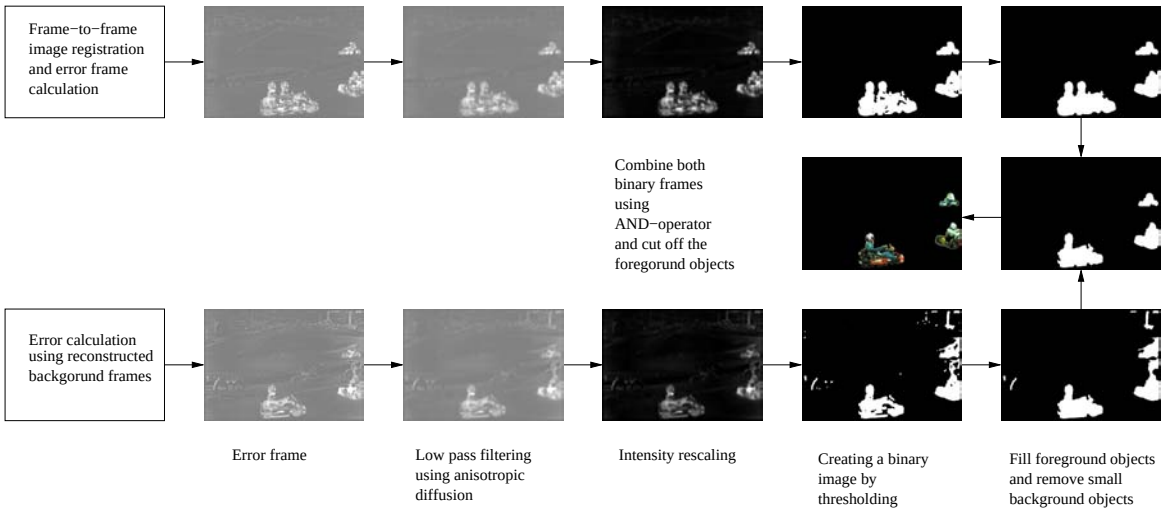


Figure 6.17: Exemplary operation results for change detection and mask fusion for frame 39 of sequence "Race01".

6.4.2 Segmentation Results

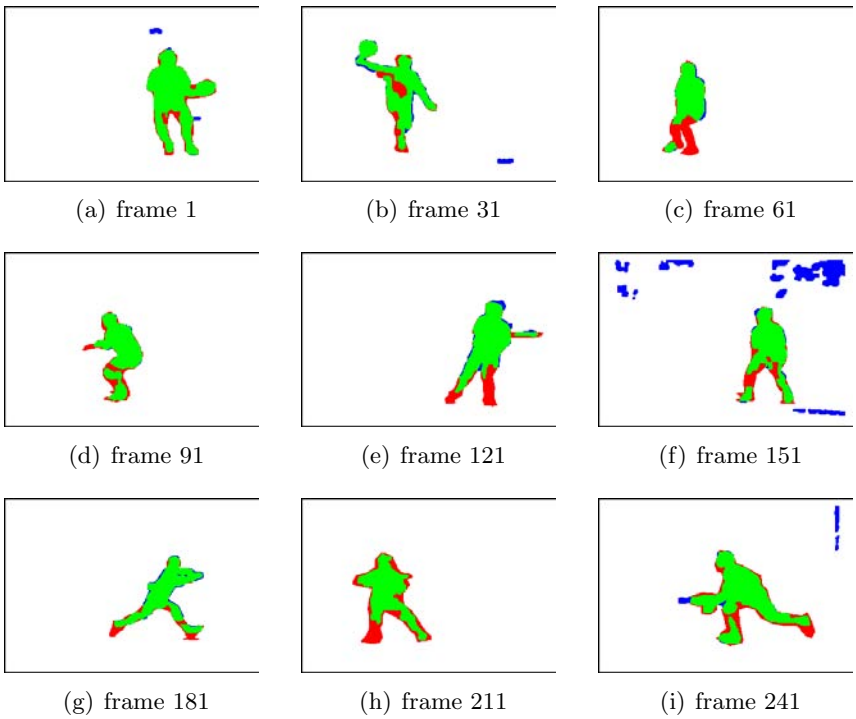
For segmentation quality evaluation we applied the presented approach as well to sequences "Stefan" and "Biathlon". The evaluated change masks with marked true positives, false positives, and false negatives are shown in Figures 6.18 and 6.19. The measured segmentation quality metrics over frames are given in Figures 6.20 and 6.21. The average values for precision, recall, and F-measure are $\bar{P} = 0.88$, $\bar{R} = 0.77$, and $\bar{F} = 0.82$ for "Stefan" and $\bar{P} = 0.78$, $\bar{R} = 0.91$, and $\bar{F} = 0.84$ for "Biathlon". The F-measure for this approach is about 3% to 4% higher than in approach 1.

6.5 Comparison with other Approaches

In order to compare the results obtained by both proposed segmentation methods we assess the segmentation mask for different types of videos. First, we perform a comparison with other motion-based segmentation techniques for scenes with moving camera. To strengthen our results, we additionally compare the resulting change mask for scenes recorded with still camera. Thus, the degradation caused by registration errors is prevented. Moreover, we can show that sprite background estimation and our change detection approaches can compete with latest techniques proposed in the literature.

6.5.1 Moving Camera

We compare our results with the segmentation masks obtained by optical flow-based object segmentation proposed in [61] and [69]. The technique estimates the dense optical flow between neighboring images of a sequence. The object segmentation is then achieved by a motion-based clustering method, i.e., adjacent pixels are classified into the same object

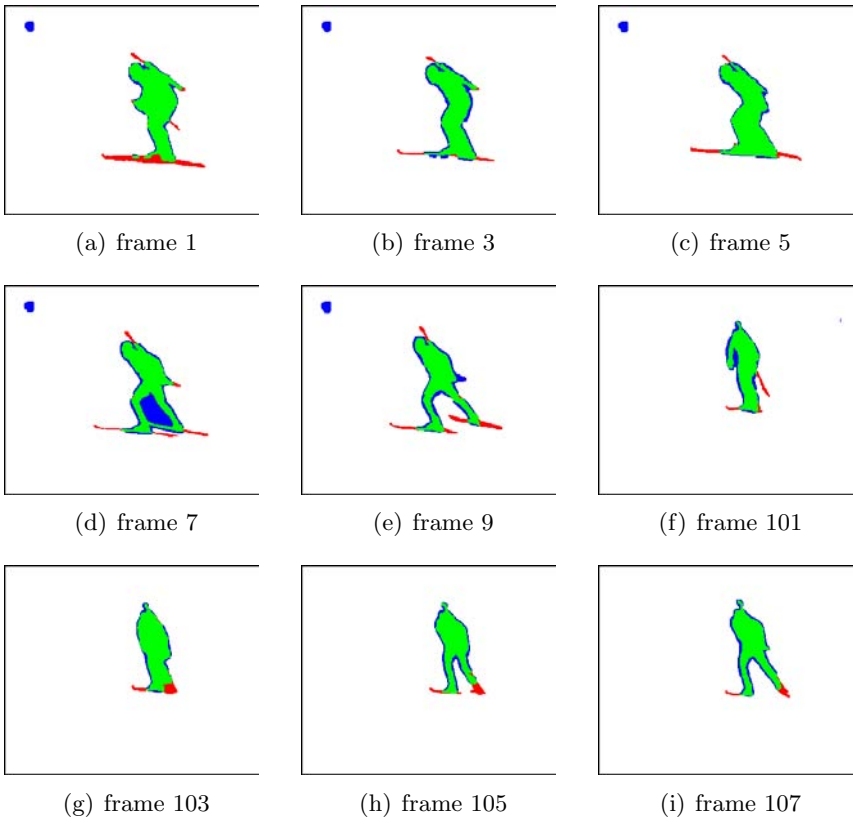


(a) frame 1 (b) frame 31 (c) frame 61

(d) frame 91 (e) frame 121 (f) frame 151

(g) frame 181 (h) frame 211 (i) frame 241

Figure 6.18: Segmentation results using approach 2 for sequence "Stefan": green - true positives; blue - false positives; red - false negatives.



(a) frame 1 (b) frame 3 (c) frame 5

(d) frame 7 (e) frame 9 (f) frame 101

(g) frame 103 (h) frame 105 (i) frame 107

Figure 6.19: Segmentation results using combined long-term and short-term compensation for sequence "Biathlon": green - true positives; blue - false positives; red - false negatives.

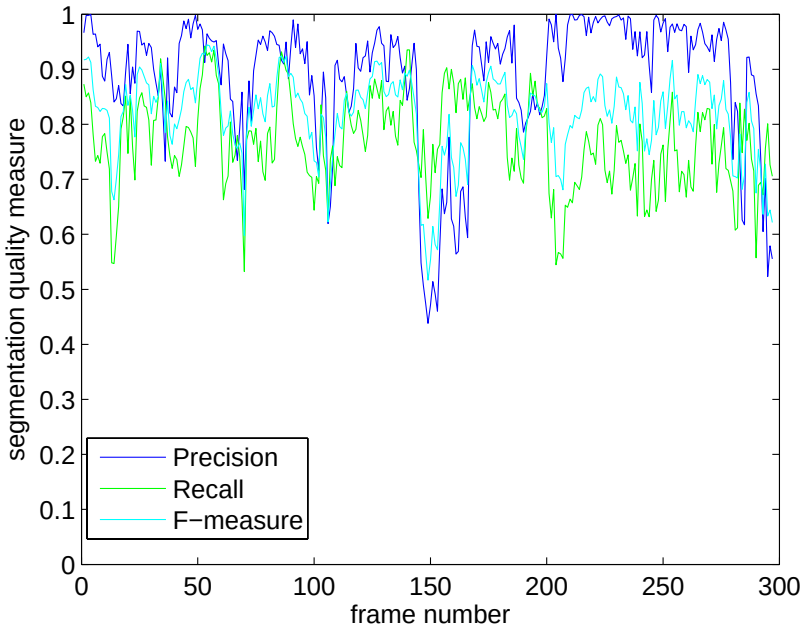


Figure 6.20: Segmentation quality measures for combined long-term and short-term compensation-based segmentation of sequence "Stefan".

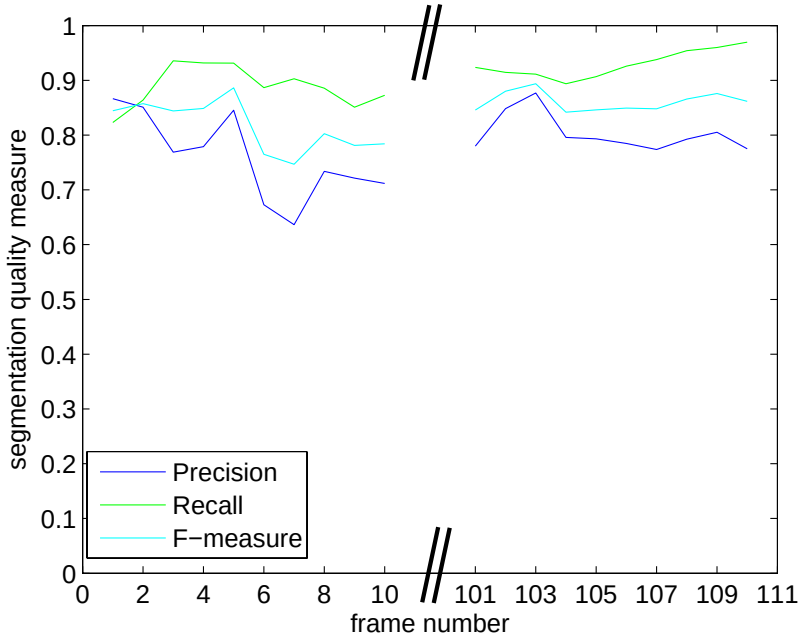
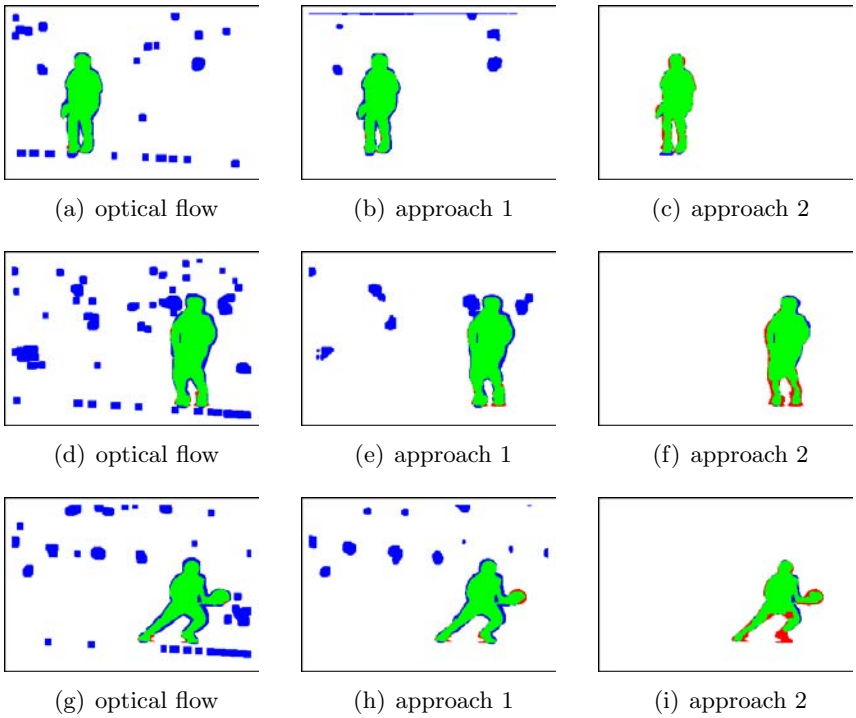


Figure 6.21: Segmentation quality measures for combined long-term and short-term compensation-based segmentation of sequence "Biathlon".



(a) optical flow (b) approach 1 (c) approach 2

(d) optical flow (e) approach 1 (f) approach 2

(g) optical flow (h) approach 1 (i) approach 2

Figure 6.22: Pixel classification for frame 54 (upper), frame 129 (middle), and frame 170 (lower) of sequence "Stefan": green - true positives; blue - false positives; red - false negatives.

if the distance of their motion vectors is below a given threshold. Table 6.1 gives the segmentation accuracy measures of this method compared to our approaches for sequence "Stefan". From the values for the F-measure, which is most meaningful, it can be seen that both approaches outperform the flow-based segmentation by far. Hereby, approach 2 is slightly more exact than approach 1. This can be explained by a lower false positive rate due to combination of short-term and long-term change mask and thus a much higher precision P. Pixel classification results for example frames of "Stefan" are shown in Figure 6.22. Due to the coloration of false positives and false negatives, the improvement can also be verified subjectively. Finally, Figure 6.23 depicts the F-measure over the sequences frame number.

	Mean			Standard derivation		
Segmentation	F-measure	Precision	Recall	F-measure	Precision	Recall
Optical flow [69]	**0.69**	0.54	0.96	0.074	0.094	0.029
Approach 1	**0.79**	0.70	0.93	0.056	0.089	0.051
Approach 2	**0.82**	0.88	0.77	0.074	0.116	0.082

Table 6.1: Segmentation accuracy measures for sequence "Stefan" (SIF, 300 frames, 30 fps).

6.5.2 Still Camera

In order to compare the segmentation results with other change detection algorithms from the literature we applied the presented approaches, the MRF-based change detection (ap-

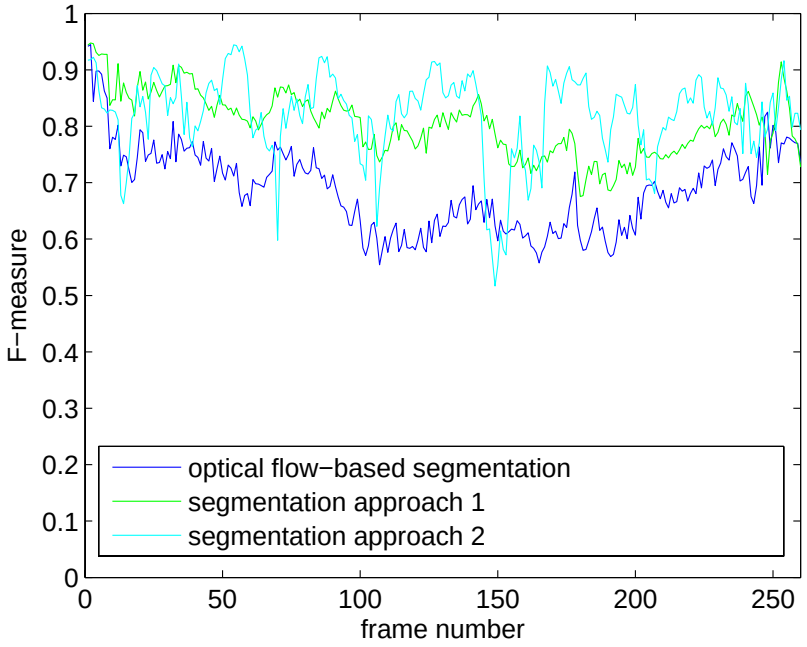


Figure 6.23: Comparison of F-measure for object segmentation of sequence "Stefan".

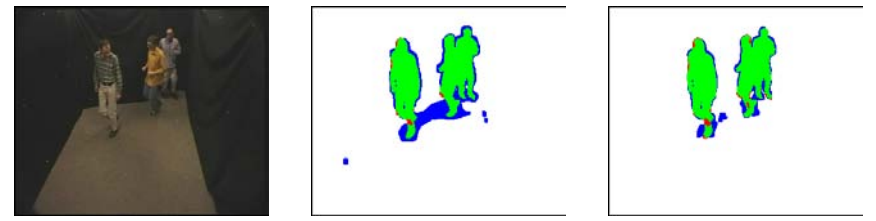
proach 1) and the combined short-term and long-term compensation-based object segmentation (approach 2), on scenes captured with motionless camera. We compared our object segmentation techniques to three different approaches, presented by Shen [121], Hong et al. [44], and Cavallaro et al. [11]. An in-depth comparison of these methods can be found in [57]. It is clear that the transformation $\mathcal{T}$ for still camera is always the identity. Thus, the image registration is not necessary. However, the sprite generation technique presented in Chapter 3 will generate an accurate background estimation. The segmentation measures for sequence "Group" (CIF, 60 frames, 25 fps) are compared against each other in Table 6.2. The results for the F-measure show that our approaches, and here especially approach 2, can

	Mean			Standard derivation		
Segmentation	F-measure	Precision	Recall	F-measure	Precision	Recall
Cavallaro [11]	**0.82**	0.70	0.99	0.038	0.053	0.008
Hong [44]	**0.81**	0.77	0.85	0.023	0.026	0.036
Shen [121]	**0.91**	0.90	0.92	0.012	0.021	0.019
Approach 1	**0.78**	0.61	0.96	0.018	0.028	0.015
Approach 2	**0.85**	0.81	0.90	0.019	0.026	0.035

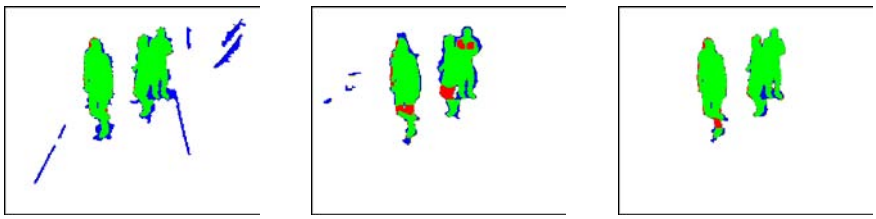
Table 6.2: Segmentation accuracy measures for sequence "Group".

compete with latest change detection techniques. Only the method proposed by Shen [121] is a higher value. Images for subjective segmentation comparison are given in Figures 6.24 and 6.25. The images show that the amount of wrongly classified pixels (blue and red) is as small as (or even smaller than) for the compared approaches.

The average values for precision, recall, and f-measure of the segmentation results for sequence "Hall and Monitor" (CIF, 200 frames, 25fps) are shown in Table 6.3. For this

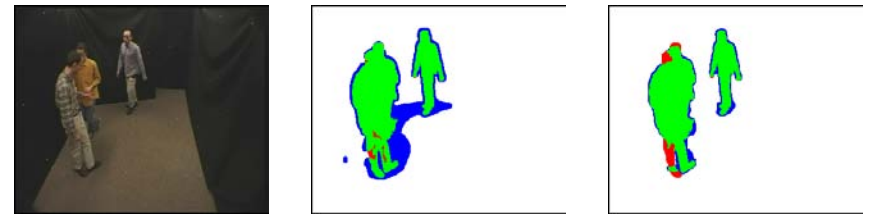


(a) original (b) approach 1 (c) approach 2

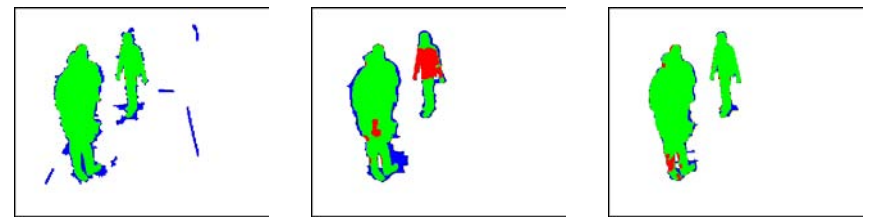


(d) Cavallaro [11] (e) Hong [44] (f) Shen [121]

Figure 6.24: Pixel classification results for frame 16 of sequence "Group": green - true positives; blue - false positives; red - false negatives.



(a) original (b) approach 1 (c) approach 2



(d) Cavallaro [11] (e) Hong [44] (f) Shen [121]

Figure 6.25: Pixel classification results for frame 40 of sequence "Group": green - true positives; blue - false positives; red - false negatives.

sequence the results achieved by the combination of shot-term-based and long-term-based change detection (approach 2) are in the range of the best object segmentation technique. Results for pixel classification of two frames are shown in Figures 6.26 and 6.27.

	Mean			Standard derivation		
Segmentation	F-measure	Precision	Recall	F-measure	Precision	Recall
Cavallaro [11]	**0.85**	0.78	0.93	0.034	0.056	0.039
Hong [44]	**0.80**	0.89	0.73	0.038	0.045	0.065
Shen [121]	**0.88**	0.93	0.83	0.047	0.029	0.079
Approach 1	**0.82**	0.72	0.95	0.028	0.041	0.029
Approach 2	**0.86**	0.80	0.91	0.035	0.054	0.031

Table 6.3: Segmentation accuracy measures for sequence "Hall".

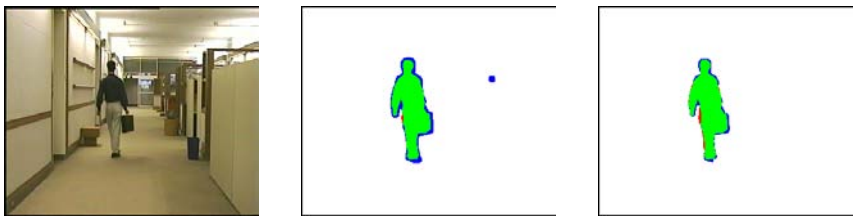
6.6 Chapter Summary

In this chapter a brief introduction into the vast literature on moving object segmentation was given. Following the classification scheme in [166] we summarized the different types of segmentation. Hereby, it is important to distinguish between pure motion-based segmentation techniques and spatio-temporal techniques. While the former type relies only on temporal changes, the latter also exploits spatial characteristics of the single images and thus is usually more exact.

In the main part of this chapter we presented two segmentation approaches, both relying on change detection applied on global motion compensated image subtraction. The first approach utilizes the difference between the re-projected sprite image free of any foreground objects and the associated original frame. It exploits only motion-based changes. In order to stabilize the results a Markov random field (MRF) approach is chosen where pixel states of the previous change mask influence the state of a pixel in the mask of the actual frame. Thus, this segmentation technique can also be regarded as a combined object segmentation and tracking.

The second approach is a combination of change detection using short-term global motion compensation between adjacent frames and change detection based on long-term motion compensation from the sprite image. While the short-term registration is more exact, the long-term re-projection has the advantage of removed foreground objects. Combining both advantages results in a very exact moving foreground segmentation method. Applying anisotropic diffusion filter to smooth the difference frames eases the automatic selection of a threshold value for change detection. Only in the absence of any foreground object the segmentation process fails.

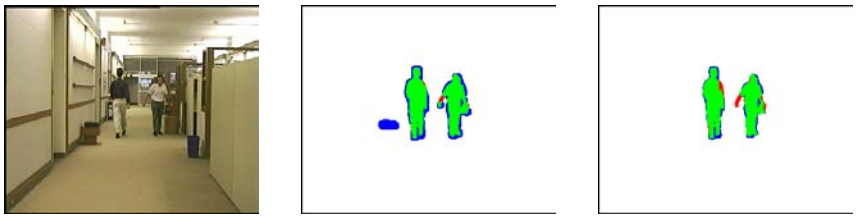
Finally we compared the object segmentation results achieved by both presented approaches with other methods. In case of moving camera both approaches significantly



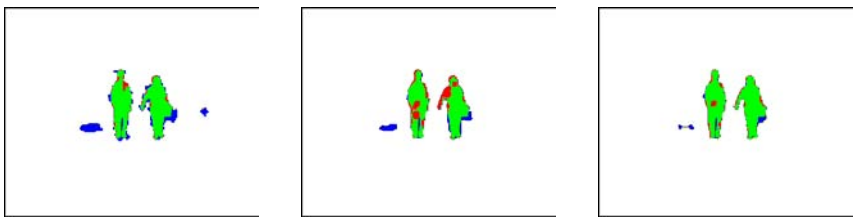
(a) original (b) approach 1 (c) approach 2

(d) Cavallaro [11] (e) Hong [44] (f) Shen [121]

Figure 6.26: Pixel classification results for frame 56 of sequence "Hall": green - true positives; blue - false positives; red - false negatives.



