RESEARCH ON KEY TECHNOLOGIES IN MULTIVIEW VIDEO AND INTERACTIVE MULTIVIEW VIDEO STREAMING

by

Xiaoyu Xiu

B.E., Beijing University of Technology, 2003

M.E., Beijing University of Technology, 2006

A Thesis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

in the
School of Engineering Science
Faculty of Applied Science

APPROVAL

Name:	Xiaoyu Xiu
Degree:	Doctor of Philosophy
Title of Thesis:	Research on Key Technologies in Multiview Video and Interactive Multiview Video Streaming

Examining Committee: Dr. Sami (Hakam) Muhaidat
Chair

____
Dr. Jie Liang, Senior Supervisor

____
Dr. Ivan V. Bajić, Supervisor

____
Dr. Mirza Faisal Beg, Supervisor

____
Dr. Jiangchuan Liu, Internal Examiner

____
Dr. Zhou Wang, External Examiner
Associate Professor of Electrical and Computer Engineering, University of Waterloo

Date Approved: 22 September 2011

Abstract

Emerging video applications are being developed where multiple views of a scene are captured. Two central issues in the deployment of future multiview video (MVV) systems are compression efficiency and interactive video experience, which makes it necessary to develop advanced technologies on multiview video coding (MVC) and interactive multiview video streaming (IMVS). The former aims at efficient compression of all MVV data in a rate-distortion (RD) optimal manner by exploiting both temporal and inter-view redundancy, while the latter offers a viewer the ability to freely interact with MVV data, such that she can periodically request her desired viewpoint as the video is played back. Based on the observation that MVC and IMVS are fundamentally different MVV problems, in this thesis, we focus on developing different algorithms for practical MVC and IMVS designs.

The first part of the thesis focuses on our research works on MVC. We first develop projective rectification-based view interpolation and extrapolation methods and apply them to MVC. Experimental results show that these schemes can achieve better RD performance than the current joint multiview video coding (JMVC) standard as well as view interpolation and extrapolation-based MVC schemes without using rectification. To explain the experimental results, we also develop mathematical models for the rectification-based view interpolation and extrapolation, from which we develop an improved theoretical model to compare the performances of various MVC schemes. Simulation results can verify the experimental results very well.

In the second part of the thesis, we propose three major technological improvements to existing IMVS works to enhance its interactivity experience and implement it in a realistic network condition. First, in addition to camera-captured views, we make available additional virtual views between each pair of captured views for viewers' selection, by transmitting both texture and depth maps of neighboring captured views and synthesizing intermediate views at decoder using depth-based image rendering (DIBR). Second, we construct a Markovian view-switching model that more accurately captures viewers' behaviors. Third, we optimize frame structures and schedule the transmission of frames in a network-delay-cognizant manner, so that viewers can enjoy zero-delay view-switching even over transmission network with non-negligible network delay.

Acknowledgments

During the completion of this doctoral work, I have been accompanied and supported by many people. It is my great pleasure to take this opportunity to thank all of them.

First of all, I would like to express my deepest gratitude to my senior supervisor, Professor Jie Liang, who influenced me most and guided me during the study towards my doctoral degree at Simon Fraser University. Dr. Liang is not only a great professor with knowledgable and deep vision but also and most importantly a kind person. He always provided me great opportunities, incredible encouragements as well as generous supports to pursue my research. Without his supervision, none of my research work in this thesis would have been finished.

I would also like to express my sincere gratitude to Professor Gene Cheung at National Institute of Informatics and Professor Antonio Ortega at University of Southern California, who introduced me to the field of interactive multiview video streaming. Their invaluable ideas and guidance have made a great impact to the completion of this thesis and my research work in future. I would like to acknowledge a generous financial support from National Institute of Informatics, Japan, for supporting me during four months of internship in Japan.

I am very grateful to my supervisors, Professor Ivan V. Bajić and Professor Mirza Faisal Beg, for their constructive advices and comments that have made substantial contributions to this thesis. Also, I have benefited enormously from their informative courses: Information Theory taught by Dr. Bajić and Computational Anatomy and Medical Image Analysis taught by Dr. Beg. Their inspiring lectures indeed broadened my eyes to interesting research fields and helped me solve several problems in my research. I would also like to thank Professor Jiangchuan Liu for serving as the internal examiner of my thesis.