(a) original (b) approach 1 (c) approach 2



(d) Cavallaro [11] (e) Hong [44] (f) Shen [121]

Figure 6.27: Pixel classification results for frame 180 of sequence "Hall": green - true positives; blue - false positives; red - false negatives..

outperform the compared optical flow-based segmentation. In order to strengthen the segmentation results the algorithms were also applied on scenes captured by still camera. In comparison to latest change detection techniques our approaches, especially approach 2, yield similar results in terms of pixel-based segmentation accuracy measures.

The segmentation masks can be utilized in several different applications. In many cases, e.g., object-based video coding, it is of higher importance to achieve high recall values R, i.e., a low rate of false negatives, than to obtain a good precision value. This is because a full detection of the moving foreground objects is often required. Thus, in many cases the obtained segmentation masks are strongly dilated to achieve good subjective quality. In the next chapter, we will show that this is the case for sprite-based video coding. Hence, in terms of a high recall together with an acceptable overall segmentation accuracy both of our segmentation approaches can be utilized.

Chapter 7

Background Sprites and Video Coding

Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius – and a lot of courage – to move in the opposite direction.

Albert Einstein (1879-1955)

The compression of digital video is one of the big challenges for modern media technologies. The aim is to reduce redundancies within one or several videos as much as possible and to represent the data in an optimal way, such that the information contained is stored with minimal usage of storage capacity or bandwidth. The development of video coding techniques has seen a very large improvement over the last 20 years. Applied science has pushed the efficiency of such video codecs (coders/decoders) towards satisfying results. The most successful concept, so far, is a combination of predictive coding – spatial and/or temporal – and a transform coding. This so-called *hybrid video codecs* have been found to be superior for both, reliability and efficiency [110].

With the standardization of the actual best performing and most flexible Advanced Video Codec H.264/AVC in 2003, the newest video coding standard of the ITU-T Video Coding Expert Group (VCEG) and the ISO/IEC Moving Picture Expert Group (MPEG) that form the Joint Video Team (JVT) has been released. It marks another high point in the logical development of hybrid video codecs beginning with H.261 in 1990, followed by MPEG-1 Part 2 (1991), H.262/MPEG-2 Part 2 (1994), H.263 (1995/96), and MPEG-4 Part 2 (1999). Actually, H.264/AVC is part 10 of the group of multimedia standards classified under "MPEG-4". It is designed to meet network-friendly "conversational" and "non-conversational" requirements [160].

The growing storage capacity and higher bandwidths for digital multimedia transmission may raise the question for the need of further development of video codecs. Latest research into media production and storage capabilities have shown that the digital information, recorded by consumers and professionals, is growing much faster than the storage capacity

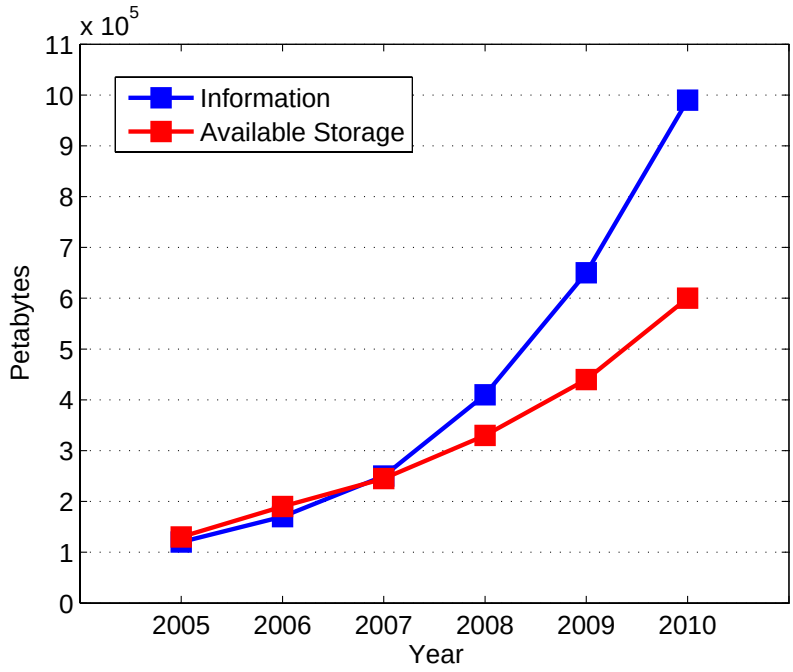


Figure 7.1: Prediction of the world wide storing capacity and the emergence of digital information. Note, that from the year 2008 on the "produced information" exceeds the available storage by far. Source: [35]

(see Figure 7.1 as published by the International Data Corporation IDC [35]). Thus, in consequence of this phenomenon, digital content has to be thrown away over the next years. Higher efficiency in media compression can alleviate the effect since it is economic and universally applicable.

Retuning and improvement of the basic tools that follow the paradigms of block-based hybrid video codecs has always led to best coding performances and therefore, has put out all rivaling coding approaches, e.g., wavelet video coding, of the competition. This has, among other things, become possible through growing hardware performances compensating for growing algorithmic complexity [97]. The advantage of such continuous development is the reliability and universality of the resulting codecs. But then, major improvements in complexity reduction and coding performance are not likely to happen following the same paradigms. In Figures 7.2 and 7.3 the flowchart of commonly used hybrid video codecs, e.g., H.264/AVC, are depicted. Since usually several prediction methods in the coder compete for the best *rate-distortion* (RD) performance, a prediction control unit has to be used to select the optimal prediction mode. In the history coding efficiency has improved with the number of possible prediction modes.

Several proposals were made in order to change the direction of coding and its coding paradigms [96], [20], [98]. In MPEG-4 Part 2 (MPEG-4 Visual) a new application driven coding concept was introduced. Core and Main profile of that standard where designed to support *object-based video coding* (OBVC) [125], [126]. Following the concept, a video sequence consists of different arbitrarily shaped video objects that represent a particular

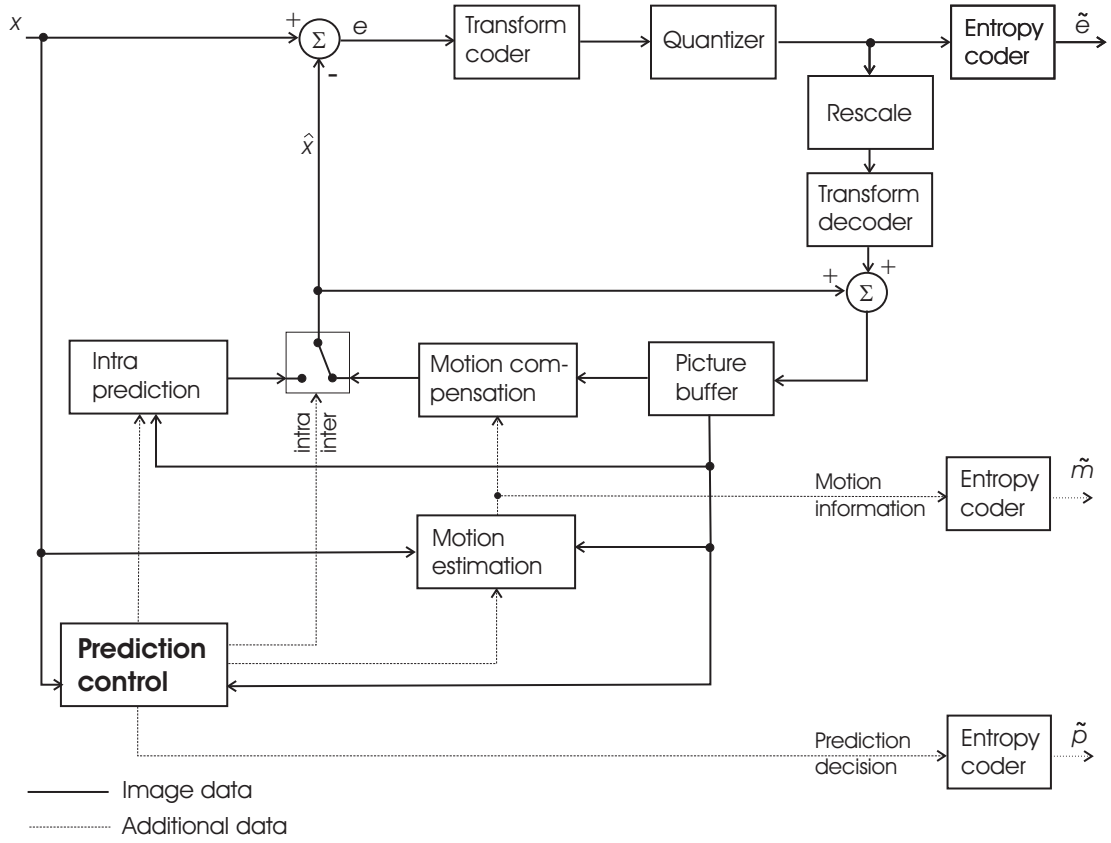


Figure 7.2: Flowchart of hybrid video coder for block-based coding. The intra prediction unit utilizes only parts of the actual picture that have already been processed.

scene. Thus, it is a move away from regarding videos as a collection of rectangular frames. It allows higher flexibility for the user and promises better coding performances for certain scenarios. Unfortunately, the user has to generate all the *video objects* (VO) on his own. In order to do this automatically, methods from computer vision and machine learning science can be applied. Since these methods are computational complex itself and sometimes lack of reliability, the object-based part of MPEG-4 video standard is not used in praxis.

However, several coding tools of MPEG-4 Visual deserve a reinvestigation since they promise performance improvement of – and convergence with the conventional rectangular hybrid coding strategy [127]. The important fact for successful usage is that not only the coding strategy but also the object generation technique has to be provided. A very promising coding approach already provided in MPEG-4 is the sprite-based video coding or sprite coding (SC). The main idea is to separately code the background sprite and the foreground video object. The decoder then recomposes the video by combining the re-projected background with the foreground object. Over the last 10 years authors have proposed various approaches for efficient sprite coding in [16], [38], [51], [55], [73], [78], [133], and [163]. There exist two techniques for utilizing sprites for video coding: *static sprite coding* and *dynamic sprite coding*. While the former method handles a scene as rigid, the latter provides a continuous update of the sprite. Since the update is difficult to handle and

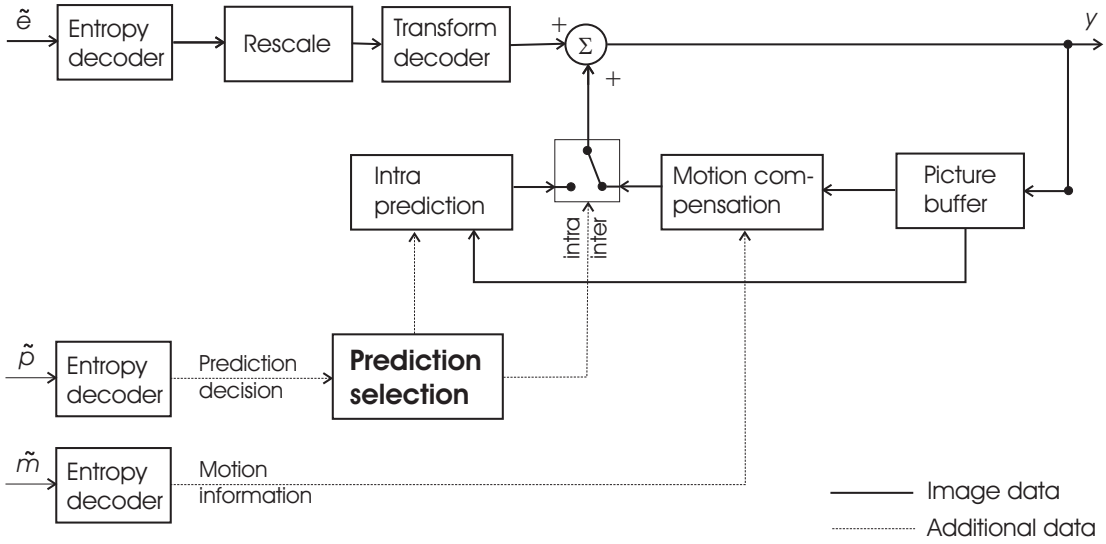


Figure 7.3: Flowchart of hybrid video decoder for block-based decoding.

lessens the compression efficiency, it has not been standardized at all [101]. Only a very simplified form, i.e., *global motion compensation* (GMC) has been adopted.

In this chapter we will introduce two static sprite-based coding strategies for efficient video coding. Both are not built up on the MPEG-4 Visual coding schema but adopt the latest version of H.264/AVC as basis codec. However, both strategies are somewhat contrary in their handling of independently moving foreground objects and have therefore very different coding performance. First, we present a method that uses sprite prediction as additional intra prediction mode for H.264/AVC on block level. It has the advantage of an implicit segmentation being performed by the prediction control unit of the coder. Second, we introduce a sprite coding technique that explicitly performs the segmentation of foreground objects using the methods presented in Chapter 6. The foreground objects are coded as rectangular frames. On the decoder side sprite-based video background and foreground objects are combined. The main difference between both methods is the background residual handling. While for the first approach the residual is automatically coded in the coding loop, the second method only provides background information from the compressed sprites. The performance of both techniques is extensively assessed in the experiments section. Additionally, we investigate the effect of *super-resolution sprite coding* and *multiple sprite coding*, which promise further improvement of the coding efficiency.

7.1 Static Sprite Coding

A chart of the basic functionalities of sprite-based video coding, as it is usually applied, is shown in Figure 7.4. Prior to the coding of the information the sprite and the *video object planes* (VOP), i.e., the segmentation masks, have to be available. If an automatic generation process is included, sprite-based coding cannot be utilized for real-time applications, since it

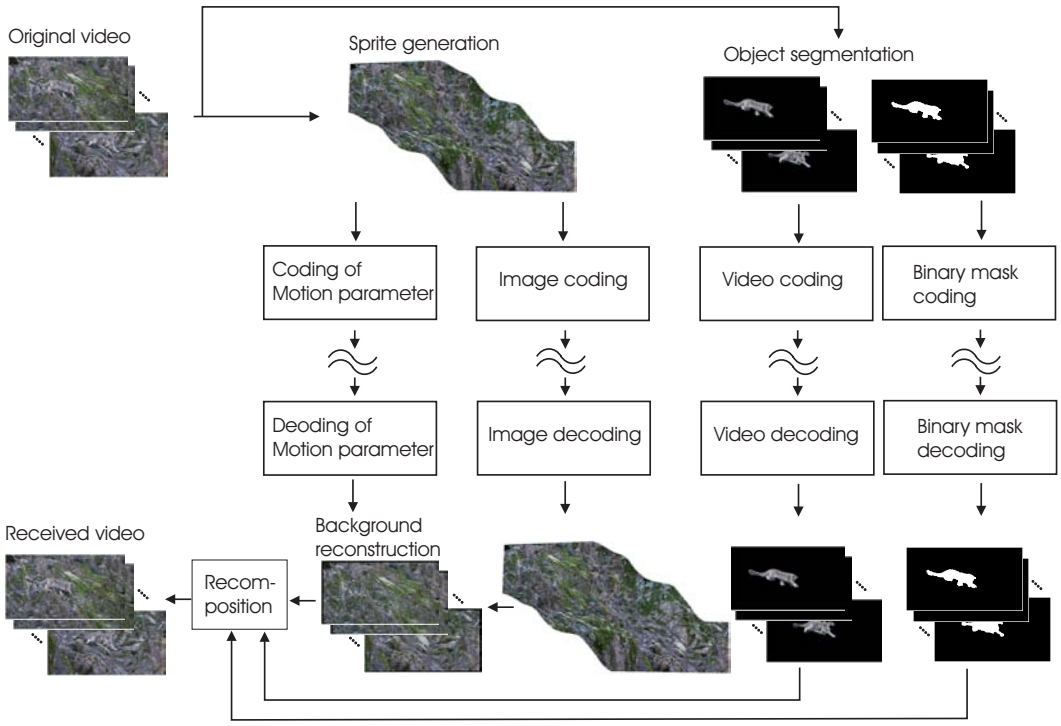
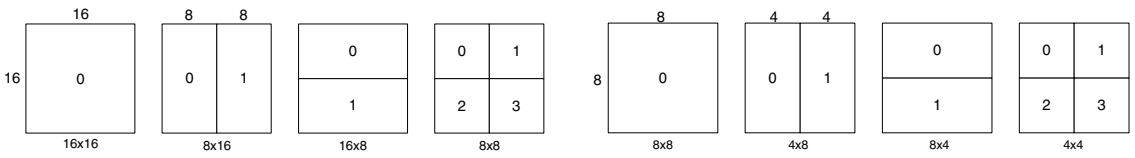


Figure 7.4: Scheme of static sprite coding technique.

requires long-term temporal processing. Background motion parameters, sprite, foreground objects and foreground mask are then coded independently. At the decoder's side all objects are combined to generate the final video sequence. In its basic form the sprite coder is limited in terms of objective video quality. Since no background residual, i.e., the difference between sprite predicted background and the original background, is transmitted, the quality can maximally achieve the results for the uncoded sprite. Due to registration errors, this limitation can be very strict but in subjective evaluations this limit is not existent since the observer cannot see the slight displacements of the background. In [163] a H.264-based background residual coding was presented. The author achieved a coding gain compared to the basis codec even for quality levels above the sprite registration limit.

7.2 Sprite-based Video Coding with Embedded Segmentation

In this section we will describe two new approaches for static sprite coding of digital video sequences applying H.264/AVC as basis codec. The first is a direct implementation of a new 16x16 block prediction mode into the H.264 encoder and decoder and therefore is fully compatible to decode the bitstream of the H.264/AVC standard. The second method is more related to the classical approach presented in Section 7.1. It uses the H.264 standard to encode the sprite image and the previously and automatically generated object sequence. For this approach, H.264 can be replaced by any other hybrid video codec designed for compressing rectangular video frames.



(a) Macroblock partition (b) Sub-macroblock partition

Figure 7.5: Partition of macroblocks and sub-macroblocks.

7.2.1 A new Prediction Mode for H.264/AVC Applying Sprite Coding with Implicit Segmentation

In order to explain the integration of a new prediction mode on 16x16 macroblock basis into the H.264/AVC codec we will briefly discuss the functionalities of the H.264 *reference codec*, especially the prediction control of the reference encoder, which is actually not a part of the standard. Since the standard only describes the bitstream itself, which mainly defines the decoder, it is up to the codec designer to implement an intelligent prediction control that selects the prediction mode in an optimal way. In some codecs this is achieved by *rate-distortion optimization*. It aims to find an operating point in the rate-distortion diagram that is as close as possible to the optimum point [109].

7.2.1.1 Prediction Modes in H.264/AVC Main Profile

The video picture is coded in H.264/AVC as one or more *slices* that in sum contain all macroblocks, i.e., 16 pixels × 16 pixels image blocks, of the picture. Slices can be chosen in a way that spatial inter-dependencies are utilized to limit transmission errors. Dependent on the *slice type* the number of possible prediction modes for an actual macroblock can vary. Possible slice types are: I (Intra), P (Predicted), and B (Bi-predictive). Possible macroblock prediction modes are SKIP, INTRA, INTER, and their respective submodes. Macroblocks in an I-slice can only be predicted in intra prediction mode, which means only spatial prediction is allowed. This can be performed on macroblock level or 4x4 block size. Macroblocks in a P-slice can be predicted in intra mode as well as in inter mode, whereas only one list of reference pictures is used. Inter prediction means that the block is temporally predicted applying motion estimation. Possible block sizes are 16x16, 16x8, 8x16, and 8x8. In 8x8 (sub-macroblock) prediction mode a further partition into 8x4, 4x8, and 4x4 can be made. Macroblocks in a B-slice can be predicted in intra mode as well as in inter mode, whereas one or two lists of reference pictures are used. The block sizes for temporal prediction are the same as for P-predicted blocks. The possible block partitioning modes are also shown in Figure 7.5. All macroblocks in inter mode slices (P and B) can also be coded in skip mode, which means that the content of the temporal previous block is used for this block.

For every possible prediction p per macroblock the residual $e(p)$ is computed and afterwards transform coded, quantized, and entropy coded in order to determine the distortion D and the rate R, i.e., the number of written bits.

7.2.1.2 Transform Coding and Quantization in H.264/AVC

The residual e is transformed on luma and chroma blocks using a DCT-based integer transform for the residual of a block. Depending on whether the residual was computed in 16x16 Intra mode or not, a Hadamard transform is chosen to again transform the DC coefficients. The transformation is usually performed on 4x4 blocks except for DC coefficients of the chroma components (here a 2x2 Hadamard transform is applied). The advantage of these transforms is that loss of information occurs applying both, transform and inverse transform [110]. Thus, in contrast to previous standardized codecs no mismatch between transformed and inverse transformed coefficient appears due to full integer arithmetic. The only unit of the coder that introduces a loss is the quantizer.

H.264/AVC uses a scalar quantizer and the basic quantization operation is performed by

$$Z_{ij} = \text{round}\left(\frac{Y_{ij}}{Q_{step}}\right), \tag{7.1}$$

where Z_{ij} is the quantized value of any transform coefficient Y_{ij}. Q_{step} is the quantization step size and is addressed by the quantization parameter $QP = 0, \ldots, 51$. Q_{step} is growing exponentially and doubles for every increment of 6 in QP. Actually, the coefficients are not only scaled according to the QP but also to their position in the transform block. The inverse quantization, which is rather a rescaling, is achieved following Equation 7.2.

$$Y'_{ij} = Z_{ij} Q_{step} \tag{7.2}$$

The quantization for DC coefficients using 16x16 Intra prediction differs slightly from the specification for the other coefficients but in principle has the same impact.

Applying transform coding, quantization, rescaling, and inverse transform coding on any residual $e(p)$ one obtains the decoded residual $e'(p)$, which is used to determine the distortion D as sum of squared errors (SSE)

$$D = \sum_{x=0}^{15} \sum_{y=0}^{15} \left(e(p;x,y) - e'(p;x,y)\right)^2. \tag{7.3}$$

7.2.1.3 Entropy Coding in H.264/AVC

Entropy coding is the lossless coding of a discrete source signal with source alphabet $\mathbf{A} = \{a_1, \ldots, a_A\}$ utilizing a code alphabet $\mathbf{B} = \{b_1, \ldots, b_B\}$. The code alphabet is a code of variable length. According to their probability of occurrence the source symbols or a combination of source symbols is associated with the symbols of the code alphabet, being

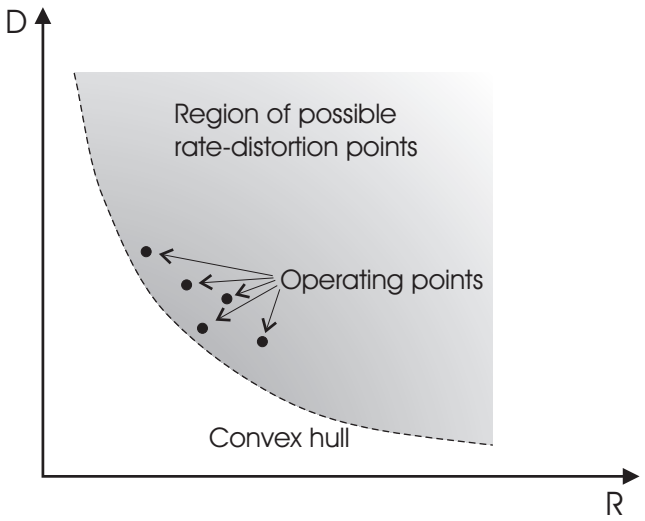


Figure 7.6: Rate-distortion points.

rather short for source symbols with high probability and vice versa. Thus, a rate close to entropy $H(\mathbf{A})$ can be achieved. Usually the probabilities have to be estimated a priori to the encoding. Since they may vary from context to context, in H.264/AVC *context-based adaptive entropy coding* strategies are applied.

Starting with slice layer and below, syntax elements in H.264/AVC are either coded in *context-based adaptive binary arithmetic coding* (CABAC) for all data, or as mixture of *context-based adaptive variable length coding* (CAVLC) for residual data and Exp-Golomb codes for other data. The advantage of context-based adaptive models over context- independent models is twofold. First, different probability models for different syntax elements can be chosen and second, the probabilities of the models can be adapted according to a measure of the local statistics. For CABAC there are almost 400 context models for the various syntax elements.

On macroblock level we can count the *number of written bits* of any prediction mode while measuring the outcome of the entropy encoder. This, of course, is a sum of encoding results for all necessary syntax elements that stand for different coding parameters, e.g., *residual data*, *macroblock type*, *coded block pattern*, QP, and possible *reference frame index* plus *motion vector* data. The number of bits can directly be used as a measure of the rate R since the number of macroblocks and the video's frame rate remains constant.

Having rate R and distortion D for any possible macroblock prediction mode the prediction control can decide which prediction mode is optimal in sense that its operating point is as close as possible to the theoretical limit.

7.2.1.4 Rate-distortion Optimization in the H.264/AVC Reference Encoder

Following information theoretical fundamentals there exists a lower bound in rate R_X for any signal X to be compressed in a lossless way. This lower bound is marked by the entropy

$H(X)$ of this signal.

$$R(X) \geq H(X) \tag{7.4}$$

For lossy compression, rate-distortion theory tells us that this bound is a convex hull in the rate-distortion diagram. In Figure 7.6 such a diagram is depicted exemplary. For every acceptable distortion D exists a minimum rate $R(D)$ that lies exactly on the convex hull. A distortion D can only be achieved if the actual rate is equal or greater than $R(D)$. Usually as distortion the sum of squared errors between the original pixels and the predicted is computed. Other measures like the sum of absolute errors (SAE) can be adopted as well.

A simple but powerful solution for finding the operating point closest to the convex hull is achieved by using *Lagrangian optimization*. We formulate a cost function $J(R, D)$ as follows:

$$CP_{opt} = \min_{\forall CP} J(R, D) = \min_{\forall CP} D + \lambda R. \tag{7.5}$$

In Equation 7.5 CP represents the set of all variable coding parameters such as the prediction mode p or the estimated motion vectors mv. The *Lagrangian multiplier* λ is the slope of the convex hull $R(D)$. Determining the optimal λ is usually a very complex process. In the reference model it is empirically determined as a function of the quantization parameter QP indexing the quantization step size for the transformed coefficients [151].

$$\lambda_{SSE} = 0.85 \cdot 2^{\frac{QP-12}{3}} \tag{7.6}$$

λ_{SSE} represents the multiplier for utilizing the SSE as distortion measure. If the SAE is applied the Lagrangian multiplier yields $\lambda_{SAE} = \sqrt{\lambda_{SSE}}$.

7.2.1.5 Integrating Sprite Prediction into H.264/AVC - SPAVC

The first step for integrating sprite prediction into H.264/AVC is the independent coding of the sprite image S. Typically every image coder can be used. Since the performance of the H.264/AVC Intra prediction is much better than JPEG or JPEG2000 still image codec, it is reasonable to code the sprite image S using H.264/AVC intra prediction. In order to ensure close loop prediction of the video sequence, the encoded sprite image $\tilde{S}$ has to be decoded on the encoder side as well. The decoded image S' is the basis for the macroblock prediction in the codec.

The predicted macroblock from the sprite competes with all other possible prediction modes in the prediction control. Since it is always available, it can be chosen in Intra slices, P slices, and B slices. From the prediction control point of view it is very similar to any 16x16 intra prediction since only the prediction type and the residual have to be transmitted. In the rate-distortion measure the data transmitted outside the coding loop (sprite image and warping parameters) are not considered. The reason for this is that the sprite has to be transmitted anyway. Only a global R-D measure over the whole sequence could solve the question if sprite coding should be applied or not.

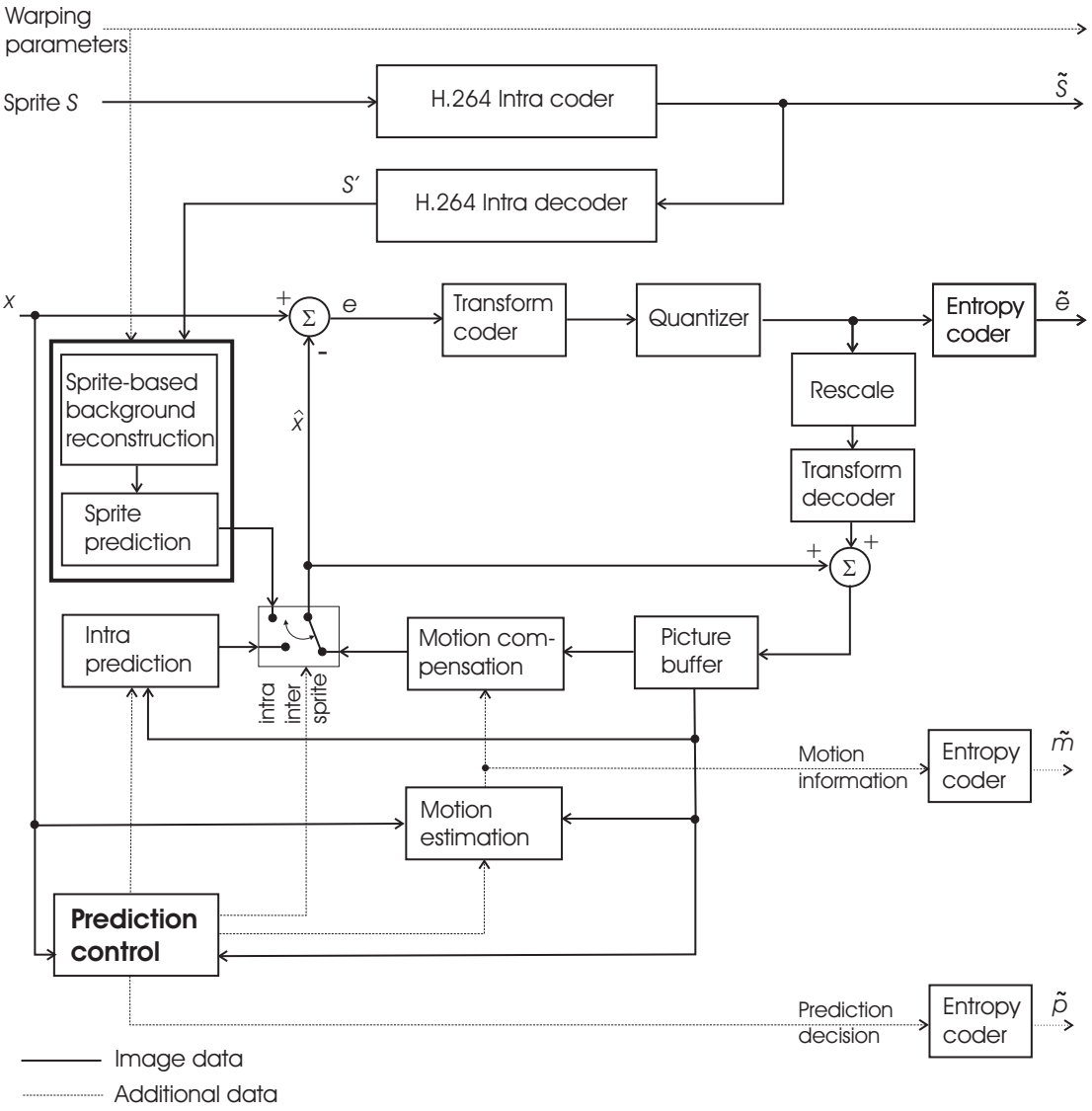


Figure 7.7: Flowchart of SPAVC: H.264/AVC hybrid video coder for block-based coding with integrated sprite prediction.

In Figure 7.7 the flow chart of H.264/AVC with integrated sprite prediction - *sprite prediction AVC* (SPAVC) - is shown. In contrast to the normal H.264/AVC flow chart (Figure 7.2) a new prediction block is introduced. Other parts of the coder remain similar and compatible to the standard.

7.2.1.6 Implicit Block-based Segmentation

In this section we will show that the proposed SPAVC performs an implicit segmentation on macroblock level. Due to the rate-distortion optimization described in section 7.2.1.4, macroblocks that contain foreground objects are unlikely to be predicted in sprite prediction mode since the residual e will be of high energy, which causes high rate compared to intra prediction or inter prediction modes. An example for an I-frame and an P-frame is shown in Figure 7.8. The colored blocks represent all blocks coded in sprite prediction mode. The

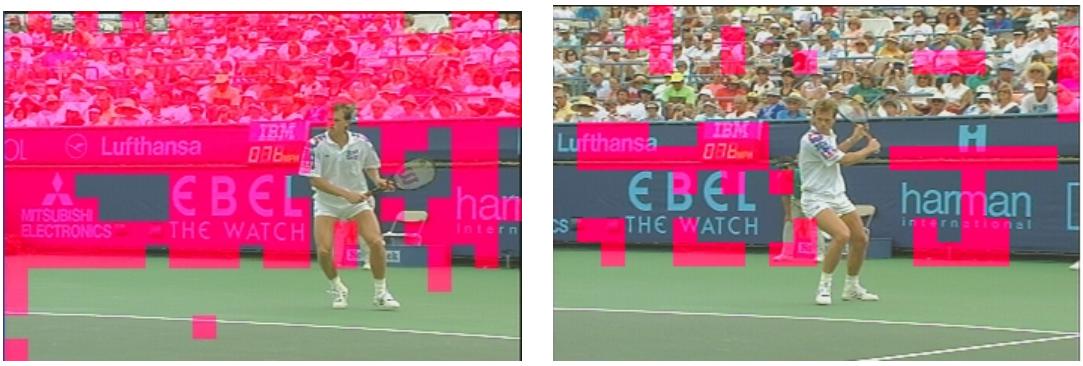


Figure 7.8: Blocks to be encoded in sprite prediction mode for sequence "Stefan", frame 1 (left) and frame 10 (right).

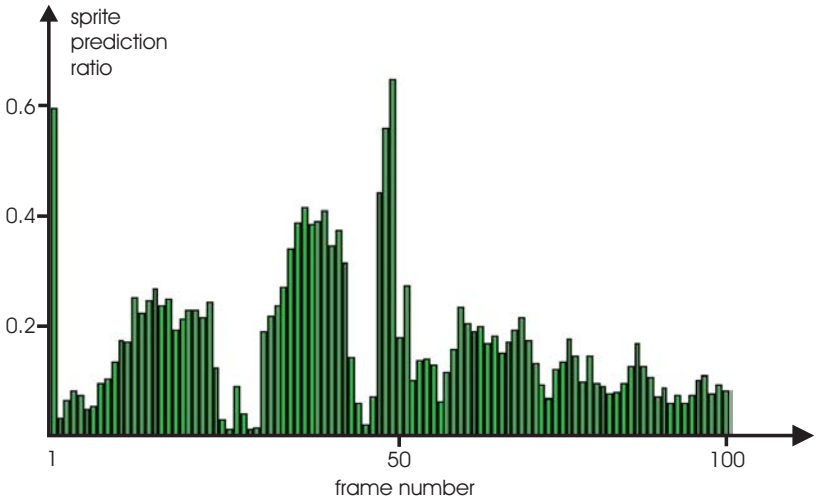


Figure 7.9: Ratio of macroblocks per frame being predicted in sprite mode for first 100 frames of sequence "Stefan" (Sequence format: IPPP...). Source: [63]

foreground object is always optimally predicted in another mode. Thus, the big merit of this sprite coding approach is that segmentation is not necessary at all. This saves computational load and makes the technique very attractive for universal usage.

The ratio of sprite predicted macroblocks in a frame strongly depends on the compression loss of the sprite image, i.e., the quantization parameter for the still image coder, and on the selected quantization parameter of the extended coder. If a fine quantization for the coder loop is chosen and the quantization of the sprite image is rather rough, other prediction modes will win in the rate-distortion optimization process. As a thumb rule the sprite quantization should always be of a higher quality than the desired video quantization. In Figure 7.9 the ratio of sprite prediction mode for test sequence "Stefan", applying a sequence format of a first I-frame followed by P-frames, is shown. For the I-frame (first frame) the sprite prediction is superior for most macroblocks except for homogeneous regions. Thus, a ratio of more than 60% can be achieved. In average the sprite prediction ratio in this example is about 15% to 20%. If the bit savings by choosing sprite mode exceeds the size

of the compressed sprite image plus warping parameter it is likely to obtain a coding gain over the H.264/AVC coder.

7.2.1.7 Warping Parameter Transmission

The sprite transformation parameters also known as warping parameters have different influences on the transformation accuracy according to their position in the model. We mainly use the perspective transformation for coding experiments since it nearly achieves the registration quality of nonlinear models but is easily invertible and thus consumes less computation power. The perspective model can be described by

$$x' = \frac{a_0 + a_1 x + a_2 y}{1 + c_1 x + c_2 y} \tag{7.7}$$

$$y' = \frac{b_0 + a_1 x + b_2 y}{1 + c_1 x + c_2 y}. \tag{7.8}$$

The parameters c_1 and c_2 in the denominator of both equations have much higher influence to the motion accuracy of the model. Therefore, a quantization of these coefficients has to be conducted with higher accuracy than for the other parameters. Also the influence of translation parameters a_0 and b_0 differs from the scaling and shearing parameters a_1, a_2, b_1, and b_2 [17].

In order to equalize the quantization of the warping parameters we chose another description as already used in the MPEG-4 Visual standard [163]. Since every perspective transformation is uniquely defined by four point correspondences, we only need to transmit the position of the corner points of every frame. The transform accuracy is high enough by transmitting those positions up to the second decimal place, which can be coded in one byte. We utilize fixed point representation. Therefore, every dimension – x or y – of the four corner positions can be represented by three bytes. In [17] it was shown that the whole parameter set can be represented by 12 bytes per frame without applying any compression technique. The portion of the parameters in the overall bit rate is rather small. Thus 24 bytes per frame are sufficient and the additional parameter frame rate R_P is calculated to

$$R_P = 24 \cdot 8 \cdot framerate, \qquad \text{with} \quad [R_P] = \frac{bit}{s}. \tag{7.9}$$

7.2.1.8 Coding Results

For experiments we configured the H.264/AVC basis encoder to use the context-based adaptive variable length coder (CAVLC) as entropy coder. It makes it easier to implement a new macroblock type, i.e., the sprite prediction mode, because it uses simpler Exp-Golomb codes for non-residual data. Since we want to be as close as possible to the original standard, we did not exchange the order of the prediction modes. Thus, the sprite prediction mode is entropy coded with a rather long code word, which has an impact on the rate-distortion optimization. Since the probability of a macroblock being predicted in sprite mode in the

very low bit rate domain is very high (depending on the chosen quantization), a shorter code word would be more appropriate. Thus, the results represent a lower bound of what can be achieved including the new sprite prediction mode. As measure of quality we use the luma *peak-signal-to-noise-ratio* (PSNR) per frame, which is defined for 8 bit quantized data as follows

$$PSNR = 10 \log \frac{255^2}{\frac{1}{M \cdot N} \sum_{u=1}^{M} \sum_{v=1}^{N} \left(I(u,v) - I_c(u,v)\right)^2}, \tag{7.10}$$

where M and N represent horizontal and vertical dimensions of the sequence and I and I_c represent the luma values of the original sequence and the coded and decoded sequence, respectively.

Figures 7.10 and 7.11 depict the average objective quality of two sequences over the achieved bit rate. No bit rate control was adopted in the experiments. The selection of operating points only depends on the chosen QPs for the sprite image and the presented coder. The format sequence was chosen to be $IBBPBBP$.... It can be seen that only for sequence "Stefan" we can achieve slightly better results than the compared H.264/AVC codec with similar settings in the range between 350 kbit/s and 800 kbit/s. The maximum gain in PSNR is about 0.4 dB. For sequence "Biathlon" we can only achieve the same performance as the basis encoder in the range of 220 kbit/s. It is important to know that for high bit ranges (very high video coding quality) the performance is always worse than the H.264/AVC standard. Since the ratio of blocks being predicted in sprite mode goes toward zero for fine quantization, we always obtain a higher rate at same quality due to the additional data, e.g., the sprite image and the warping parameters.

Another phenomenon is the difference between objective and subjective evaluation for sprite coded sequences. A lot of loss in SNR-based measures results from the re-projection error of the sprite content. Since this error cannot be noticed by a human observer, the subjective comparison can have better results. Figure 7.12 shows exemplarily frames of "Stefan" in a bit range where objectively sprite coding yields worse results. In the subjective comparison the results are very similar or even better (see letters in the advertisements in sequence "Stefan").

In Sections 7.2.3 and 7.2.4 we will show that the gain using the sprite prediction mode can be significantly improved applying extended sprite construction techniques, i.e., super-resolution sprites and multiple sprites, which were presented in the previous chapters.

7.2.2 Object-based Video Coding using H.264/AVC and Explicit Segmentation

The system we present in this section is rather close to the object-based sprite coding technique depicted in Figure 7.4. The main difference to MPEG-4 Part 2 is the embedded segmentation process, which generates the video objects prior to coding. The basic encoder

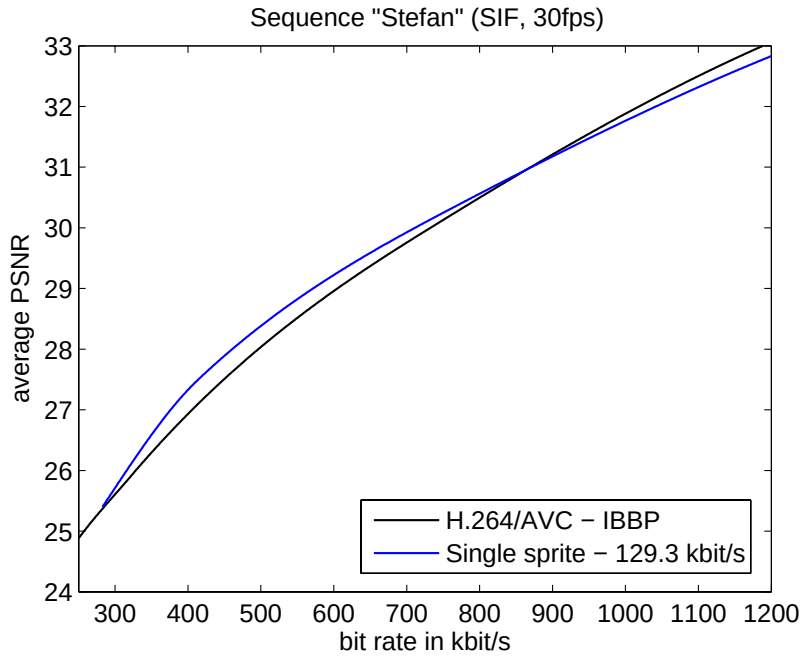


Figure 7.10: Coding results for sequence "Stefan" using the presented SPAVC mode in comparison to the standard H.264/AVC performance (IBBP, CAVLC). Several sprite image compression levels have been tested. The presented curve at 129.3 kbit/s was empirically found to have the best performance.

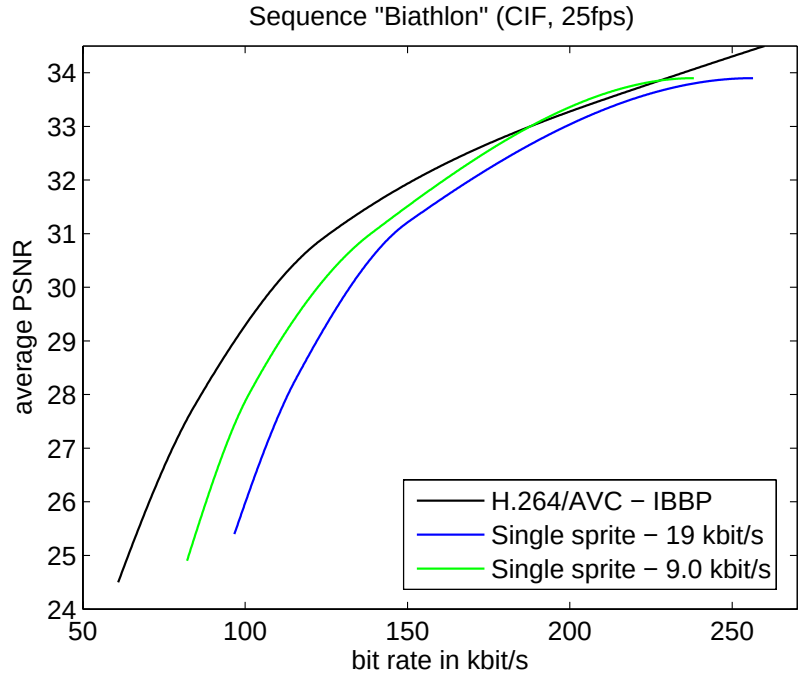


Figure 7.11: Coding results for sequence "Biathlon" using the presented SPAVC mode in comparison to the standard H.264/AVC performance (IBBP, CAVLC). Several sprite image compression levels have been tested. The presented curves at 19.0 kbit/s and 9.0 kbit/s were empirically found to have the best performance.

(a) H.264 at 278.16 kbit/s - frame 1

(b) SPAVC at 282.7 kbit/s - frame 1

(c) H.264 at 278.16 kbit/s - frame 100

(d) SPAVC at 282.7 kbit/s - frame 100

(e) H.264 at 278.16 kbit/s - frame 200

(f) SPAVC at 282.7 kbit/s - frame 200

(g) H.264 at 278.16 kbit/s - frame 300

(h) SPAVC at 282.7 kbit/s - frame 300

Figure 7.12: Subjective comparison for exemplary frames of sequence "Stefan" for H.264/AVC and SPAVC at similar bit rates.

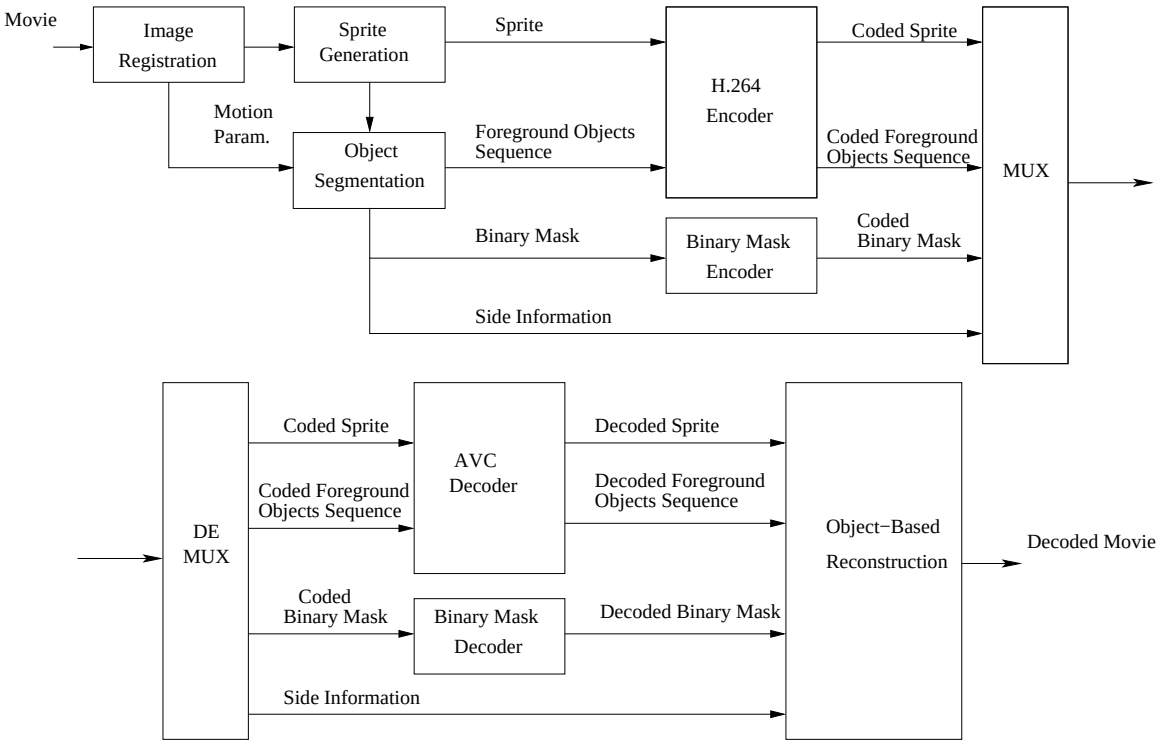


Figure 7.13: Encoder (upper) and decoder (lower) of the objec-based video codec (OBVC) using sprites and explicit object segmentation.

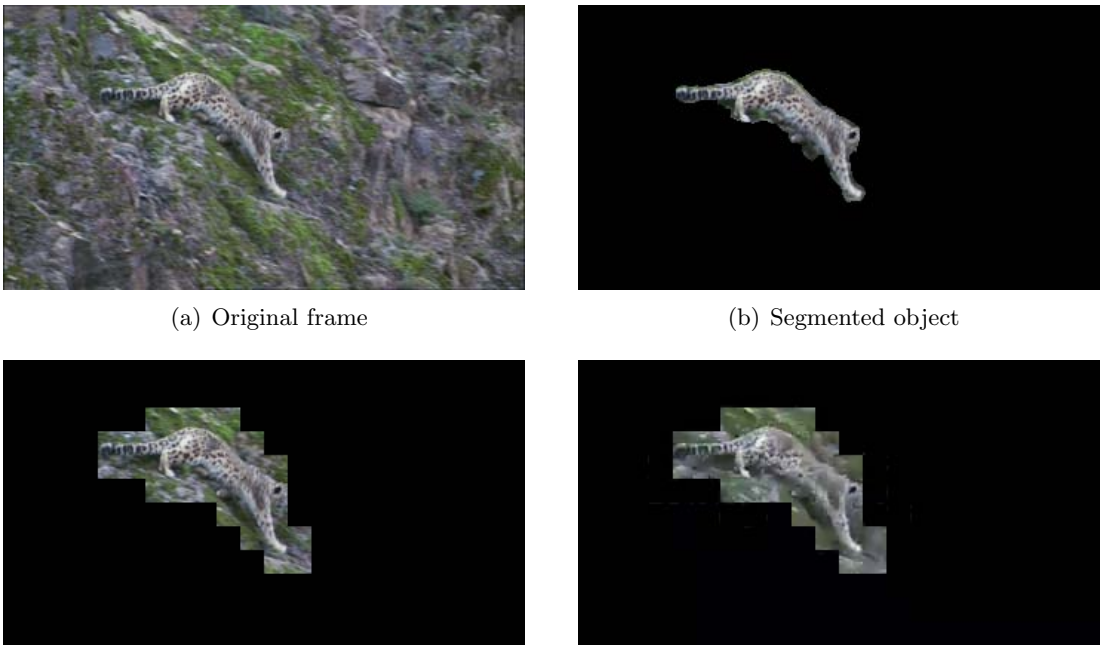
and decoder of our *object-based video codec* (OBVC) is shown in Figure 7.13 [70], [65]. Main coding units of this codec are the H.264/AVC and a binary mask coder. Sprite image S and the segmented foreground objects sequence are transmitted using H.264 intra coding and classical H.264 coding, respectively. Actually, the coding approach is not restricted to this standard. Each available hybrid video codec coding rectangular video frames can be used. The generation of the foreground objects and the fitting into the 16x16 block raster will be explained later.

For the coding of the binary mask a context-based adaptive binary coder [82] with eight contexts according to the state of previously coded pixels at the same time instance is used (see Section 7.2.2.2). Warping parameters are transmitted with 3 bytes per parameter in the same way as shown in Section 7.2.1.7.

At the decoder side all decoded information is used to recombine the background from the sprite and the foreground objects utilizing the binary mask. The rate-distortion behavior is mainly influenced by the quantization quality of the sprite image and the foreground sequence.

7.2.2.1 Object Generation and Coding

The segmentation of foreground objects is achieved using the algorithms presented in Chapter 6. Depending on the spatial proportion of the foreground object in the frame, huge



(a) Original frame (b) Segmented object

(c) Fitted to 16x16 raster (d) Object after decoding

Figure 7.14: Example for object segmentation, raster fitting, and object coding for frame 20 of sequence "Mountain".

parts of the objects sequence remain without information and thus can be coded very efficiently. Since the prediction modes in H.264/AVC are processed on a block raster, coding of arbitrarily shaped objects may have large impact on the image quality. This is especially true for the contour regions of the objects. In order to minimize the degradation due to block-based processing we fit the objects to the 16x16 macroblock raster of the frame. See an example for the objects processing in Figure 7.14.

The H.264/AVC standard also provides a *loop filter*. This filter smoothes the block borders to minimize the block effects caused by the quantization of the transform coefficients. Using the loop filter for the fitted objects sequence (block objects in a black frame) will cause a diffusion of the surrounding color into the object. Therefore, before re-combining all information we erode the objects sequence by two pixels. Since this could lead to loss of information in cases where real objects and the fitted raster have the same border pixels, a dilation has to be applied prior to the fitting process.

7.2.2.2 Binary Mask Coding

As binary mask coder we use a context-based adaptive binary coder with 8 contexts implemented as *modulo coder* (M coder) as it is used for CABAC in the H.264 reference encoder software. Since the mask data is already in a binary form, we can leave out the process of binarization as described in [82]. For any pixel to code the context is defined by the state of the 3 adjacent pixels that have already been coded. In Figure 7.15 the mask pixels

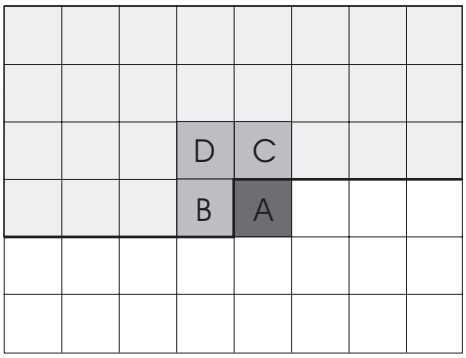


Figure 7.15: Pixel that define the context for binary mask coding: In order to predict the state probabilities of binary pixel A, pixels B, C, and D define 8 contexts (3 bit) for which the the probabilities are known and adapted during the process.

that define the context are shown. The symbol probabilities are estimated adaptively after encoding each symbol, whereas only 128 different probability states are possible. See [82] for an in-depth description of context-based adaptive binary arithmetic coding.

7.2.2.3 Coding Results

As no manipulation of the basis encoder is necessary we configured H.264/AVC to perform with a maximum efficiency. Hence, we used CABAC as entropy coder and applied a sequence format containing hierarchical B pictures [119] with a group-of-picture (GOP) length of 15 frames. Instead of the typical configuration using $IBBP$, hierarchical B picture are bi-predictive frames that predict other B pictures of the GOP in a tree-based hierarchical structure. Using GOP size 15 leads to 4 hierarchical levels plus a first I-frame. Assuming a frame rate of 30 fps, we achieve direct access of 0.5 seconds. The curves showing the average picture quality over the sequence frame rate are compared to the standard H.264/AVC using the same configuration.

Figures 7.16 and Figures 7.17 show the coding results using the OBVC with embedded explicit segmentation for sequences "Stefan" and "Biathlon". We found out that only in the very low bit rate domain (strong compression of the sprite image) we can achieve a gain over the standard coding approach for sequence "Stefan". Here, the bit rate saving is maximal 25.6% at a quality level of 23.8 dB. This is equivalent to a gain of 1.2 dB at 90 kbit/s. It can also be observed that the R-D curves are quickly saturating towards higher bit rates. This is caused by the maximal achievable PSNR value due to missing residual data for the sprite background. Upper limits for the reconstruction are 25.1 dB for "Stefan" and 30.91 dB for "Biathlon", respectively.

A subjective comparison is shown in Figures 7.18 and 7.19. While for "Stefan" the sprite-based approach is always superior, we experience a decreasing quality over the sequence for OBVC. This can be explained by the zooming character of the sequence. The background of the last frames is an upsampled version of the transformed sprite background. This

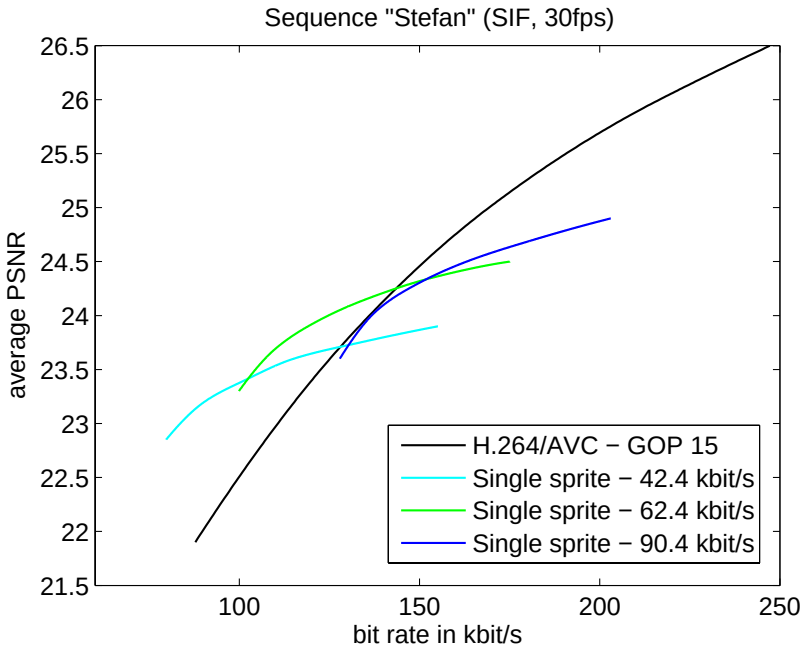


Figure 7.16: Coding results for sequence "Stefan" using the presented OBVC with explicit object segmentation in comparison to the standard H.264/AVC performance (hierarchical B frames, CABAC). Only for low bit rates a performance gain can be achieved.

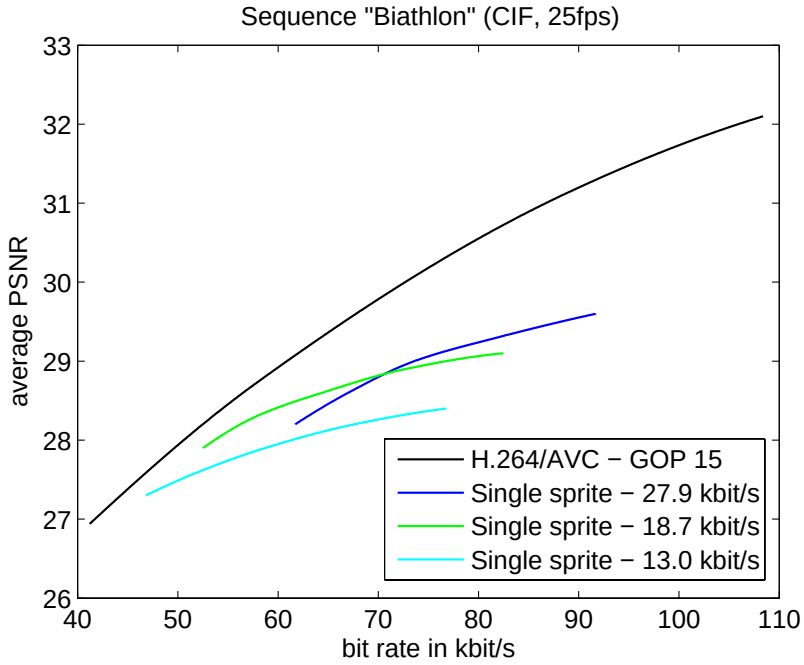


Figure 7.17: Coding results for sequence "Biathlon" using the presented OBVC with explicit object segmentation in comparison to the standard H.264/AVC performance (hierarchical B frames, CABAC). No performance gain was achieved applying the sprite-based method.

could be prevented by selecting the last frame as reference frame. However, the sprite size would increase tremendously. Therefore, for the first half of "Biathlon" OBVC yields better subjective results while for the second half standard H.264/AVC takes the lead. Note, that objectively H.264/AVC has better quality performance.

In order to improve the quality and the performance of both presented sprite-based coding approaches, we exploit the usefulness of applying super-resolution sprites and coding with multiple sprites in the next sections.

7.2.3 Super-resolution Sprite Coding

Generating the sprites using super-resolution techniques described in Chapter 4 we obtain sprite images of much bigger image size. Compressing these images we have shown that the coding cost grows less than linear with the image area. Therefore the question is whether the higher reconstruction quality of SR sprite compensates for higher memory cost in terms of R-D performance.

In Figures 7.20 and 7.21 we compare the coding results achieved using the SR sprite of sequence "Stefan" with results using the LR sprite. The SR sprite was generated with double magnification in each dimension, i.e., four times the size of the LR sprite. For the SPAVC approach (Figure 7.20) the curves using SR sprites exceed the standard H.264/AVC for a much wider area of bit rate. The maximal bit rate saving is up to 11.3% at a 25.8 dB PSNR level against H.264. Against the usage of the LR sprite the saving is still about 9%. This is equal to a maximal gain of 0.55 dB.

The gain we are able to achieve is even higher applying the object-based approach OBVC (Figure 7.21). The quality exceeds the H.264/AVC coding by 1.1 dB to 1.5 dB over the whole area between 50 kbit/s and 200 kbit/s depending on what quality the sprite image was coded. Thus SR sprite generation enables the codec to improve the performance over a broad bandwith.

Subjective comparisons for both coding methods applying super-resolution sprites can be found in Figures 7.22 and 7.23. In both examples the subjective background quality is much better. In several frames the foreground object has slightly worse quality using SR sprite coding techniques. Depending on the focus and the experience of the observer this can influence the overall subjective video quality.

7.2.4 Multiple Sprite Coding

Utilizing multiple sprites rather than a single sprite should improve the coding performance due to a minimization of the memory costs for the sprite images. For the generation of multiple sprites we applied the algorithms presented in Chapter 5. Both sprite coding approaches where utilized to encode the test sequences. Since multiple sprites have not been

(a) H.264 at 87.8 kbit/s - frame 1

(b) OBVC at 90.3 kbit/s - frame 1

(c) H.264 at 87.8 kbit/s - frame 100

(d) OBVC at 90.3 kbit/s - frame 100

(e) H.264 at 87.8 kbit/s - frame 200

(f) OBVC at 90.3 kbit/s - frame 200

(g) H.264 at 87.8 kbit/s - frame 295

(h) OBVC at 90.3 kbit/s - frame 295

Figure 7.18: Subjective comparison for exemplary frames of sequence "Stefan" for H.264/AVC and OBVC at similar bit rates.

(a) H.264 at 57.9 kbit/s - frame 1

(b) OBVC at 57.8 kbit/s - frame 1

(c) H.264 at 57.9 kbit/s - frame 50

(d) OBVC at 57.8 kbit/s - frame 50

(e) H.264 at 57.9 kbit/s - frame 100

(f) OBVC at 57.8 kbit/s - frame 100

Figure 7.19: Subjective comparison for three frames of sequence "Biathlon" coding with H.264/AVC and OBVC at similar bit rates.

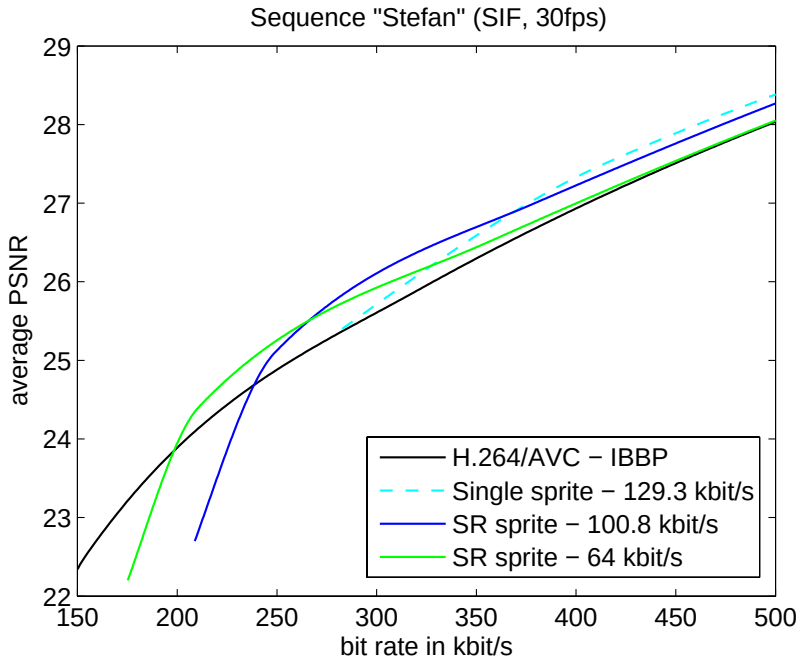


Figure 7.20: Coding results for super-resolution sprite coding of sequence "Stefan" using the presented SPAVC (sprite prediction mode) in comparison to the standard H.264/AVC performance (IBBP, CAVLC) and low resolution sprite coding (single). Interestingly, a stronger compression of the SR sprite image yields better results than a less strong compression of the LR sprite.

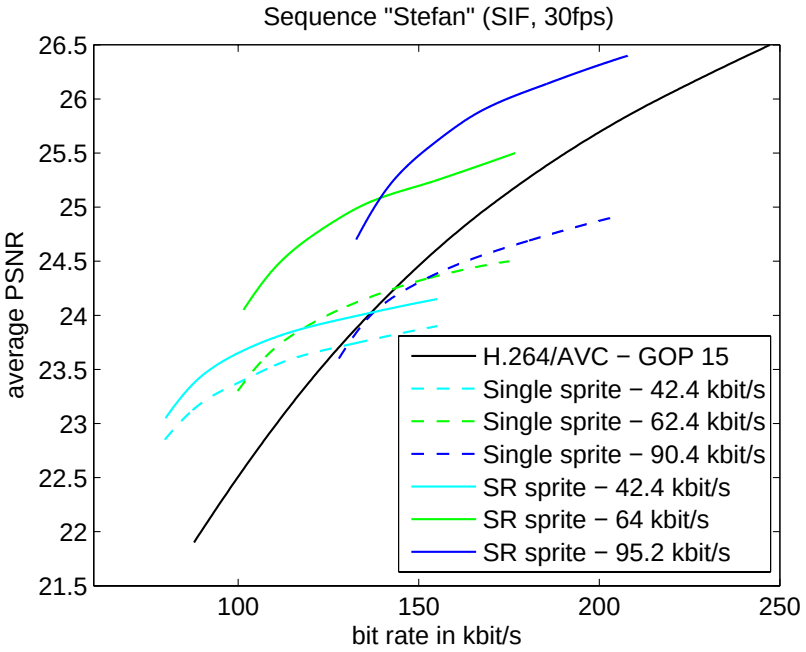


Figure 7.21: Coding results for super-resolution sprite coding of sequence "Stefan" using the presented OBVC in comparison to the standard H.264/AVC performance (hierarchical B frames, CABAC) and low resolution sprite coding (single).

(a) H.264 at 379.38 kbit/s - frame 1

(b) SPAVC at 379.3 kbit/s - frame 1

(c) H.264 at 379.38 kbit/s - frame 100

(d) SPAVC at 379.3 kbit/s - frame 100

(e) H.264 at 379.38 kbit/s - frame 200

(f) SPAVC at 379.3 kbit/s - frame 200

(g) H.264 at 379.38 kbit/s - frame 300

(h) SPAVC at 379.3 kbit/s - frame 300

Figure 7.22: Subjective comparison for exemplary frames of sequence "Stefan" coded with H.264/AVC and SPAVC using SR sprites at similar bit rates.

(a) H.264 at 130.5 kbit/s - frame 1

(b) OBVC at 123.4 kbit/s - frame 1

(c) H.264 at 130.5 kbit/s - frame 100

(d) OBVC at 123.4 kbit/s - frame 100

(e) H.264 at 130.5 kbit/s - frame 200

(f) OBVC at 123.4 kbit/s - frame 200

(g) H.264 at 130.5 kbit/s - frame 295

(h) OBVC at 123.4 kbit/s - frame 295

Figure 7.23: Subjective comparison for exemplary frames of sequence "Stefan" coded with H.264/AVC and OBVC using SR sprites at similar bit rates.

implemented into the sprite prediction mode technique (SPAVC), for each sub-sequence an independent bitstream was created containing its own header information. Thus, for very short sequences the gain using multiple sprites can be lost because of the additional header information. For the object-based coding approach (OBVC) multiple sprites should not generate extra data to transmit.

We examined the effects of multiple sprite coding for two sequences, "Stefan" and "Biathlon". Figures 7.24 and 7.25 show the PSNR values over the bit rate of "Stefan" for both coding approaches using multiple sprites. The curves for the singles sprites are shown in the same diagram. For the SPAVC we gain about 0.7 dB over H.264/AVC for the whole measured bit rate range, whereas the single sprite approach exceeded the H.264/AVC for only a small range about 500 kbit/s. For the object-based approach with explicit segmentation (OBVC) the gain using multiple sprites over standard coding is up to 1.1 dB. Here, the maximum gain over single sprite coding is 0.7 dB, which corresponds to the value achieved in Chapter 5, where only the background was coded (Section 5.3.2).

For sequence "Biathlon" only the object-based approach achieves an improvement. The utilization of multiple sprites enables the codec to perform better than the standard. Since the SPAVC is not able to handle multiple sprites in one coding loop, the redundant header information negates the effect of memory saving. Thus, no performance gain can be achieved.

For subjective evaluation example frames of sequence "Biathlon" for both approaches are shown in Figures 7.28 and 7.29. As already stated, the quality for the object-based sprite coding approach decreases with increasing focal length of the camera. Utilizing multiple sprites lessens this effect but it still can be observed.

7.2.5 Comparison of Strategies

In order to compare the behavior of both sprite-based coding approaches, SPAVC and OBVC, we have to level the base encoder. For the experiments CAVLC as entropy encoder and a sequence format of $IBBP$ was applied. With this configuration H.264/AVC does not perform optimally but the tendency remains independently from the settings.

First, we measured the performance for a test sequence without any foreground objects. This shows what maximum improvement over the standard is achievable and also reveals the differences between the two strategies. Figure 7.30 depicts the R-D curves for sequence "Monitor". The compression gain over standard H.264 coding is very remarkable. Especially the gain applying OBVC in terms of bit rate savings is very huge because no foreground data has to be coded. The diagram shows as well the demerit of the OBVC. The objective reconstruction quality is limited by the quality achieved with the uncoded sprite. This limit is shown as horizontal dashed line. Here, the SPAVC approach is more flexible since it is able to code the background residual. In the diagram also the suitable bit rate ranges for both approaches are given. While OBVC performs better for very low bit rates, the SPAVC

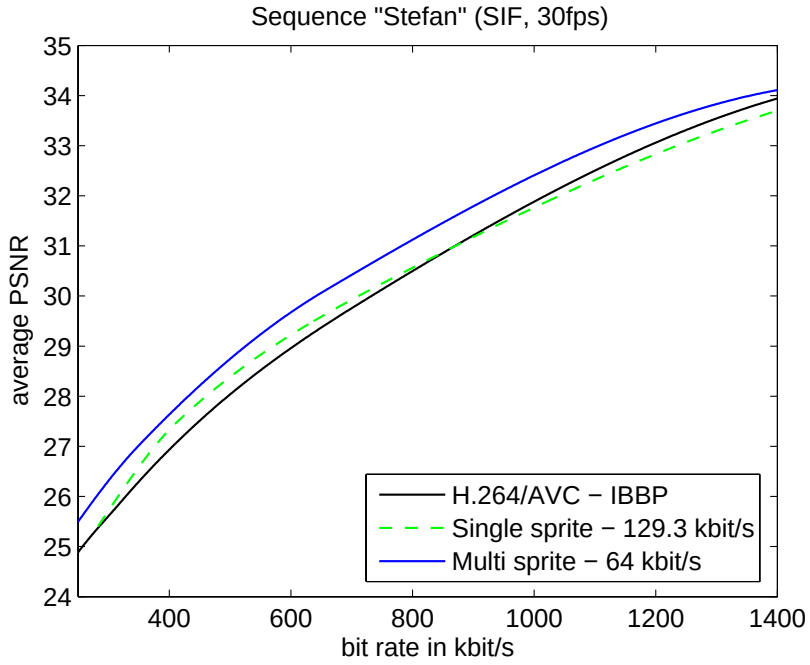


Figure 7.24: Coding results for multiple sprite coding of sequence "Stefan" using the presented SPAVC (sprite prediction mode) in comparison to the standard H.264/AVC performance (IBBP, CAVLC) and the single sprite approach.

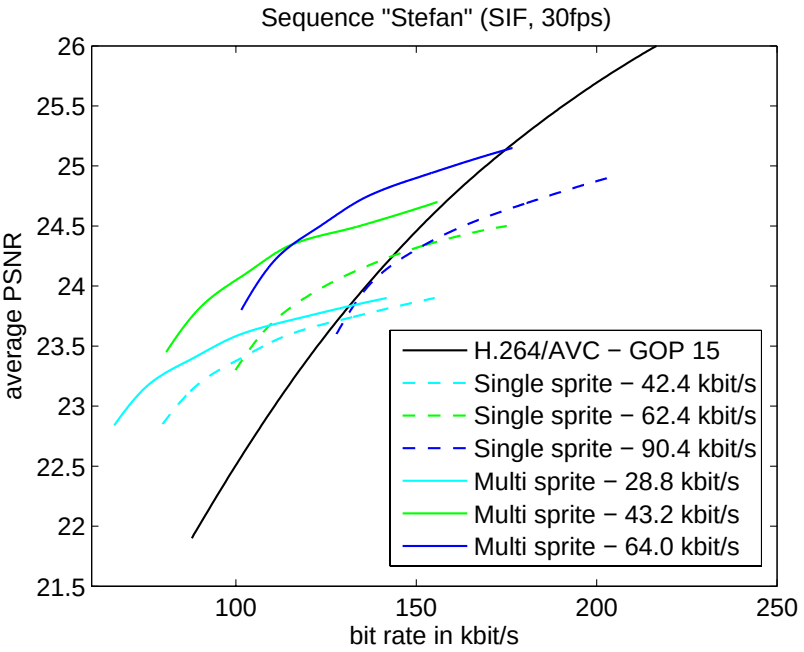


Figure 7.25: Coding results for multiple sprite coding of sequence "Stefan" using the presented OBVC in comparison to the standard H.264/AVC performance (hierarchical B frames, CABAC) and the single sprite approach.

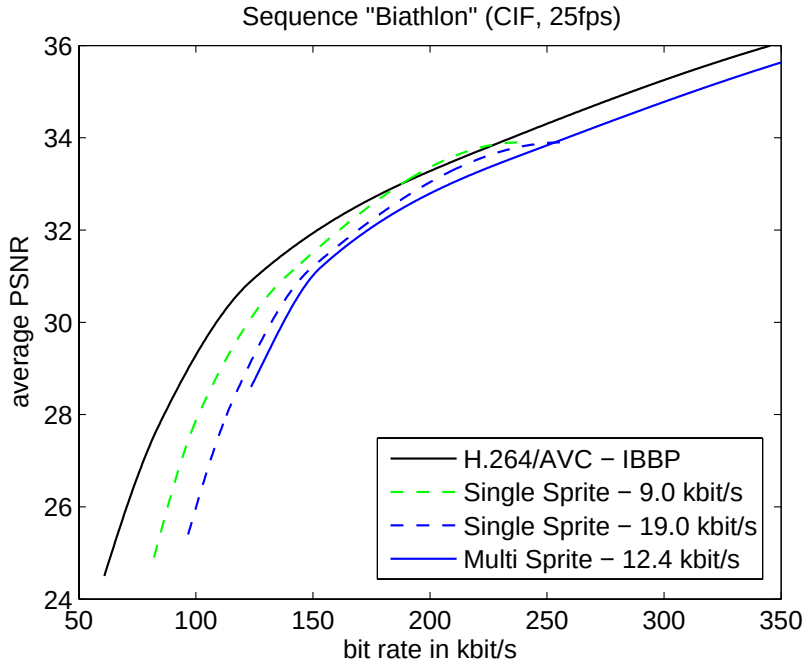


Figure 7.26: Coding results for multiple sprite coding of sequence "Biathlon" using the presented SPAVC (sprite prediction mode) in comparison to the standard H.264/AVC performance (IBBP, CAVLC), and the single sprite approach.

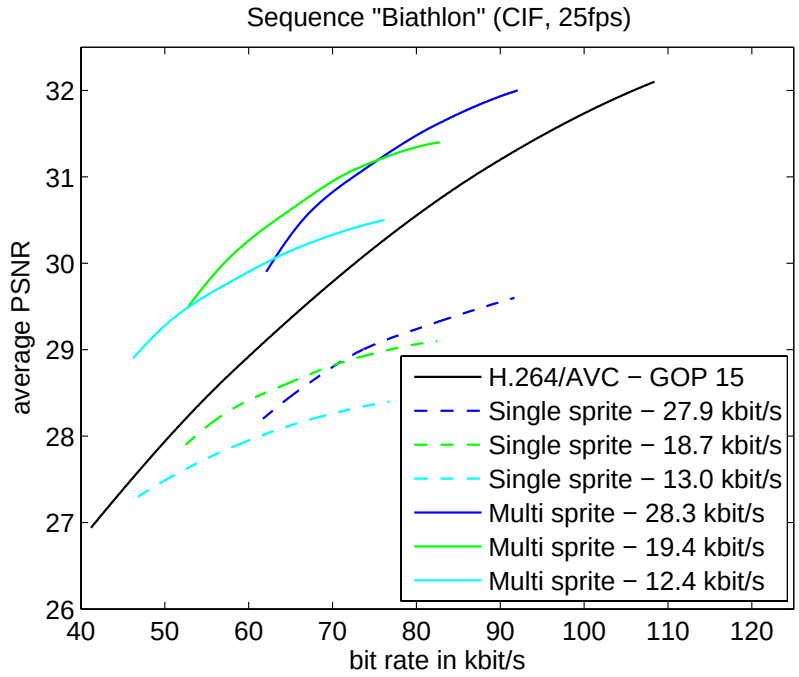


Figure 7.27: Coding results for multiple sprite coding of sequence "Biathlon" using the presented OBVC in comparison to the standard H.264/AVC performance (hierarchical B frames, CABAC) and the single sprite approach.

(a) H.264 at 257.5 kbit/s - frame 1

(b) SPAVC at 253.9 kbit/s - frame 1

(c) H.264 at 257.5 kbit/s - frame 100

(d) SPAVC at 253.9 kbit/s - frame 100

(e) H.264 at 257.5 kbit/s - frame 200

(f) SPAVC at 253.9 kbit/s - frame 200

Figure 7.28: Subjective comparison for three frames of sequence "Biathlon" coding with H.264/AVC and SPAVC using the multiple sprite approach at similar bit rates.

(a) H.264 at 57.9 kbit/s - frame 1

(b) OBVC at 58.1 kbit/s - frame 1

(c) H.264 at 57.9 kbit/s - frame 100

(d) OBVC at 58.1 kbit/s - frame 100

(e) H.264 at 57.9 kbit/s - frame 196

(f) OBVC at 58.1 kbit/s - frame 196

Figure 7.29: Subjective comparison for three frames of sequence "Biathlon" coding with H.264/AVC and OBVC using the multiple sprite approach at similar bit rates.

achieves a gain for slightly higher bit rates. The disadvantage of the SPAVC is a rather small gain compared to OBVC. The very balancing rate-distortion optimization controlling the prediction and the rather long word length for the sprite prediction mode using Exp-Golomb codes prevents the codec to perform like OBVC.

In a second experiment, examining a sequence with foreground objects, the already acquired results are approved (Figure 7.31). The OBVC is also limited in terms of reconstruction quality at a level of 30.6 dB, whereas the SPAVC approach achieves performance improvement over standard H.264/AVC above this limit. On the other hand, OBVC can achieve higher bit rate savings for very low bandwiths – up to 78.9% at a poor quality of 23.7 dB PSNR. The maximum quality gain is up to 1.9 dB. A subjective comparison is given in Figures 7.32 and 7.33. The sequence were coded at similar bit rates of about 170 dB. While standard H.264/AVC achieves a PSNR of 26.1 dB, SPAVC yields 27.3 dB, and OBVC yields 28.6 dB. This tendency can also be observed in the sample images.

Finally we compare the maximum bit rate savings for all conducted experiments in Table 7.1. Note, that not always the same basis encoder configuration was used. Since the bit rate saving is a relative measure, always compared to the standard with same coding configurations, the results can be used to testify the impressive performance of both sprite coding approaches.

	Sequence			
Coding approach	"Stefan"	"Biathlon"	"Monitor"	"Mountain"
SPAVC - single sprite	8.8%	0%	20.8	25.3
SPAVC - SR sprite	11.3%	-	-	-
SPAVC - multiple sprites	14.3%	-7.4%	-	-
OBVC - single sprite	25.6%	-5.0%	93.1%	78.9%
OBVC - SR sprite	27.5%	-	-	-
OBVC - multiple sprites	37.3%	23.1%	-	-

Table 7.1: Maximum bit rate savings against the H.264/AVC basis encoder for different sprite coding approaches and different sequences.

7.3 An Universal Sprite Coding Approach - Outlook

As stated in the introduction new coding tools will become important if, in addition to a better performance, universal usability and a minimal compatibility to already existing codecs can be provided. Therefore, we present an sprite coding idea that comprises all these requirements. In Figure 7.34 the main units of such a codec are shown. Beginning with a typical shot boundary detection [74] that classifies a sequence into several shots the system determines for each shot whether sprite construction is possible and sprite coding should be applied. If the sprite coding is selected, multiple super-resolution sprites are generated and stored in a database. In many scenes, e.g., sport videos, we find a recurrent background.

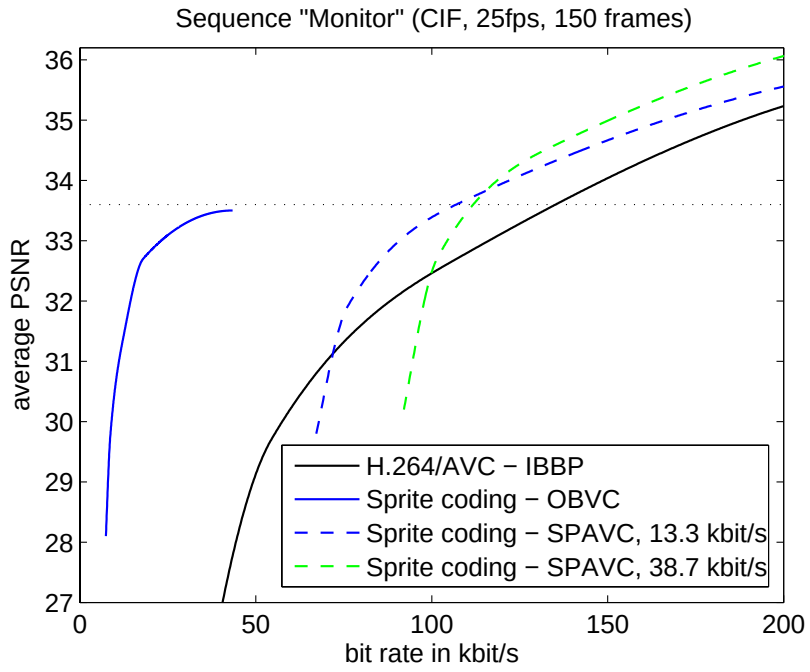


Figure 7.30: Comparison of coding results for sequence "Monitor" (no foreground objects) using classical H.264/AVC, SPAVC and OBVC. The basis encoder was configured to utilize CAVLC as entropy coder and a sequence format of $IBBP$.

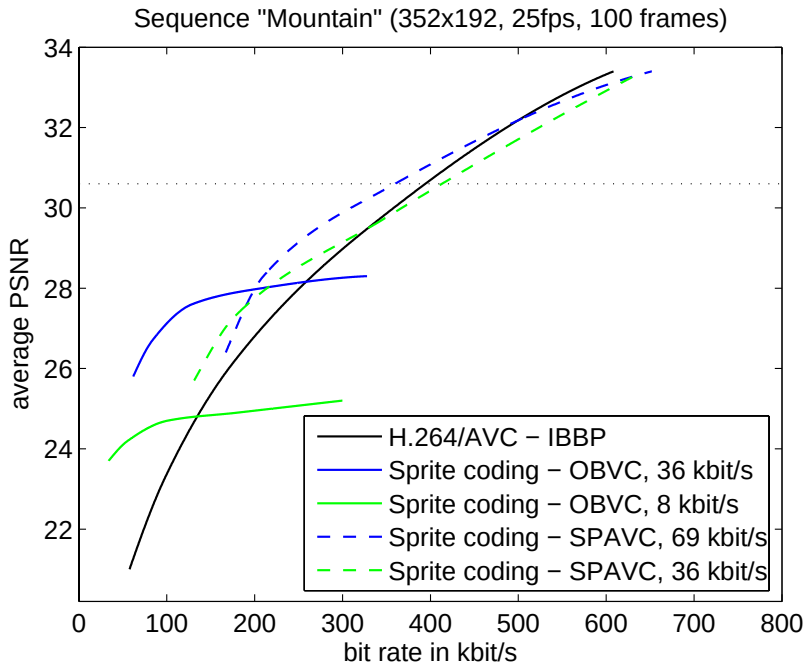


Figure 7.31: Comparison of coding results for sequence "Mountain" using classical H.264/AVC, SPAVC and OBVC. The basis encoder was configured to utilize CAVLC as entropy coder and a sequence format of $IBBP$.

(a) Original sequences

(b) H.264/AVC at 174.0 kbit/s

(c) SPAVC at 177.5 kbit/s

(d) OBVC at 161.8 kbit/s

Figure 7.32: Subjective comparison for frame 40 of sequence "Mountain" (352x192, 100 frames, 25fps) applying both presented sprite-based video coding techniques at similar bit rates.

(a) Original sequences

(b) H.264/AVC at 174.0 kbit/s

(c) SPAVC at 177.5 kbit/s

(d) OBVC at 161.8 kbit/s

Figure 7.33: Subjective comparison for frame 99 of sequence "Mountain" (352x192, 100 frames, 25fps) applying both presented sprite-based video coding techniques at similar bit rates.

Thus, in the best case one sprite background can be used for multiple video shots. Therefore, a sprite recognition unit is introduced into the scene. This can simply be a recognition of a fixed camera or a real image-based analysis tool. In case a background is already existent no transmission is necessary. Additionally, the background can be enhanced if new areas are discovered. The units that are achieved by this or previous work are marked in Figure 7.34. As sprite-based coder both presented approaches are possible. Since the OBVC has better performance for very low bit rates, it is preferred. Also an optimization of the sprite-based codecs has not been performed yet. It is highly desirable to determine the best coding performance depending on the compression of the sprite image and a required bit rate.

7.4 Chapter Summary

In this chapter we have presented two new approaches for sprite-based video coding without a need for providing already segmented video objects. The first method, SPAVC, is based on the rate-distortion optimization process of the H.264/AVC reference encoder. It provides a new prediction mode on macroblock size for the H.264 standard. Due to the rate-distortion optimization, only precisely predictable regions are selected to be coded in the new mode. Thus, an inherent segmentation process selects different prediction modes for the foreground regions. The maximal bit rate savings we achieved during the experiments was 25.3%. Due to automatically encoding of the background residual, there is no upper limit for the reconstructed sequence at the decoder side. As we have shown, for high bit rates (high video quality) the SPAVC approach has always worse results than H.264/AVC.

The second sprite-based coding technique explicitly uses foreground segmentation prior to coding. It is therefore an object-based coding approach (OBVC). Since the segmentation is as well based on the generated sprite background, the additional computational load is not very high. The segmentation procedures that were utilized, have been presented in this thesis (Chapter 6). In this approach the residual for the background, i.e., the difference between original image and sprite-based prediction, was not send to the decoder. Thus, the re-projection error sets a limitation for the objective reconstruction quality. However, for sequences containing foreground objects the bit rat savings are up to 78.9% at poor quality levels.

In our experiments we have shown that both approaches have the potential to outperform classical hybrid video codecs, e.g., the latest standard H.264/AVC. The behavior of both approaches in terms of rate-distortion performance is in some way complementary. While the OBVC has high performance gain for lowest bit rates, the SPAVC has its best performance at higher bit rates and is not limited in terms of objective video quality measures. However, both approaches only outperform standard coding for rather low bandwiths.

The particular performance of a specific codec strongly depends on the quality and the memory cost for the sprites themselves. Hence, we applied super-resolution sprite coding as

well as multiple sprite coding in order to improve quality or lower the memory costs. We have shown that both techniques improve the coding performance significantly. For each method we improved the codec performance by values of greater than 1[dB] depending on the sequence and the chosen bit rate, respectively. A combination of both presented techniques should even yield a better performance, but comes along with very high computational costs.

An important issue for the success of a coding strategy is not only the gain in objective measures but also a good subjective evaluation performance. Here, two facts have to be mentioned. First, in most cases the quality of the video background is even better than the objective measure indicates. Second, for many subjects it is not acceptable to have lower quality for the foreground objects. This means that even with better results in terms of a PSNR-measure the sprite coded video might be evaluated with a lower subjective score due to differences between background and foreground objects. Thus, it is very important to shift the gained bandwith from the background coding into the foreground. In this chapter we have shown that it is possible to achieve both, better overall performance and acceptable foreground quality. A technique to control the bandwith distribution together with a rate control is a worthwhile future research direction.

Finally, we have presented a future coding system, which dependent on the presented video sequences uses sprites for the coding process. It has the advantages of being universally applicable and is compatible to decode today's standard video bitstreams. Unfortunately, not all units of the presented systems where implemented in this work. However, important parts towards this new generation of video codecs were presented.

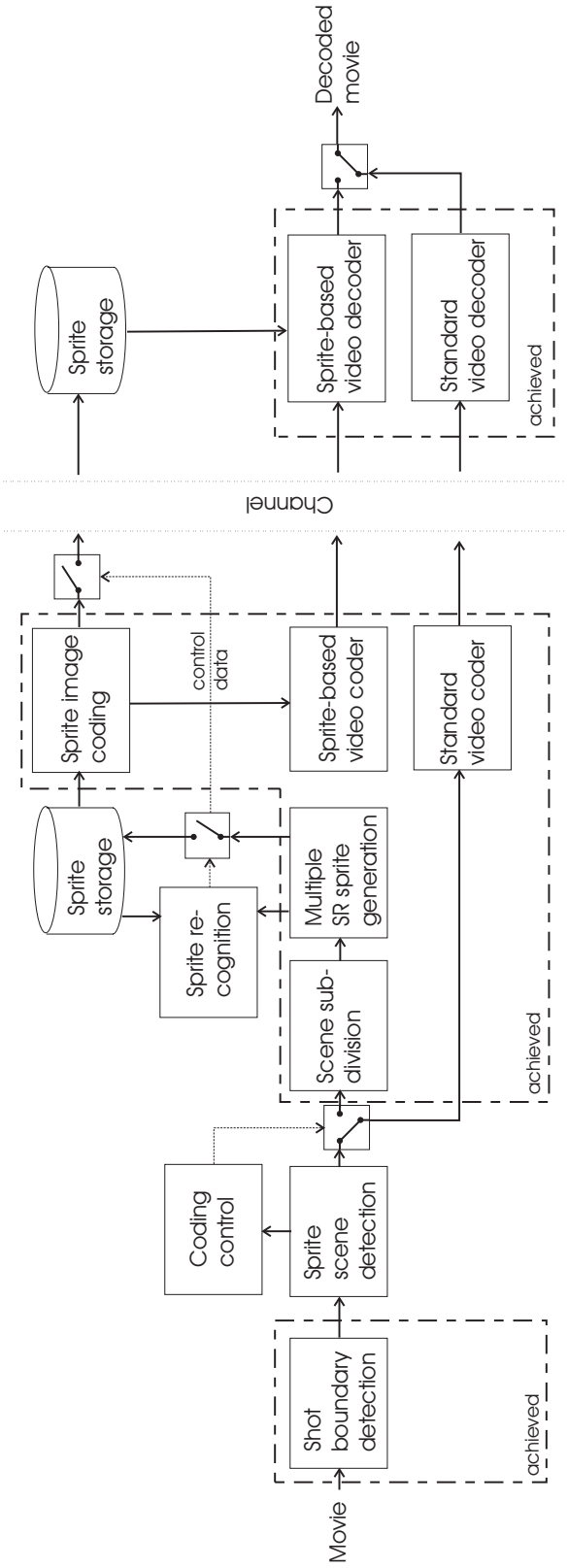


Figure 7.34: Outlook: An universally applicable video codec utilizing sprites.

Chapter 8

Conclusions and Outlook

That's the veritable miracle of technology;
it truly destroys everything it is compensating for.

Karl Kraus (1874-1936)

In this thesis, various techniques for sprite generation, sprite-based segmentation and sprite-based video coding have been presented. The achievements, the drawable conclusions, future improvements, and future research topics are pictured in the next sections.

8.1 Achievements

We presented a full system for sprite-based encoding of digital video data that outperforms conventional hybrid video codecs for rectangular frames in the low bit rate domain. In order to achieve this improvement three main processes had to be accomplished in a robust manner:

1. A reliable, very exact, and broadly applicable sprite generation.
2. A robust and reliable discrimination of camera-independently moving foreground objects from rigid background objects, i.e., a moving object segmentation method.
3. A coding system that integrates sprite-based background reconstruction and efficiently encodes and transmits the foreground video content.

For all three processes novel methods and algorithms were presented that achieve satisfying results and level or even exceed the performance of already existing techniques.

Main achievements of the sprite generation process were the introduction of a novel frame-to-mosaic registration structure with advanced pixel blending methods (Chapter 3), a super-resolution sprite construction technique (Chapter 4), and an approach for the generation of multiple sprites, that yield excellent reconstruction quality while at the same time minimizing the memory costs (Chapter 5). In order to subdivide a scene shot into several sub-shots for multiple sprites a rough and low complex camera calibration technique was

applied (Chapter 2). Thus, an estimation of the physical camera parameters during the capturing process was achieved.

For the segmentation of independently moving objects we concentrated on sprite-based segmentation techniques, since the sprite or multiple sprites are estimated anyway. Due to the re-projection of sprite content into the frame coordinate system, a good estimation of the image background is available. Therefore, we presented two segmentation approaches applying sophisticated change detection analysis on the difference image, resulting from the background subtraction for each channel (Chapter 6). In experiments we have shown that the processes determining the binary change masks are as good and reliable as actual change detection approaches for still cameras.

Finally, two novel techniques for sprite-based coding were presented (Chapter 7). This chapter constitutes the synthesis of this work, where all described and implemented techniques are combined into a system. Both coding approaches build up on the latest video coding standard H.264/AVC but the basic ideas behind do not necessarily need the standard. The fundamental handling of foreground objects and residual coding differs for both techniques and, thus, the performances of both techniques differ as well. While the object-based approach (OBVC) utilizing explicit segmentation yield strong gain for lowest bit rates, the technique introducing block-based sprite prediction (SPAVC) has better performance in a higher quality range and thus is more applicable for higher bandwiths. However, due to non-optimal implementation the second technique still has potential to yield larger improvements.

8.2 Conclusions

The methods and techniques of this thesis mark another step toward an efficient and universally applicable video codec based on background sprites. Not only their combination in a video codec but also their application as independent tools can be useful for a variety of video analysis and machine learning methods. We have shown that there is a lot more margin in the improvement of sprite reconstruction quality, especially applying super-resolution and multiple sprites construction techniques. Moreover, only these accuracy augmentation methods enable a sprite-based codec to perform better than latest complex video coding standards in terms of a rate-distortion measurement. Second, for a general acceptance of the presented coding strategies it is essential that only minimal user interaction is required for such coding systems. Therefore, necessary processes as automatic foreground segmentation and calibration-based scene partition have been proposed in this work.

Complexity efficiency was not an objective of this thesis. Therefore, in case of combining the proposed techniques, e.g., for object segmentation and sprite-based coding, the computational load for these algorithms is still high. All presented methods are away from utilization in real-time applications. Nevertheless, off-line procedures are a common way for

many video processing systems. Especially in the video coding domain it is often important that only the decoder works computational efficient. This is also the case for the presented coding approaches.

8.3 Outlook

There exist a lot of further extensions and improvements for the presented systems and algorithms concerning sprite-based techniques. Additionally, other applications are thinkable and have been implemented during the research. Unfortunately, not all of the possibilities sprite processing offers have been exploited and investigated. In the next sections, proposals for novel research activities are made that have not been considered in this thesis.

Sprite Generation

Extensions and improvements of the presented sprite generation techniques make sense in the following fields:

- Fast image registration methods utilizing phase correlation as proposed by [67] and [60].
- An optimized approach for super-resolution sprites that jointly estimates the image transformation and the best pixel values while blending.
- Incorporation of luminance equalization within an image and between adjacent images. In [65] we have shown that the coding efficiency can be improved when parametrically estimating the luminance characteristics for a certain camera.
- Consider zoom-only sequences in the multiple sprite generation process. For zoom shots the selection of the reference frame is of high importance. It is necessary to balance between good reconstruction quality – the frame with largest focal length should be selected – and the optimal sprite image sizes – the frame with smallest focal length should be selected. Under certain conditions a shot partition can lead to optimal results.

Sprite-based Object Segmentation

Optimal object segmentation is always dependent on the background estimation. Hence, all techniques that improve the image reconstruction quality for re-projected content of the sprite are helpful. Another direction is the investigations of scenes where the foreground object occludes parts of the background over the whole sequence. This implicates a sprite with holes for the particular region. Here, background subtraction will not be useful in order to segment the foreground. A fusion with other spatio-temporal segmentation methods should be applied. Also advanced tracking algorithms, i.e., an identification of certain objects that have already been segmented in previous frames, can improve the sprite-based segmentation.

Sprite-based Video Coding

The following proposals are future improvements of the presented sprite-based coding strategies:

- Application of nonlinear higher order transformation models in order to improve the reconstruction quality with very little increase in bit rate for additional parameters.
- A spatial prediction technique in order to reduce the redundancies between multiple sprites.
- Utilization of spherical, cubical, or cylindrical spites, which automatically makes the generation of multiple sprites unnecessary. For this approach a good estimation of the focal length of the frames is necessary.
- A discrimination technique for scenes that are not applicable for sprite generation and, thus, sprite-based coding. In [67] a system was proposed that statistically, based on the root square mean error (RMSE) between adjacent global motion compensated frames, distinguishes between sequences that can be utilized for sprite generation and sequences that do not.
- A global rate-distortion optimization with rate control that determines the best sprite coding parameters for best coding performance.
- In order to strengthen the results obtained by sprite-based coding subjective test can be performed. Since objective measures yield bad results due to the impact of irrelevant information, e.g., reconstruction displacement and background movements, a subjective quality assessment will probably show better performance values.

Other Applications using Sprites

The applications given in this work are mainly investigated to be used for video coding. Other applications are:

- Utilization of super-resolution sprites for video enhancement and spatial up-sampling. In [71] we have shown that using super-resolution techniques we can spatially enhance several types of video. This can also be useful as post-processing method for decoded video.
- Sprites can also be used as key frames in video summarization methods. Video summarization attempts to describe the main characteristics of a sequence by automatically creating images that give a sense of its content. Those images are called key frames and together contain almost as much information as the original video. Usually, at least one key frame per shot is generated. Using sprite images as key frames can improve the describing character of the summarization.

- Sprites can be applied in powerful content editing tools. Because they allow a projection from one image into a number of frames, manipulation and editing of the background content of a scene shot is as easy as photo manipulation.

Appendix A

Mathematical Details

A.1 The epipolar line

From Equation 2.20 we obtain

$$x_2' = \frac{w_1 x_a + x_b}{w_1 z_a + z_b} \text{ and } x_2' = \frac{w_1 y_a + y_b}{w_1 z_a + z_b}. \tag{A.1}$$

For w_1 we get

$$w_1 = \frac{x_b - x_2' z_b}{x_2' z_a - x_a} \text{ and } w_1 = \frac{y_b - y_2' z_b}{y_2' z_a - y_a}. \tag{A.2}$$

Setting $w_1 = w_1$ from Equations A.2 it follows that

$$x_b z_a y_2' - x_b y_a + y_a z_b x_2' = y_b z_a x_2' - x_a y_b + x_a z_b y_2' \tag{A.3}$$

and finally

$$0 = (y_b z_a - y_a z_b) x_2' + (x_a z_b - x_b z_a) y_2' + (x_b y_a - x_a y_b) = \mathbf{l}_1' \mathbf{p}_2'. \tag{A.4}$$

Since the epipolar line $\mathbf{l}_1'$ only depends on the vectors $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$, which are constant for two camera views, we can derive an unique mapping between points in one view and the corresponding epipolar line in the second view. The mapping is given by a matrix $\mathbf{F}$, the *fundamental matrix*. It has eight degrees of freedom, three for the translation of the camera centers, three for the rotation, and two for the different focal length. The mapping can be described by

$$\mathbf{l}' = \mathbf{F}\mathbf{p}' \tag{A.5}$$

A.2 Homography Between Views on Plane Objects

From extension of Equation 2.8 to $\mathbb{P}^3$ we can derive the plane equation in projective space using Equation 2.28 as

$$\pi^T \mathbf{p} = 0 \text{ with } \pi = \left(\mathbf{n}^T, -d\right)^T. \tag{A.6}$$

Here, $\mathbf{n}$ is the normal vector of the plane and d is the distance from the world origin. For the first camera we can write

$$w_1 \mathbf{p}'_1 = \mathbf{K}_1 [\, \mathbf{I} \mid \mathbf{0} \,] \mathbf{p},$$

from what follows that

$$\mathbf{p} = \begin{pmatrix} \mathbf{K}_1^{-1} w_1 \mathbf{p}'_1 \\ \rho \end{pmatrix}. \tag{A.7}$$

Replacing $\mathbf{p}$ in Equation A.6 yields

$$\left(\mathbf{n}^T, -d\right) \begin{pmatrix} \mathbf{K}_1^{-1} w_1 \mathbf{p}'_1 \\ \rho \end{pmatrix} = 0 \tag{A.8}$$

and thus,

$$\mathbf{p} = \begin{pmatrix} \mathbf{K}_1^{-1} w_1 \mathbf{p}'_1 \\ \frac{n^T}{d} \mathbf{K}_1^{-1} w_1 \mathbf{p}'_1 \end{pmatrix}. \tag{A.9}$$

Replacing the result in the camera matrix of the second view, we obtain

$$\begin{aligned} w_2 \mathbf{p}'_2 &= \mathbf{K}_2 [\, \mathbf{R} \mid -\mathbf{R}\tilde{\mathbf{c}} \,] \mathbf{p} \\ &= \mathbf{K}_2 \left(\mathbf{R}\mathbf{K}_1^{-1} w_1 \mathbf{p}'_1 - \mathbf{R}\mathbf{c} \frac{n^T}{d} \mathbf{K}_1^{-1} w_1 \mathbf{p}'_1 \right) && \text{(A.10)} \\ &= \underbrace{\mathbf{K}_2 \mathbf{R} \left(\mathbf{I} - \tilde{\mathbf{c}} \frac{\mathbf{n}^T}{d} \right) \mathbf{K}_1^{-1}}_{\mathbf{H}} w_1 \mathbf{p}'_1. && \text{(A.11)} \end{aligned}$$

A.3 The Parabolic Transformation and Coefficient Derivation

In order to derive the coefficients $\mathbf{k}_{par} = (a_0, ..., b_5)^T$ for the parabolic transformation we have to determine first and second order derivatives of $\mathbf{x}'(mathbfx)$. The first order derivatives are

$$\begin{aligned} \frac{\partial x'}{\partial x} &= \frac{1}{\Gamma(\mathbf{x})} \left(a_1 - c_1 \frac{A(\mathbf{x})}{\Gamma(\mathbf{x})} \right), && \text{(A.12)} \\ \frac{\partial x'}{\partial y} &= \frac{1}{\Gamma(\mathbf{x})} \left(a_2 - c_2 \frac{A(\mathbf{x})}{\Gamma(\mathbf{x})} \right), && \text{(A.13)} \end{aligned}$$

and

$$\begin{aligned} \frac{\partial y'}{\partial x} &= \frac{1}{\Gamma(\mathbf{x})} \left(b_1 - c_1 \frac{B(\mathbf{x})}{\Gamma(\mathbf{x})} \right), && \text{(A.14)} \\ \frac{\partial y'}{\partial y} &= \frac{1}{\Gamma(\mathbf{x})} \left(b_2 - c_2 \frac{b(\mathbf{x})}{\Gamma(\mathbf{x})} \right). && \text{(A.15)} \end{aligned}$$

The second order derivatives result in

$$\frac{\partial^2 x'}{\partial x^2} = \frac{1}{\Gamma(\mathbf{x})^2}\left(2\frac{c_1^2 A(\mathbf{x})}{\Gamma(\mathbf{x})} - 2a_1c_1\right), \tag{A.16}$$

$$\frac{\partial^2 x'}{\partial x \partial y} = \frac{1}{\Gamma(\mathbf{x})^2}\left(2\frac{c_1c_2 A(\mathbf{x})}{\Gamma(\mathbf{x})} - a_1c_2 - a_2c_1\right), \tag{A.17}$$

$$\frac{\partial^2 x'}{\partial y^2} = \frac{1}{\Gamma(\mathbf{x})^2}\left(2\frac{c_2^2 A(\mathbf{x})}{\Gamma(\mathbf{x})} - 2a_2c_2\right), \tag{A.18}$$

and

$$\frac{\partial^2 y'}{\partial x^2} = \frac{1}{\Gamma(\mathbf{x})^2}\left(2\frac{c_1^2 B(\mathbf{x})}{\Gamma(\mathbf{x})} - 2b_1c_1\right), \tag{A.19}$$

$$\frac{\partial^2 y'}{\partial x \partial y} = \frac{1}{\Gamma(\mathbf{x})^2}\left(2\frac{c_1c_2 B(\mathbf{x})}{\Gamma(\mathbf{x})} - b_1c_2 - b_2c_1\right), \tag{A.20}$$

$$\frac{\partial^2 y'}{\partial y^2} = \frac{1}{\Gamma(\mathbf{x})^2}\left(2\frac{c_2^2 B(\mathbf{x})}{\Gamma(\mathbf{x})} - 2b_2b_2\right). \tag{A.21}$$

Using Equation 2.48 and 2.51 we can write

$$\begin{aligned} x'_{par} &= x'(\mathbf{x}_0) + \frac{\partial x'}{\partial x}(x-x_0) + \frac{\partial x'}{\partial y}(y-y_0) \\ &\quad + \frac{1}{2}\frac{\partial^2 x'}{\partial x^2}(x-x_0)^2 + \frac{\partial^2 x'}{\partial x \partial y}(x-x_0)(y-y_0) + \frac{1}{2}\frac{\partial^2 x'}{\partial y^2}(y-y_0)^2 \end{aligned} \tag{A.22}$$

and

$$\begin{aligned} y'_{par} &= y'(\mathbf{x}_0) + \frac{\partial y'}{\partial x}(x-x_0) + \frac{\partial y'}{\partial y}(y-y_0) \\ &\quad + \frac{1}{2}\frac{\partial^2 y'}{\partial x^2}(x-x_0)^2 + \frac{\partial^2 y'}{\partial x \partial y}(x-x_0)(y-y_0) + \frac{1}{2}\frac{\partial^2 y'}{\partial y^2}(y-y_0)^2, \end{aligned} \tag{A.23}$$

where the partial derivatives are evaluated at $\mathbf{x} = \mathbf{x}_0$. Equations A.22 and A.23 yield

$$\begin{aligned} x'_{par} &= x'(\mathbf{x}_0) - \frac{\partial x'}{\partial x}x_0 - \frac{\partial x'}{\partial y}y_0 + \frac{1}{2}\frac{\partial^2 x'}{\partial x^2}x_0^2 + \frac{\partial^2 x'}{\partial x \partial y}x_0y_0 + \frac{1}{2}\frac{\partial^2 x'}{\partial y^2}y_0^2 \\ &\quad - \left(\frac{\partial^2 x'}{\partial x^2}x_0 + \frac{\partial^2 x'}{\partial x \partial y}y_0 - \frac{\partial x'}{\partial x}\right)x - \left(\frac{\partial^2 x'}{\partial y^2}y_0 + \frac{\partial^2 x'}{\partial x \partial y}x_0 - \frac{\partial x'}{\partial y}\right)y \\ &\quad + \frac{1}{2}\frac{\partial^2 x'}{\partial x^2}x^2 + \frac{1}{2}\frac{\partial^2 x'}{\partial y^2}y^2 + \frac{\partial^2 x'}{\partial x \partial y}xy \end{aligned}$$

and

$$\begin{aligned} y'_{par} &= y'(\mathbf{x}_0) - \frac{\partial y'}{\partial x}x_0 - \frac{\partial y'}{\partial y}y_0 + \frac{1}{2}\frac{\partial^2 y'}{\partial x^2}x_0^2 + \frac{\partial^2 y'}{\partial x \partial y}x_0y_0 + \frac{1}{2}\frac{\partial^2 y'}{\partial y^2}y_0^2 \\ &\quad - \left(\frac{\partial^2 y'}{\partial x^2}x_0 + \frac{\partial^2 y'}{\partial x \partial y}y_0 - \frac{\partial y'}{\partial x}\right)x - \left(\frac{\partial^2 y'}{\partial y^2}y_0 + \frac{\partial^2 y'}{\partial x \partial y}x_0 - \frac{\partial y'}{\partial y}\right)y \\ &\quad + \frac{1}{2}\frac{\partial^2 y'}{\partial x^2}x^2 + \frac{1}{2}\frac{\partial^2 y'}{\partial y^2}y^2 + \frac{\partial^2 y'}{\partial x \partial y}xy. \end{aligned}$$

Replacing the derivatives with Equations A.12 to A.21 we finally obtain Equation 2.49 and Equation 2.52.

A.4 Linear Feature-based Estimation: SVD

Starting with Equation 3.4

$$\begin{aligned} 0 &= \mathbf{x}'_1 - \mathcal{T}_{opt}(\mathbf{x}_1) \\ \vdots &\quad \vdots \\ 0 &= \mathbf{x}'_{N_f} - \mathcal{T}_{opt}(\mathbf{x}_{N_f}). \end{aligned}$$

and rearranging the system of equations we obtain Equation 3.5:

$$\mathbf{b} = \mathbf{B}\mathbf{k},$$

The design matrices are specified in the following.

Translational Image Transformation

$$\mathbf{b}_{tr} = \begin{pmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ \vdots \\ x'_N - x_N \\ y'_N - y_N \end{pmatrix} \qquad \mathbf{B}_{tr} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{A.24}$$

Affine Image Transformation

$$\mathbf{b}_{aff} = \begin{pmatrix} x'_1 \\ y'_1 \\ \vdots \\ x'_N \\ y'_N \end{pmatrix} \qquad \mathbf{B}_{aff} = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_N & y_N \end{bmatrix} \tag{A.25}$$

Perspective Image Transformation

$$\mathbf{b}_{pers} = \begin{pmatrix} x'_1 \\ y'_1 \\ \vdots \\ x'_N \\ y'_N \end{pmatrix} \qquad \mathbf{B}_{pers} = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & 1 & x_1 & y_1 & -x_1y'_1 & -y_1y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & 0 & 0 & 0 & -x_Nx'_N & -y_Nx'_N \\ 0 & 0 & 0 & 1 & x_N & y_N & -x_Ny'_N & -y_Ny'_N \end{bmatrix} \tag{A.26}$$

Parabolic Image Transformation

$$\mathbf{b}_{par} = \begin{pmatrix} x'_1 \\ y'_1 \\ \vdots \\ x'_N \\ y'_N \end{pmatrix} \mathbf{B}_{par} = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & y_1^2 & x_1y_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_1 & y_1 & x_1^2 & y_1^2 & x_1y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & x_N^2 & y_N^2 & x_Ny_n & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_N & y_N & x_N^2 & y_N^2 & x_Ny_N \end{bmatrix} \tag{A.27}$$

A.5 Partial Derivatives for Newton-Raphson

From the equation for coordinate transformation depending on the chosen 2D image transformation $(x', y')^T = \mathcal{T}(\mathbf{k}; x, y)^T$ we can derive the partial derivatives

$$\frac{\partial x'}{\partial \mathbf{k}} = \begin{pmatrix} \frac{\partial x'}{\partial k_1} \\ \vdots \\ \frac{\partial x'}{\partial k_m} \end{pmatrix} \quad \text{and} \quad \frac{\partial y'}{\partial \mathbf{k}} = \begin{pmatrix} \frac{\partial y'}{\partial k_1} \\ \vdots \\ \frac{\partial y'}{\partial k_m} \end{pmatrix} \tag{A.28}$$

as follows.

Translational Image Transformation

$$\frac{\partial x'}{\partial \mathbf{k}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \frac{\partial y'}{\partial \mathbf{k}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{A.29}$$

with $\mathbf{k} = (a_0, b_0)^T$.

Affine Image Transformation

$$\frac{\partial x'}{\partial \mathbf{k}} = \begin{pmatrix} 1 \\ x \\ y \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \frac{\partial y'}{\partial \mathbf{k}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ x \\ y \end{pmatrix} \tag{A.30}$$

with $\mathbf{k} = (a_0, a_1, a_2, b_0, b_1, b_2)^T$.

Perspective Image Transformation

$$\frac{\partial x'}{\partial \mathbf{k}} = \begin{pmatrix} \frac{1}{\Gamma(x,y)} \\ \frac{x}{\Gamma(x,y)} \\ \frac{y}{\Gamma(x,y)} \\ 0 \\ 0 \\ 0 \\ -\frac{A(x,y)x}{\Gamma^2} \\ -\frac{A(x,y)y}{\Gamma^2} \end{pmatrix} \quad \text{and} \quad \frac{\partial y'}{\partial \mathbf{k}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\Gamma(x,y)} \\ \frac{x}{\Gamma(x,y)} \\ \frac{y}{\Gamma(x,y)} \\ -\frac{B(x,y)x}{\Gamma^2} \\ -\frac{B(x,y)y}{\Gamma^2} \end{pmatrix}, \tag{A.31}$$

with $\mathbf{k} = (a_0, a_1, a_2, b_0, b_1, b_2, c_1, c_2)^T$ and

$$\begin{aligned} A(x,y) &= a_0 + a_1 x + a_2 y \\ B(x,y) &= b_0 + b_1 x + b_2 y \\ \Gamma(x,y) &= 1 + c_1 x + c_2 y. \end{aligned}$$

Parabolic Image Transformation

$$\frac{\partial x'}{\partial \mathbf{k}} = \begin{pmatrix} 1 \\ x \\ y \\ x^2 \\ y^2 \\ xy \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \frac{\partial y'}{\partial \mathbf{k}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ x \\ y \\ x^2 \\ y^2 \\ xy \end{pmatrix}, \tag{A.32}$$

with $\mathbf{k} = (a_0, a_1, a_2, a_3, a_4, a_5, b_0, b_1, b_2, b_3, b_4, b_5)^T$.

Appendix B

Description of Test Sequences

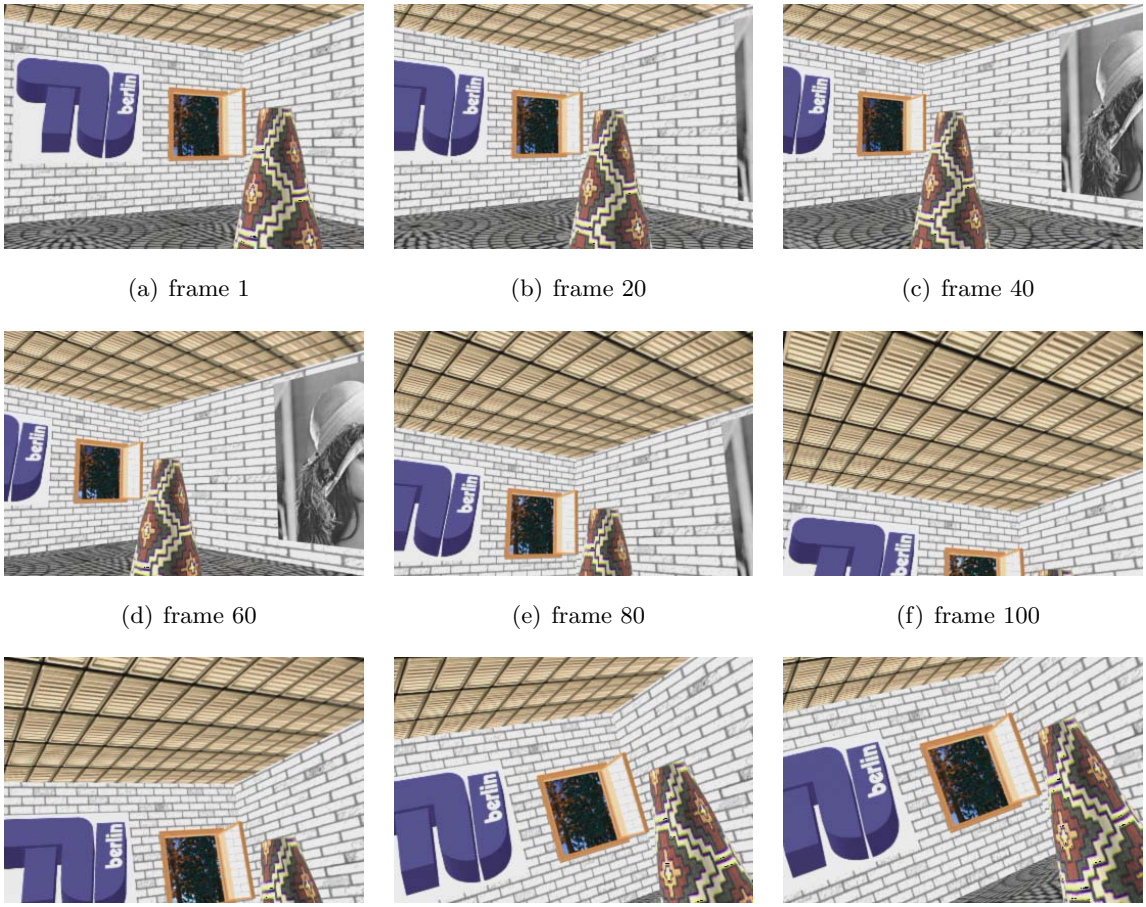
In this chapter we will give an overview of all utilized test sequences. Additionally, a short description of the objects, the object's motion as well as the camera motion is provided. Thus, the reader can get an impression of the type of scene shot that was used to assess a particular algorithm. Some of the sequences are standard MPEG test sequences often used by the scientific community, some are manually captured by the author, and some are recordings of regular digital television broadcasts. In order to make this survey complete also the well-known MPEG test sequences are explained.

Sequence "TUB-room"

"TUB-room" is an artificial test sequence rendered with the commercial 3D rendering software 3ds max. A virtual room was constructed with just one motionless object, a cone, in the center. The sequence consists of 150 frames in SIF-format (352x240 pixels, 30 fps). The camera motion can be subdivided into three segments, a horizontal pan from left to right, a pan back superimposed with a tilt towards the ceiling, and a tilt back towards a horizontal camera superimposed with a roll in clockwise direction. Additionally, the camera zooms out for the first segment and zooms in for the second and third segment. The textures have a high density of feature points and the lightening conditions are globally equal. The selected camera was fixed at one point and approximates a pinhole camera. Hence, no out-of-focus effects appear. Exemplary frames and a sprite image for scene overview are shown in figures Figures B.1 and B.2, respectively.

Sequence "Hall and monitor"

Sequence "Hall and monitor" (300 frames, CIF, 50 fps) belongs to the standard test sequences provided by MPEG. It contains a motionless camera view of a hall where people



(a) frame 1 (b) frame 20 (c) frame 40

(d) frame 60 (e) frame 80 (f) frame 100

(g) frame 120 (h) frame 140 (i) frame 150

Figure B.1: Exemplary frames of sequence "TUB-room" (150 frames, 352x240, 30 fps).

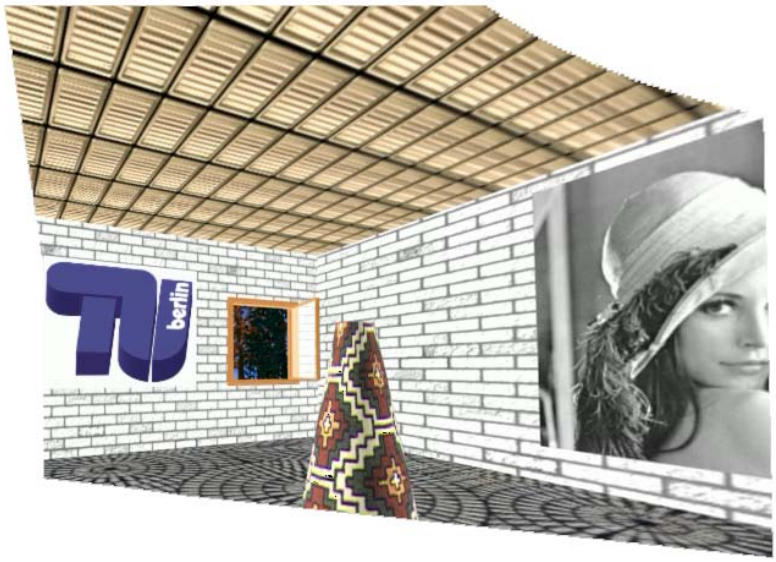


Figure B.2: Sprite of sequence "TUB-room".

(a) frame 1 (b) frame 40 (c) frame 80

(d) frame 120 (e) frame 160 (f) frame 200

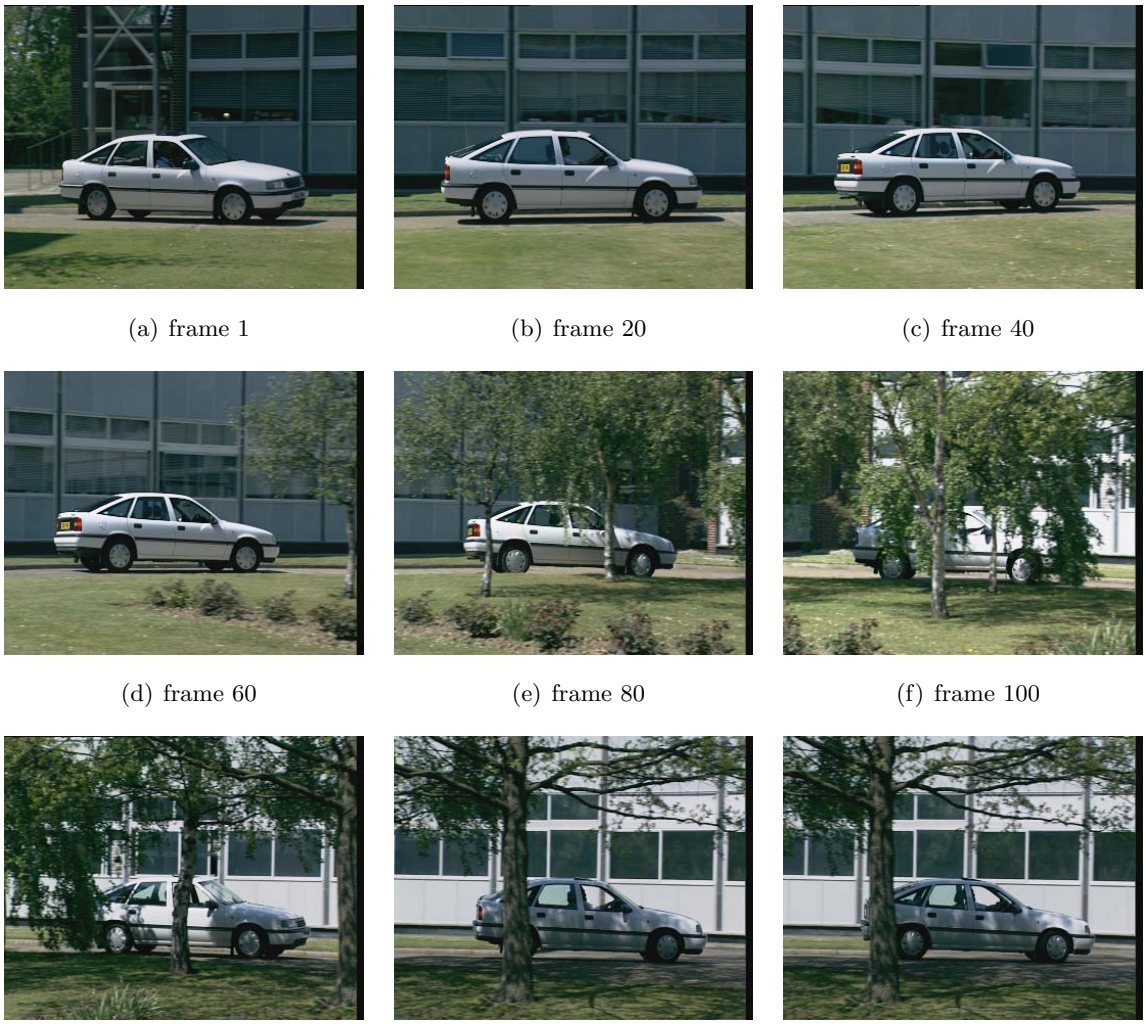
(g) frame 240 (h) frame 280 (i) frame 300

Figure B.3: Exemplary frames of sequence "Hall and monitor" (300 frames, 352x288, 50 fps).

deposit and take away different objects. The sequence is mainly used to assess background estimation and foreground segmentation methods, which is also the reason for usage in this thesis. Figure B.3 shows nine exemplary frames of this sequence.

Sequence "Vectra"

The "Vectra" sequence depicts a car followed by the camera. Thus the camera performs a pan from left to right. The shot consists of 142 frames in CIF format (352x288 pixels). The frame rate is 25 fps. Exemplary frames are shown in Figure B.4 while a sprite of the last 67 frames is shown in Figure B.5.



(a) frame 1 (b) frame 20 (c) frame 40

(d) frame 60 (e) frame 80 (f) frame 100

(g) frame 120 (h) frame 140 (i) frame 142

Figure B.4: Exemplary frames of sequence "Vectra" (142 frames, 352x288, 25 fps).

Figure B.5: Sprite of sequence "Vectra" using luminance values of frames 67 to 142.

Sequence "Mountain"

Sequence "Mountain" was recorded from the BBC documentation "Planet Earth" (16:9). For faster processing it was downsampled to 352x192 pixels. It consist of 100 frames and the frame rate is 25 fps. The translation-less camera and follows a leopard traversing a mountain. The main panning direction goes from upper left to lower right width a large focal length. Example frames and the resulting sprite are depicted in Figures B.6 and Figures B.7, respectively.

Sequence "Desert"

Sequence "Desert" was recorded from the BBC documentation "Planet Earth" (16:9). The frame size of the sequence is 720x400 pixels. It consist of 240 frames with a frame rate of 25 fps. The camera follows an antelope traversing a desert mountain. The camera movements are vertical tilt followed by a horizontal pan. Example frames and the resulting sprite are depicted in Figures B.8 and Figures B.9, respectively.

Sequence "Charlottenburg"

Sequence "Charlottenburg" was captured with a camcorder. Hence, it has the typical characteristics of home videos, e.g., unsteady movements and changes in the lighting conditions. Since it was not mounted on a tripod, also small translational camera movements occur. The frame size was downsampled to 360x288 pixels. The shot consists of 450 frames with a frame rate of 25 fps. The camera follows a person walking through the forecourt of Charlottenburg castle in Berlin. Over the final 100 frames the camera approaches a full view of the entrance building. During the pan the camera zoom (focal length) changes very dynamically form wide angle to close-up view and back. Example frames and a sprite are shown in Figures B.10 and B.11, respectively.

Sequence "Stefan"

Sequence "Stefan" is one of the well-known MPEG test sequences mainly used for assessing sprite construction and sprite coding approaches. It consists of 300 frames at a frame rate of 30 fps showing tennis player Stefan Edberg in action. The camera follows the player. The audience in the background mainly focuses on the position of the ball. Since the the camera movement is very quick, the movements of the audience cannot be realized by an observer. Therefore, these movements can be regarded as irrelevant. In our experiments we used the SIF version (standard intermediate format) of this sequence with a frame size of 352x240 pixels. The original video is four times the size but interlaced. Example frames and a sprite are shown in Figures B.12 and B.13, respectively.

(a) frame 1 (b) frame 10 (c) frame 20

(d) frame 30 (e) frame 40 (f) frame 50

(g) frame 60 (h) frame 70 (i) frame 90

Figure B.6: Exemplary frames of sequence "Mountain" (100 frames, 352x192, 25 fps).

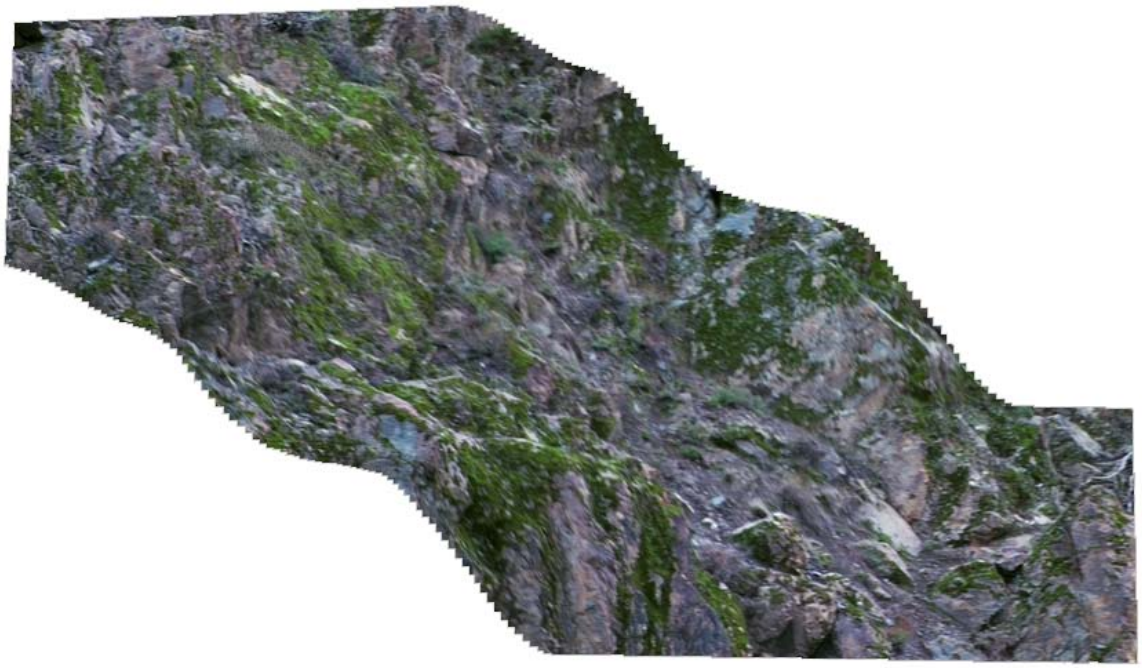


Figure B.7: Sprite of sequence "Mountain".

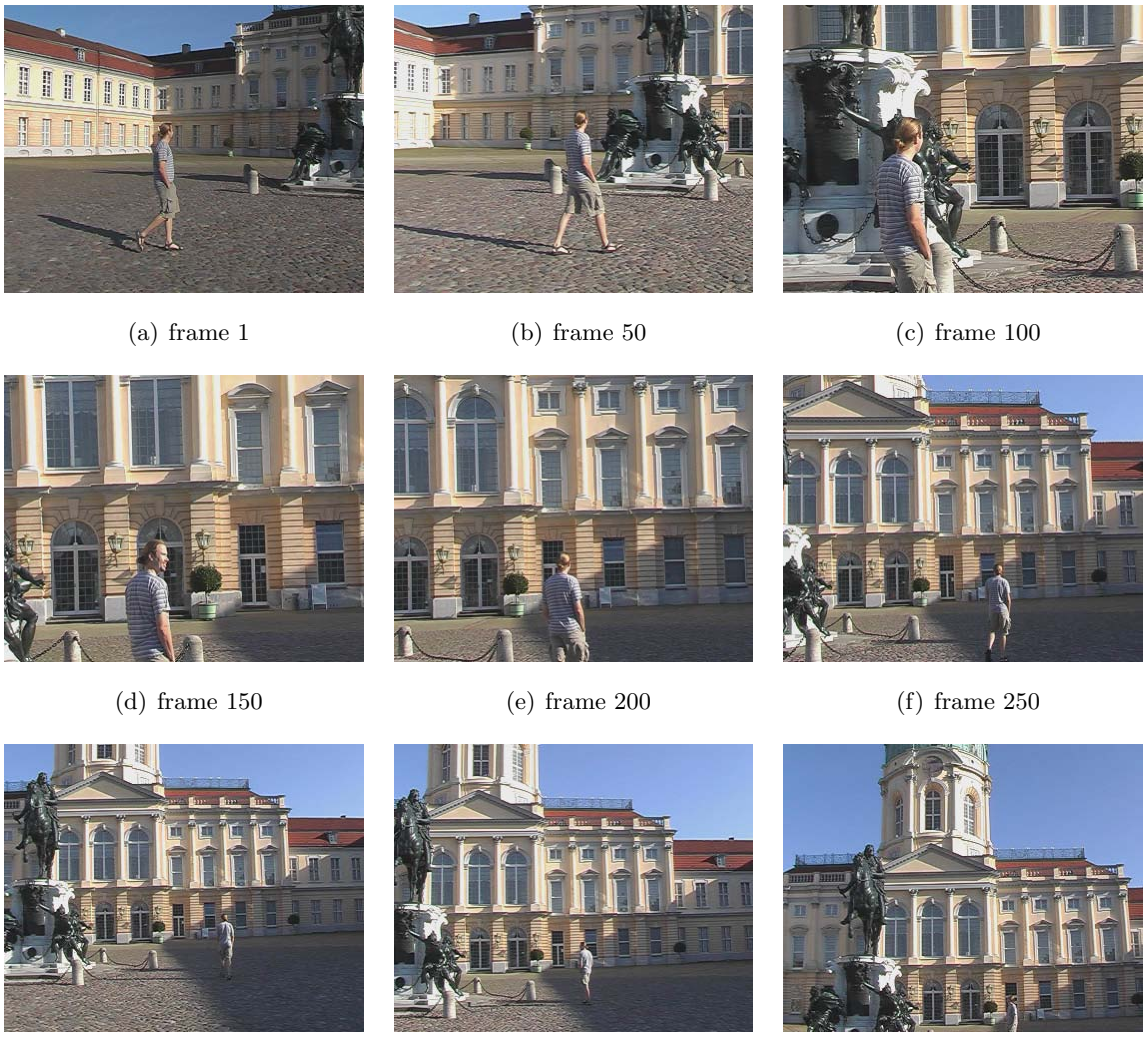
(a) frame 1 (b) frame 30 (c) frame 60

(d) frame 90 (e) frame 120 (f) frame 150

(g) frame 180 (h) frame 210 (i) frame 240

Figure B.8: Exemplary frames of sequence "Desert" (240 frames, 720x400, 25 fps).

Figure B.9: Sprite of sequence "Desert".



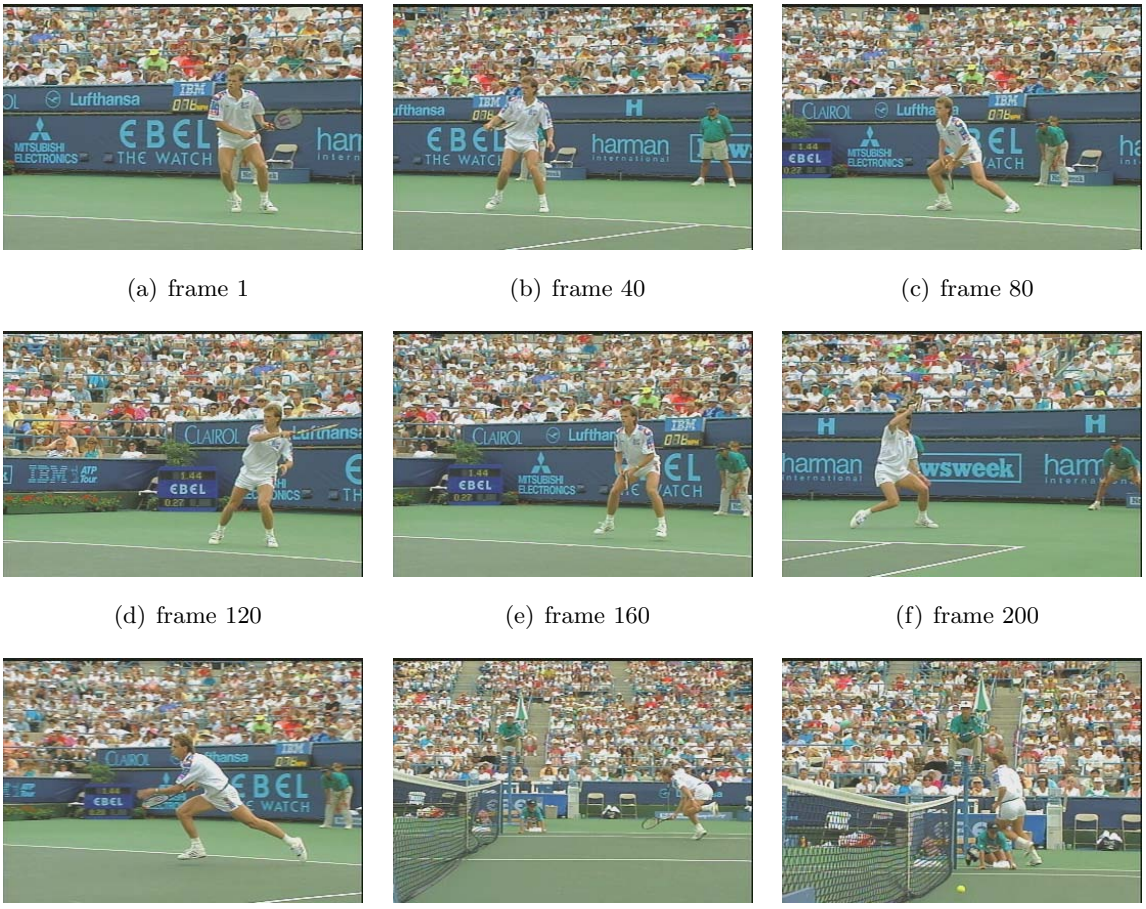
(a) frame 1 (b) frame 50 (c) frame 100

(d) frame 150 (e) frame 200 (f) frame 250

(g) frame 300 (h) frame 350 (i) frame 400

Figure B.10: Exemplary frames of sequence "Charlottenburg" (450 frames, 360x288, 25 fps).

Figure B.11: Sprite of sequence "Charlottenburg".



(a) frame 1 (b) frame 40 (c) frame 80

(d) frame 120 (e) frame 160 (f) frame 200

(g) frame 240 (h) frame 280 (i) frame 300

Figure B.12: Exemplary frames of sequence "Stefan" (300 frames, 352x240 (SIF), 30 fps).

Figure B.13: Sprite of sequence "Stefan".

Sequence "Horse"

Sequence "Horse" (500 frames) actually consists of two shots. For our processing only the second shot, starting at frame 133, was used. Hence, the shot consists of 368 frames. The frame size is 360x288 pixels at a frame rate of 50 fps. The scene shows a participant of a cross country equestrian jumping tournament while crossing a ditch. The fixed camera follows the horse in a horizontal left-to-right pan. Exemplary frames and a sprite image are shown in Figures B.14 and B.15.

Sequence "Monitor"

Sequence "Monitor" is another shot captured with a camcorder. It shows only rigid background objects while the camera slowly pans over the scene. Thus, the redundancies between adjacent frames are very high. Therefore, this scene is used to figure out what coding gains over conventional coding strategies can be reached using sprite-based coding. The shot consists of 150 frames with a frame size of 360x288 pixels. The frame rate is 25 fps. Exemplary frames are displayed in Figure B.16 while a sprite is shown in Figure B.17.

Sequence "Biathlon"

This sequence is a recording of digital video broadcast (DVB). It shows a biathlete skiing in front of some banner ads. The camera is fixed and pans from right to left in order to follow the biathlete. Since the distance between sportsman and camera increases, a constant zoom-in occurs during the scene. The shot was downsampled to CIF format (common intermediate format) with a frame size of 352x288 pixels. It consists of 200 frames at a frame rate of 25 fps. Images of the sequence an the sprite are shown in Figures B.18 and B.19, respectively.

Sequence "Race01"

Sequence "Race01" is one view of the MPEG-3DAV multiple view test sequences "Race". The sequence was captured with a row of VGA cameras. Hence, the image quality is rather low, e.g., color shifts and darkening effects towards the image borders. It shows a part of a kart racing tournament. While following the karts, the camera array pans from left to right. We selected 100 frames of the leftmost camera as test sequence. For the sake of robustness we cropped the frames to size 544x336. The frame rate is 30 fps. Example frames and the resulting sprite are depicted in Figures B.20 and B.21. Multiple views of this sequence were used to propose a sprite-based multiple view coding (MVC) approach [65].

(a) frame 1

(b) frame 40

(c) frame 80

(d) frame 120

(e) frame 160

(f) frame 200

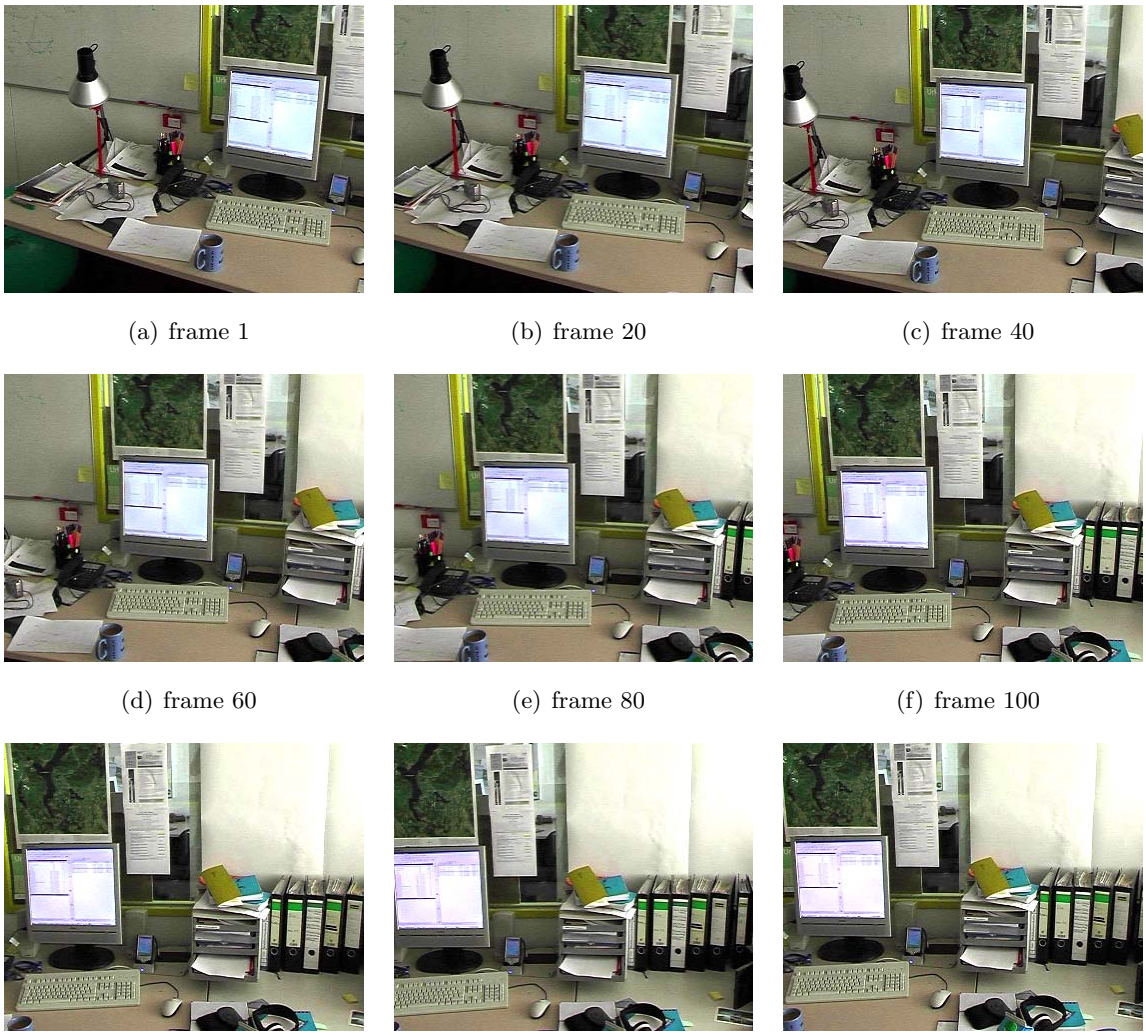
(g) frame 240

(h) frame 280

(i) frame 350

Figure B.14: Exemplary frames of sequence "Horse" (300 frames, 360x288, 25 fps).

Figure B.15: Sprite of sequence "Horse".



(a) frame 1 (b) frame 20 (c) frame 40

(d) frame 60 (e) frame 80 (f) frame 100

(g) frame 120 (h) frame 140 (i) frame 150

Figure B.16: Exemplary frames of sequence "Monitor" (150 frames, 360x288, 25 fps).

Figure B.17: Sprite of sequence "Monitor".

(a) frame 1 (b) frame 25 (c) frame 50

(d) frame 75 (e) frame 100 (f) frame 125

(g) frame 150 (h) frame 175 (i) frame 200

Figure B.18: Exemplary frames of sequence "Biathlon" (200 frames, 352x288 (CIF), 25 fps).

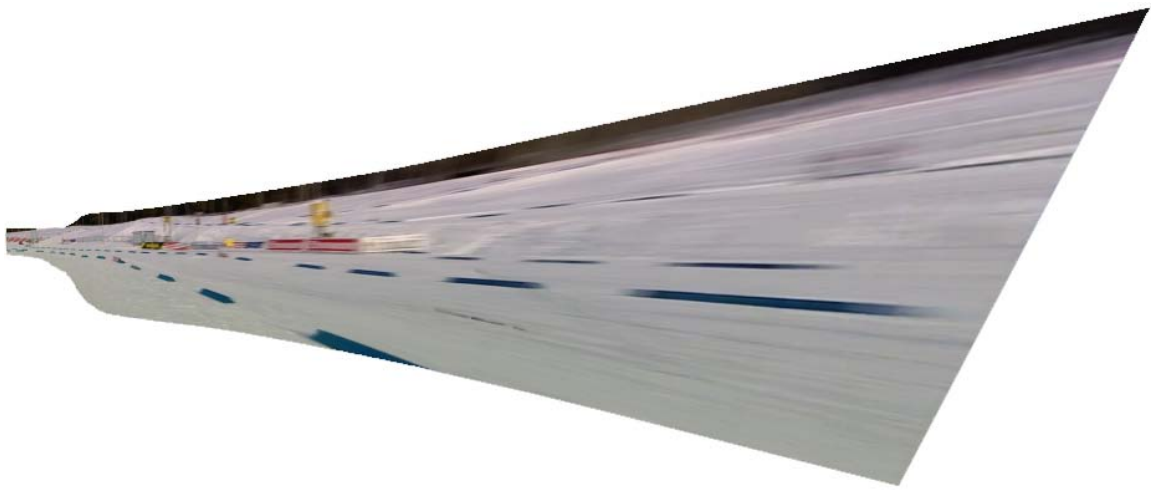


Figure B.19: Sprite of sequence "Biathlon".

(a) frame 1 (b) frame 10 (c) frame 20

(d) frame 30 (e) frame 40 (f) frame 50

(g) frame 60 (h) frame 70 (i) frame 90

Figure B.20: Exemplary frames of sequence "Race01" (100 frames, 544x336, 30 fps).

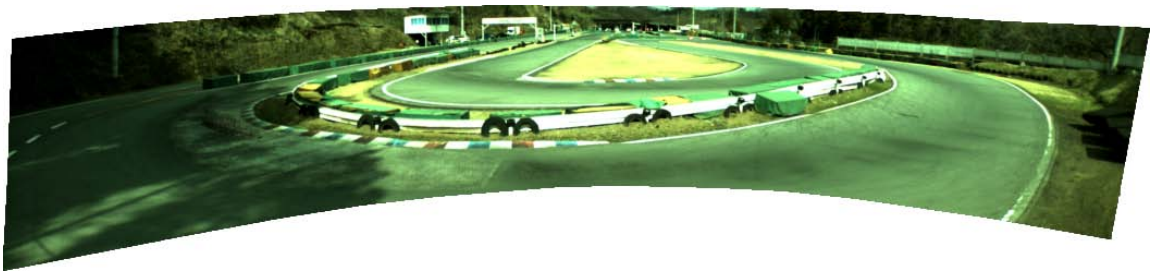


Figure B.21: Sprite of sequence "Race01" using frame 35 as reference frame.

(a) frame 1 (b) frame 8 (c) frame 16

(d) frame 24 (e) frame 32 (f) frame 40

(g) frame 48 (h) frame 56 (i) frame 60

Figure B.22: Exemplary frames of sequence "Group" (60 frames, 352x288, 25 fps).

Sequence "Group"

The sequence "Group" is another test sequence without any camera movement. Thus, it is mainly used for robust background estimation and segmentation evaluation. Since a foreground mask is provided with the sequence, pixel-based segmentation quality measures can be used. The sequence shows a group of three persons entering a room. The scene consists of 60 frames and the frame size is 352x288 pixels. Exemplary frames can be found in B.22.

Bibliography

[1] T. Aach and A. Kaup. Bayesian algorithms for change detection in image sequences using markov random fields. *Signal Processing: Image Communication*, 7(2):147 – 160, 1995.

[2] P. J. Allwardt. *Entwicklung und Validierung eines hierarchischen prädiktiven Schätzerverfahrens zur Generierung von Mosaiken als Superresolution.* Studienarbeit, TU Berlin, 2004.

[3] S. Baker and T. Kanade. Super-resolution optical flow. *Technical Report CMU-RI-TR-99-36, Carnegie Mellon University*, 1999.

[4] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002.

[5] R. Bernstein. Digital image processing of earth observation sensor data. *IBM Journal of Research and Development*, 1(20):40 – 57, Jan. 1976.

[6] S. Borman and R. L. Stevenson. Super-resolution from image sequences - a review. In *IEEE Midwest Symposium on Systems and Circuits,MWSCAS'98*, pages 374 – 378, 1998.

[7] A. Capel and A. Zisserman. Computer vision applied to super resolution. *IEEE Signal Processing Magazine*, 20(3):75 – 86, May 2003.

[8] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'98*, pages 885 – 891, June 1998.

[9] P. D. Capel. *Image Mosaicing and Super-resolution.* Springer, 2004.

[10] Y. Caspi and M. Irani. Alignment of non-overlapping sequences. In *IEEE Int. Conf. on Computer Vision, ICCV'01*, volume 2, pages 76 – 83, 2001.

[11] A. Cavallaro and T. Ebrahimi. Accurate video object segmentation through change detection. In *IEEE Int. Conf. on Multimedia and Expo, ICME'02*, volume 1, pages 445 – 448, 2002.

[12] M. Chan, D. Metaxas, and S. Dickinson. Physics-based tracking of 3d objects in 2d image sequences. In *Int. Conf. on Pattern Recognition, ICPR'94*, volume 1, pages 432 – 436, 1994.

[13] S.-Y. Chien, S.-Y. Ma, and Chen L.-G. Efficient moving object segmentation algorithm using background registration technique. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(7):577 – 586, July 2002.

[14] S.-Y. Chien, S.-Y. Ma, Chao W.-M., 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 technique. In *IEEE Int. Conf. on Image Processing, ICIP'02*, volume 1, Rochester, New York, Sept. 2002.

[15] Y.H. Choi, Y. K. Seong, and T.-S. Choi. Image mosaicing with automatic scene segmentation for video indexing. In *IEEE Int. Conf. on Consumer Electronics, ICCE'02*, June 2002.

[16] R. Crinon and I. Sezan. Sprite-based video coding using on-line segmentation. In *Int. Conf. an Acoustics, Speech, and Signal Processing, ICASSP'98*, volume 5, pages 2981 – 2984, May 1998.

[17] A. Dasu, S. Raghavan, N. C. Raghavendra, and S. Panchanathan. Arithmetic precision for perspective transform in sprite decoding of MPEG-4. In *Proc. SPIE Vol. 3970, Media Processors 2000*, volume 3970, pages 138 – 145, Jan. 2000.

[18] F. Dufaux and J. Konrad. Efficient, robust and fast global motion estimation for video coding. *IEEE Transactions on Image Processing*, 9(3):497 – 501, March 2000.

[19] T. Echigo, R. Radke, P. Ramadge, H. Miyamori, and S.-I. Iisaku. Ghost error elimination and superimposition of moving objects in video mosaicing. In *Int. Conf. on Image Processing, ICIP'99*, volume 4, pages 128 – 132, Oct. 1999.

[20] P. Eisert, T. Wiegand, and B. Girod. Model-aided coding: A new approach to incorporate facial animation into motion-compensated video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):444 – 358, April 2000.

[21] B. Esteves Pires and P. M. Q. Aguiar. Featureless global alignement of multiple images. In *IEEE Int. Conf. on Image Processing, ICIP'05*, volume 1, Genova, Italy, Sept. 2005.

[22] A. N. Evans, Y. Guo, and D. M. Monro. Limited motion estimation scheme for multimedia video compression. In *6th Int. Conf. on Electronics, Circuits and Systems, ICECS'99*, volume 1, pages 453 – 456, Sept. 1999.

[23] D. Farin. *Automatic Video Segmentation Employing Object/Camera Modeling Techniques*. Ph.D. thesis Technische Universiteit Eindhoven, 2005.

[24] D. Farin and P. H. N. de With. Estimating physical camera parameters for 3DAV video coding. In *25th Symposium on Information Theory in the Benelux*, June 2004.

[25] D. Farin and P. H. N. de With. Estimating physical camera parameters based on multi-sprite motion estimation. *SPIE Image and Video Communications and Processing 2005*, 1(1):489 – 500, March 2005.

[26] 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, March 2006.

[27] D. Farin, P. H. N. de With, and W. Effelsberg. Optimal partitioning of video sequences for MPEG-4 sprite encoding. In *24th Symposium on Information Theory in the Benelux*, pages 79 – 86, Veldhoven, Netherlands, May 2003.

[28] D. Farin, P. H. N. de With, and W. Effelsberg. Robust background estimation for complex video sequences. In *IEEE Int. Conf. on Image Processing ICIP'03*, pages 145 – 148, Barcelona, Spain, May 2003.

[29] D. Farin, P. H. N. de With, and W. Effelsberg. Minimizing MPEG-4 sprite coding-cost using multi-sprites. In *SPIE Visual Communications and Image Processing, VCIP'04*, San Jose, California, Jan. 2004.

[30] D. Farin, P. H. N. de With, and W. Effelsberg. Video-object segmentation using multi-sprite background subtraction. In *IEEE Int. Conf. on Multimedia and Expo ICME'04*, Taipei, Taiwan, June 2004.

[31] D. Faugeras and Q.-T. Luong. *The Geometry of Multiple Images*. MIT Press, Cambridge, Massachusetts, 2001.

[32] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 6(24):381 – 395, 1981.

[33] M. Flierl, T. Wiegand, and B. Girod. Rate-constrained multihypothesis prediction for motion-compensated video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(11):957 – 969, Nov. 2002.

[34] A. Fusiello, M. Aprile, R. Marzotto, and V. Murino. Mosaic of a video with multiple moving objects. In *IEEE Int. Conf. on Image Processing ICIP'03*, volume 2, pages 307 – 310, Barcelona, Spain, May 2003.

[35] J. F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz. *The Expanding Digital Universe - A Forecast of Worldwide Information Growth Through 2010.* IDC White Paper, 2007.

[36] E. Gelasca Drelie, M. Karaman, T. Ebrahimi, and T. Sikora. A framework for evaluating video object segmentation algorithms. In *IEEE CVPR 2006 Workshop (Perceptual Organization in Computer Vision POCV)*, New York, USA, June 2006.

[37] Y. Gong, D LaRose, and G. Proietti. A robust image mosaicing technique capable of creating integrated panoramas. In *IEEE Int. Conf. on Information Visualization, ICIV'99*, pages 24 – 29, July 1999.

[38] N. Grammalidis, D. Beletsiotis, and M. Strintzis. Sprite generation and coding in multiview image sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(2):302 – 311, March 2000.

[39] N. Grammalidis, D. Beletsiotis, and M.G. Strintzis. Multiview sprite generation and coding. In *Int. Conf. on Image Processing, ICIP'99*, volume 2, pages 477 – 481, Oct. 1999.

[40] N. Grammalidis, D. Beletsiotis, and M.G. Strintzis. Sprite generation and coding of multiview image sequences. In *Int. Conf. on Image Analysis and Processing, ICIAP'99*, pages 95 – 100, Sept. 1999.

[41] C. J. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conf.*, pages 147 – 151, Manchester, 1988.

[42] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2nd edition, 2003.

[43] L. Hill and T. Vlachos. On the estimation of global motion using phase correlation for broadcast applications. In *7th Int. Conf. on Image Processing and its Applications, IPA'99*, volume 2, pages 721 – 725, July 1999.

[44] D. Hong and W. Woo. A background subtraction for a vision-based user interface. In *Joint Conf. of the 4th Int. Conf. on Information, Communications and Signal Processing and the 4th Pacific Rim Conference, ICICS-PCM'03*, volume 1, pages 263 – 267, Dec. 2003.

[45] C.-T. Hsu and Y.-C. Tsan. Mosaics of video sequences with moving objects. In *IEEE Int. Conf. on Image Processing, ICIP'01*, volume 2, pages 387 – 390, Oct. 2001.

[46] C.-T. Hsu and Y.-C. Tsan. Mosaics of video sequences with moving objects. *Signal Processing: Image Communications*, 19(1):81 – 98, Jan. 2004.

[47] C.-T. Hsu and J.-L. Wu. Multiresolution mosaic. *IEEE Transactions on Consumer Electronics*, 42(4):981 – 990, Nov. 1996.

[48] P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, (35):73 – 101, 1964.

[49] M. Irani and P. Anandan. Video indexing based on mosaic representations. In *Proceedings of the IEEE, PIEEE*, volume 86 of *5*, pages 905 – 921, May 1998.

[50] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu. Efficient representations of video sequences and their applications. *Signal Processing: Image Communication, special issue on Image and Video Semantics: Processing, Analysis, and Application*, 8(4):327 – 351, May 1996.

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

[52] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP: Graphical Models and Image Processing*, 53(3):231 – 239, May 1991.

[53] M. Isard and A. Blake. CONDENSATION conditional density propagation for visual tracking. *Int. Journal of Computer Vision*, 28(1):5 – 28, 1998.

[54] K. Jinzenji, S. Okada, H. Watanabe, and N. Kobayashi. Automatic two-layer video object plane generation scheme and its application to MPEG-4 video coding. In *IEEE Int. Symposium on Circuits and Systemes, ISCAS'00*, volume 3, pages 606 – 609, Geneva, Switzerland, May 2000.

[55] K. Jinzenji, H. Watanabe, S. Okada, and N. Kobayashi. MPEG-4 very low bit-rate video compression using sprite coding. In *IEEE Int. Conf. on Multimedia and Expo, ICME'01*, pages 4 – 7, Tokyo, Japan, Aug. 2001.

[56] H.-S. Jung, R.-C. Kim, and S.-U. Lee. Error-resilient video coding using long-term memory prediction and feedback channel. *Signal Processing: Image Communication*, 17(8):597 – 609, Sept. 2002.

[57] M. Karaman, L. Goldmann, D. Yu, and T Sikora. Comparison of static background segmentation methods. In *SPIE Visual Communications and Image Processing, VCIP'05*, Geneva, Switzerland, July 2005.

[58] Y. Keller and A. Averbruch. Fast global motion estimation for MPEG-4 video compression. In *PACKET VIDEO*, April 2003.

[59] Y. Keller and A. Averbruch. Fast gradient methods based on global motion estimation for video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(4):300 – 309, April 2003.

[60] Y. Keller and A. Averbruch. A projection-based extension to phase correlation image alignment. *EURASIP Signal Processing*, 87(1):124 – 133, Jan. 2007.

[61] J. Kim and T Sikora. Hybrid recursive energy-based method for robust optical flow on large motion fields. In *IEEE Int. Conf. on Image Processing, ICIP'05*, Geneva, Switzerland, Sept. 2005.

[62] M. Kourogi, T. Kurata, J. Hoshino, and Y Muraoka. Real-time image mosaicing from a video sequence. In *IEEE Int. Conf. on Image Processing, ICIP'99*, volume 4, pages 133 – 137, Oct. 1999.

[63] P. Krey. *Development and Implementation of a Block-based Mode for Video Coding integrated with the H.264/AVC Coding Standard using Background Sprites.* Diplomarbeit, TU Berlin, 2007.

[64] P. Krämer, O. Hadar, J. Benois-Pineau, and J.P. Domenger. Super-resoluiton mosaicing from MPEG compressed video. In *IEEE Int. Conf. on Image Processing, ICIP'05*, volume 1, pages 893 – 896, Genova, Italy, Oct. 2005.

[65] A. Krutz, M. Dröse, M. Kunter, M. Mandal, M. Frater, and T. Sikora. Low bit-rate object-based multi-view video coding using mvc. In *3DTV-Conference*, Kos Island, Greece, Sept. 2007.

[66] A. Krutz, M. Frater, and T. Sikora. Window-based image registration using variable window sizes. In *Int. Conf. on Image Processing, ICIP'07*, San Antonio, TX, USA, Sept. 2007.

[67] A. Krutz, M. Kunter, M. Dröse, M. Mandal, M. Frater, and T. Sikora. Content-adaptive video coding combining object-based coding and H.264/AVC. In *26th Picture Coding Symposium, PCS'07*, Lisboa, Portugal, Nov. 2007.

[68] A. Krutz, M. Kunter, M. Mandal, M. Frater, and T. Sikora. Motion-based object segmentation using sprites and anisotropic diffusion. In *8th Int. Workshop on Image Analysis for Multimedia Interactive Services WIAMIS'07*, Santorini, Greece, June 2007.

[69] 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, Dec. 2005.

[70] M. Kunter, A. Krutz, M. Dröse, M. Frater, and T. Sikora. Object-based multiple sprite coding of unsegmented videos using h.264/avc. In *IEEE Int. Conf. on Image Processing, ICIP'07*, San Antonio, Texas, USA, Sep. 2007.

[71] M. Kunter and T. Sikora. Super-resolution mosaicing for object based video format conversion. In *7th Int. Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'06*, Seoul, Korea, April 2006.

[72] I.-S. Kuo and L.-H. Chen. High visual quality sprite generator using automatic segmentation and intelligent blending. In *SPIE Visual Communications and Image Processing, VCIP'05*, pages 2128 – 2139, Bejing, China, July 2005.

[73] M.-C. Lee, W. Chen, C. B. Lin, C. Gu, T. Markoc, S. I. Zabinsky, and R. Szeliski. A layered video object coding system using sprite and affine motion modell. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):130 – 145, Feb. 1991.

[74] R. W. Lienhart. Comparison of automatic shot boundary detection algorithms. In *Proc. SPIE Vol. 3656, Storage and Retrieval for Image and Video Databases VII*, pages 290 – 301, San Jose, CA, USA, Oct. 1999.

[75] Z. Lin and H-Y. Shum. Fundamental limits of reconstruction-based superresolution algorithms under local translation. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 26(1):83 – 97, 2004.

[76] 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*, pages 118 – 125, Beijing, China, Oct. 2001.

[77] Y. Lu, W. Gao, and F. Wu. Sprite generation for frame-based video coding. In *IEEE Int. Conf. on Image Processing, ICIP'01*, volume 1, pages 473 – 476, Thessaloniki, Greece, Oct. 2001.

[78] Y. Lu, W. Gao, and F. Wu. High efficient sprite coding with directional spatial prediction. In *IEEE Int. Conf. on Image Processing, ICIP'02*, volume 1, pages 201 – 204, Rochester, New York, Sept. 2002.

[79] Y. Lu, W. Gao, and F. Wu. Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(5):394 – 405, May 2003.

[80] Y. Lu, W. Gao, F. Wu, H. Lu, and X Chen. A robust offline sprite generation approach. *ISO/IEC JTC1/SC29/WG11 MPEG01/m6778, Video Tool Proposal*, Jan. 2001.

[81] C.L. Luengo Hendriks and L.J. van Vliet. Improving resolution to reduce aliasing in an undersampled image sequence. In *SPIE Proc. Electronic Imaging*, volume 3965A-23, page S4, San Jose, CA, Jan. 2000.

[82] D. Marpe, H. Schwarz, 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.

[83] R. Marzotto, A. Fusiello, and V. Murino. High resolution video mosaicing with global alignment. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'04*, pages 692 – 698, Washington, DC, June 2004.

[84] C. McGlone, E. Mikhail, and J. Bethel, editors. *Manual of Photogrammetry*. ASPRS, 5 edition, 2004.

[85] T. Mei, X.-S. Hua, H.-Q. Zhou, S. Li, and Zhang H.-J. Efficient video mosaicing based on motion. In *IEEE Int. Conf. on Image Processing, ICIP'05*, volume 1, pages 861 – 864, Genova, Italy, Sept. 2005.

[86] T. Meier and K. N. Ngan. Automatic segmentation of moving objects for video object plane generation. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):525 – 538, Sep. 1998.

[87] Z. Mihajlovic, A. Goluban, and M. Zagar. Efficient video mosaicing based on motion. In *IEEE Int. Symposium on Industrial Electronics, ISIE'99*, volume 1, pages 193 – 198, July 1999.

[88] E. Morillon, R. Balter, L. Morin, and S. Pateux. 2D/3D hybrid modeling for video sequence. In *Int. Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS'04*, volume 1, Lisbon, Portugal, April 2004.

[89] A. Netzer and C. Gotsman. Mosaicing video sequences. *Technical Report, Technion - Israel Institute of Technology*, 2001.

[90] M. K. Ng and N. K. Bose. Mathematical analysis of super-resolution methodology. *IEEE Signal Processing Magazin*, 20(3):62 – 74, May 2003.

[91] H. Nicolas. Optimal criterion for dynamic mosaicking. In *IEEE Int. Conf. on Image Processing, ICIP'99*, volume 4, pages 138 – 142, Kobe, Japan, Oct. 1999.

[92] H. Nicolas. Robust motion segmentation for content-based video coding. In *6th Conf. on Content-based Multimedia Information Access*, pages 594 – 601, Paris, June 2000.

[93] H. Nicolas. New methods for dynamic mosaicking. *IEEE Transactions on Image Processing*, 10(8):1239 – 1251, Aug. 2001.

[94] J. M. Odobez and P. Bouthemy. Robust multiresolution estimation of parametric motion models. *Journal of Visual Communication and Image Representation*, 6(4):348 – 365, Dec. 1995.

[95] X. Orriols, L. Barcelo, and X. Binefa. Polynomial fiber description of motion for video mosaicing. In *IEEE Int. Conf. on Image Processing, ICIP'01*, pages 386 – 389, Thessaloniki, Greece, June 2001.

[96] J. Ostermann. Object-based analysis-synthesis coding (OBASC) based on the source model of moving flexible 3-d objects. *IEEE Transactions on Image Processing*, 3(5):705 – 711, Sept. 1994.

[97] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T Stockhammer, and T. Wedi. Video coding with H.264/AVC: tools, performance, and complexity. *IEEE Circuits and Systems Magazine*, 4(1):8 – 28, 2003.

[98] J. Ostermann and M. Kampmann. Source models for content-based video coding. In *IEEE Int. Conf. on Image Processing, ICIP'00*, pages 62 – 659, Vancouver, BC, Canada, Sept. 2000.

[99] S. C. Park, M. K. Park, and M. G. Kang. Super-resolution image reconstruction: A technical overview. *IEEE Signal Processing Magazin*, 20(3):21 – 36, May 2003.

[100] S. Peleg, B. Rousso, A. Rav-Akha, and A. Zomet. Mosaicing on adaptive manifolds. *IEEE Transactions on Pattern Recognition and Machine Analysis*, 22(10):1144 – 1154, Oct. 2000.

[101] F. Pereira and T Ebrahimi, editors. *The MPEG-4 Book*. Prentice Hall, Englewood Cliffs, 2002.

[102] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629 – 639, 1990.

[103] T. Q. Pham, L. J. van Vliet, and K. Schutte. Influence of signal-to-noise ratio and point spread function on limits of super-resolution. In *SPIE Int. Conf. on Visual Communications and Image Processing, VCIP'05*, pages 169 – 180, San José, CA, USA, Jan. 2005.

[104] M. Pilu. On using raw MPEG motion vectors to determine global camera motion. *HPL-97-102, HP Company*, pages 21 – 36, August 1997.

[105] L. Pinheiro da Silva, M. Auvergne, D. Toublanc, J. Rowe, R. Kuschnig, and J. Matthews. Estimation of a super-resolved psf from undersampled stellar observations. *Astronomy and Astrophysics*, pages 363 – 369, June 2006.

[106] R. Pless, T. Brodsky, and Y. Aloimonos. Independent motion: The importance of history. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'99*, volume 2, pages 2092 – 2097, Fort Collins, Colorado, June 1999.

[107] M. Pollefeys, L. J. Van Gool, A. Zisserman, and A. W. Fitzgibbon, editors. *3D Structure from Images - SMILE 2000, Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments Dublin, Ireland, July 12, 2000, Revised Papers*, volume 2018 of *Lecture Notes in Computer Science*. Springer, 2001.

[108] D. Rajan, S. Chaudhuri, and V. J. Manjunath. Multi-objective super resolution: Concepts and examples. *IEEE Signal Processing Magazin*, 20(3):49 – 61, May 2003.

[109] I. E. G. Richardson. *Video Codec Design*. John Wiley & Sons, 2002.

[110] I. E. G. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. John Wiley & Sons, 2003.

[111] F. Rooms, M. Ronsse, A. Pizurica, and W. Philips. Psf estimation with applications in autofocus and restoration. In *IEEE Benelux Signal Processing Symposium, SPS'02*, Leuven, Belgium, March 2002.

[112] Z.-C. Ruan, D. Liang, and S. Wei. Panoramic mosaicing with image interpolation. In *IEEE Int. Symposium on Intelligent Multimedia, Video and Speech Processing, ISIMP'01*, pages 72 – 74, Hong Kong, May 2001.

[113] H. S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2D&3D dominant motion estimation for mosaicing and video representation. In *IEEE Int. Conf. on Computer Vision, ICCV'95*, pages 583 – 590, Boston, MA, June 1995.

[114] H. S. Sawhney and R. Kumar. True multi-image alignment and its applications to mosaicing and lens distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3):235 – 243, March 1999.

[115] R. Schäfer, G. Heising, and A. Smolic. Improving image compression - is it worth the effort? In *Xth European Signal Processing Conference, EUSIPCO'00*, Tampere, Finland, Sept. 2000.

[116] C. Schmid and A. Zisserman. The geometry and matching of curves in multiple views. In *IEEE European Conf. on Computer Vision, ECCV'98*, pages 394 – 409, Freiburg, Germany, June 1998.

[117] O. Schreer, P. Kauff, and T. Sikora. *3D Videocommunication: Algorithms, concepts and real-time systems in human centred communication*. John Wiley & Sons, Ltd, 1 edition, 2005.

[118] B. G. Schunck. *Motion segmentation and estimation*. Ph.D. thesis, M.I.T, 1983.

[119] H. Schwarz, D. Marpe, and T. Wiegand. Analysis of hierarchical B pictures and MCTF. In *IEEE Int. Conf. on Multimedia and Expo,ICME'06*, Toronto, Canada, 2006.

[120] C. A. Segall, R. Molina, and A. K. Katsaggelos. High resolution images from low-resolution compressed video. *IEEE Signal Processing Magazin*, 20(3):37 – 48, May 2003.

[121] J. Shen. Motion detection in color image sequence and shadow elimination. In *SPIE Visual Communications and Image Processing, 5308 VCIP'04*, pages 731 – 740, San Jose, CA, USA, Jan. 2004.

[122] H. Shum and R. Szeliski. Panoramic image mosaics. Technical Report MSR-TR-97-23, Microsoft Research, 1997.

[123] H. Shum and R. Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *IEEE Int. Conf. on Computer Vision, ICCV'98*, pages 953 – 958, Sept. 1998.

[124] H.-Y. Shum, S.-C. Chan, and S. B. Kang. *Image-Based Rendering*. Springer, 1 edition, 2006.

[125] T. Sikora. The MPEG-4 video standard verification model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):19 – 31, Feb. 1997.

[126] T. Sikora. MPEG digital video-coding standards. *IEEE Signal Processing Magazine*, 14(5):82 – 100, Sep 1997.

[127] T. Sikora. Trends and perspectives in image and video coding. *Proceedings of the IEEE*, 93(1):6 – 17, Jan. 2005.

[128] E. P. Simoncelli. *Distributed Representation And Analysis of Visual Motion*. Ph.D. thesis, M.I.T., 1993.

[129] A. Smolic. *Globale Bewegungsbeschreibung und Video Mosaiking unter Verwendung Parametrischer 2-D Modelle, Schätzverfahren und Anwendungen*. Ph.D. thesis, RWTH Aachen, 2001.

[130] A. Smolic. Robust generation of 360-degree panoramic views from consumer video sequences. In *4th EURASIP-IEEE Int. Symposium on VIPromCom*, pages 431 – 435, Zadar, Croatia, June 2002.

[131] A. Smolic and J.-R. Ohm. Robust global motion estimation using a simplified m-estimator approach. In *IEEE Int. Conf. on Image Processing, ICIP'00*, volume 1, pages 868 – 871, Vancouver, Canada, Sept. 2000.

[132] A. Smolic, T. Sikora, and J.-R. Ohm. Direct estimation of long-term global motion parameters using affine and higher order polynomial models for dynamic sprite coding. In *Proc. Picture Coding Symposium, PCS'99*, Portland, OR, April 1999.

[133] A. Smolic, T. Sikora, and J.-R. Ohm. Long-term global motion estimation and its application for sprite coding, content description, and segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1227 – 1242, May 1999.

[134] A. Smolic, Y. Vatis, H. Schwarz, P. Kauff, U. Gölz, and T. Wiegand. Improved video coding using long-term global motion compensation. In *SPIE Int. Conf. on Visual Communications and Image Processing, VCIP'04*, pages 343 – 354, San Jose, CA, Jan. 2004.

[135] A. Smolic, Y. Vatis, H. Schwarz, and T. Wiegand. Long-term global motion compensation for advanced video coding. In *ITG Proc. 10. Dortmunder Fernsehseminar, ITG/FKTG-Fachtagung*, Dortmund, Germany, Sept. 2003.

[136] A. Smolic and T. Wiegand. High-resolution video mosaicing. In *Int. Conf. on Image Processing, ICIP'01*, volume 3, pages 872 – 875, Thessaloniki, Greece, Oct 2001.

[137] C.V. Stewart. Robust parameter estimation in computer vision. *SIAM Reviews*, 41(3):513 – 537, Sept. 1999.

[138] C.-H. Su, H.-M. Hang, and D. W. Lin. Global motion parameter extraction and deformable block motion estimation. *Trans. Inf. & Syst.,*, E82-D(8):1210 – 1218, Sept. 1999.

[139] Z. Sun and A. M. Tekalp. Trifocal motion modeling for object-based video compression and manipulation. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):667 – 685, Sept. 1998.

[140] R. Szeliski. Image mosaicing for tele-reality applications. In *WACV'94*, pages 44 – 53, 1994.

[141] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics for Virtual Environments*, 16(2), March 1996.

[142] R. Szeliski and H. Shum. Creating full view panoramic image mosaics and environment maps. In *Int. Conf. on Computer Graphics and Interactive Techniques*, pages 251 – 258, 1997.

[143] S. Takeuchi, D. Shibuichi, N. Terashima, and H. Tominaga. Adaptive resolution image acquisition using image mosaicing technique from video sequence. In *IEEE Int. Conf. on Image Processing, ICIP'00*, volume 1, pages 220 – 223, Vancouver, Canada, Sept. 2000.

[144] B. Tannenbaum, R. Suryadevara, and S. Hsu. Evaluation of a mosaic based approach to video compression. In *IEEE Int. Conf. on Image Processing, ICIP'96*, volume 2, pages 1213 – 1215, Lousanne, Switzerland, Sept. 1996.

[145] A. Tchikangoua Toda. *Implementierung und Untersuchung verschiedener 2D-Bildtransfomationsmodelle für die globale Bewegungsschätzung (GME)*. Diplomarbeit, TU Berlin, 2004.

[146] P. Thévenaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739 – 758, July 2000.

[147] P. H. S. Torr. *Motion Segmentation and Outlier Detection*. Ph.D. thesis, Oxford University, 1995.

[148] L. Torres, M. Kunt, and F. Pereira. Second generation video coding schemes and their role in MPEG-4. In *European Conf. on Multimedia Applications, Services and Techniques*, pages 799 – 824, Louvian-la-Neuve, Belgium, May 1996.

[149] M. Traka and G. Tziritas. Panoramic view construction. *Signal Processing: Image Communication*, 18:465 – 481, March 2003.

[150] R. Y. Tsai. A versatile camera calibration technique for 3d machine vision. *IEEE Journal on Robotics and Automation*, (4):323 – 344, Aug. 1987.

[151] Y.-K. Tu, Y.-F. Yang, and M.-T. Sun. Efficient rate-distortion estimation for H.264/AVC coders. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(5):600 – 611, March 2006.

[152] J. Turski. Projective fourier analysis in computer vision: theory and computer simulations. In *Proc. SPIE, Vision Geometry VI, Robert A. Melter; Angela Y. Wu; Longin J. Latecki; Eds.*, volume 3168, pages 124 – 135, Oct. 1997.

[153] G.K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):18 – 34, Feb. 1992.

[154] H. Wallin, C. Christopoulos, and F. Furesjo. Adaptive resolution image acquisition using image mosaicing technique from video sequence. In *IEEE Int. Conf. on Image Processing, ICIP'01*, volume 2, pages 961 – 964, Thessaloniki, Greece, Oct. 2001.

[155] A. Wang, P. A. Chou, S. Mehrotra, and Z. Xiong. Three-dimensional wavelet coding of video with global motion compensation. In *IEEE Conference on Data Compression, DCC'99*, pages 404 – 413, Snowbird, Utah, March 1999.

[156] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625 – 638, Sept. 1994.

[157] T. Wiegand, N. Färber, K. Stuhlmüller, and B. Girod. Long-term memory motion-compensated prediction for robust video transmission. In *IEEE Int. Conf. on Image Processing, ICIP'00*, pages 152 – 155, Vancouver, BC, Canada, Sept. 2000.

[158] T. Wiegand, B. Lincoln, and B. Girod. Fast search for long-term memory motion-compensated prediction. In *IEEE Int. Conf. on Image Processing, ICIP'98*, pages 619 – 622, Chicago, IL, Oct. 1998.

[159] T. Wiegand, E. Steinbach, and B. Girod. Long-term memory prediction using affine motion compensation. In *IEEE Int. Conf. on Image Processing, ICIP'99*, pages 51 – 55, Kobe, Japan, Oct. 1999.

[160] 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.

[161] T. Wiegand, X. Zhang, and B. Girod. Motion-compensating long-term memory prediction. In *IEEE Int. Conf. on Image Processing, ICIP'97*, pages 53 – 65, Santa Barbara, CA, Oct. 1997.

[162] T. Wiegand, X. Zhang, and B. Girod. Long-term memory motion-compensated prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):70 – 84, Feb. 1999.

[163] G. Ye. *Image Registration and Super-resolution Mosaicing.* Ph.D. thesis, ADFA, University of New South Wales, Canberra, Australia, 2005.

[164] G. Ye, M. Pickering, M. Frater, and J. Arnold. Efficient multi-image registration with illumination and lens distortion correction. In *IEEE Int. Conf. on Image Processing, ICIP'05*, volume 1, pages 897 – 900, Genova, Italy, Sept. 2005.

[165] G. Ye, M. Pickering, M. Frater, and J. Arnold. A robust approach to super-resolution sprite generation. In *IEEE Int. Conf. on Image Processing, ICIP'05*, volume 3, pages 1108 – 1111, Genova, Italy, Sept. 2005.

[166] D. Zhang and G. Lu. Segmentation of moving objects in image sequence: A review. *Circuits, Systems, and Signal Processing*, 20(3):143 – 183, 2001.

[167] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(22):1330 – 1334, 2000.

[168] S. Zhilong and R. Qiuqi. Image registration based on klt feature tracker in image mosaicing application. In *IEEE Int. Conf. on Signal Processing, WCCC-ICSP 2000*, volume 2, pages 1143 – 1146, Beijing, China, Aug. 2000.

[169] B Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 24:977 – 1000, Feb. 2003.

[170] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR'97*, pages 420 – 425, San Juan, Puerto Rico, June 1997.

[171] A. Zomet and S Peleg. Applying super-resolution to panoramic mosaics. In *Workshop on Applications of Computer Vision*, pages 286 – 287, Oct. 1998.

[172] A. Zomet and S Peleg. Efficient super-resolution and applications to mosaics. In *15th Int. Conf. on Pattern Recognition, ICPR'00*, volume 1, pages 579 – 583, Sept. 2000.