I also gratefully thank the external examiner, Professor Zhou Wang at University of

Waterloo, and defense chair Professor Sami Muhaidat, for their precious time to attend the defense in the midst of their busy activities.

I am also thankful to all the amazing members at Multimedia Communication Laboratory, for making my life during the last four years a fun, challenging and memorable one. Especially, I would like to thank Upul Samarawickrama, Jing Wang, Yu Gao, Dong Zhang, Xiaozheng Huang and Calvin Che to name but a few, for their insightful comments and helpful discussions.

Finally, I would like to dedicate this thesis to my parents and my girl friend Lili Li, for their endless love, limitless encouragement and unselfish sacrifice throughout my doctoral education.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Multiview Video

Conventional two-dimensional (2D) video provides a fixed viewpoint of recorded objects where viewers can only watch a video playback passively, as the viewpoint remains the same throughout video playback. In contrast, *multiview video* (MVV) consists of video sequences of the same scene captured time-synchronously by multiple closely spaced cameras from different observation viewpoints. This means that each viewer of the same video content can observe various viewpoints of a scene from different angles and locations, which further generate a *free viewpoint video* (FVV) or create realistic three-dimensional (3D) perceptions.

In the past decade, we have witnessed a rapid evolution of MVV system thanks to the dramatic progress in computer, display and signal processing. Several prototypes of such MVV systems have demonstrated an much improved viewing experience compared to previous 2D video. Many people might remember the famous bullet-time effect from movie "The Matrix" [1] where multiple cameras were arranged on a pre-designed path, forming a complex curve through space. The cameras were then triggered at the same time, so the action continued to unfold in extreme slow-motion while the viewpoint moved. In addition, MVV system "EyeVision" [2] was used for broadcasting Super Bowl XXXV by CBS, where multiple video streams were captured using more than 30 custom-built, robotic pan tilt zoom cameras. These cameras were controlled jointly so that all cameras pointed, zoomed and focused synchronously on the same spot on the field. Then, the video sequences from different angles were combined to create virtual camera movements like frozen moment and view sweeping.

Both the systems "The Matrix" and "EyeVision" create MVV data by simply switching the captured viewpoints, *i.e.*, each frame from output video comes from one of few camera-captured views. However, some MVV systems, such as *free viewpoint television* (FTV), do not impose the constraint that a selected viewpoint corresponds to one existing camera, but instead, allows the selection of an arbitrary viewpoint within the 3D scene. Virtual intermediate views are synthesized from neighboring captured views using 3D geometry. These systems have many fields of application, such as to visualize and analyze sports or dynamic arts (*e.g.* traditional dance) actions.

Driven by the existing successful MVV applications and advancement in video acquisition hardware with decreasing price, we could reasonably predict that MVV-related products and services will see a vast growth in the market. Since movie "Avatar" was released, the box office earning from 3D movies increased from 11% in 2009 to 33% in 2010, according to International 3D Society. Networking giant Cisco also anticipated that 3D and high-definition videos will increase 13 times between 2009 and 2014, which could approximately comprise 42% of annual global IP traffic by 2014.

1.1.2 Multiview Video Coding

To enhance the perceived realism of the 3D output from MVV system, camera density around a scene needs to be highly enough, generating an enormous amount of MVV data which need to be stored or transmitted. Therefore, efficient *multiview video coding* (MVC) is necessary to compress the vast amount of MVV data for practical storage and transmission, which focuses on the efficient compression of all frames of all captured videos in a rate-distortion (RD) optimal manner by exploiting the inherent temporal (across time) and spatial (across view) correlation. In other words, the goal of MVC is to pursue the best efficiency of compressing the whole MVV data. Fig. 1.1 shows one MVC example proposed in [3], where I-frames are periodically inserted every Δ' frames to permit some level of random access.

Because MVC compresses all MVV data in an interdependent way, it can be suitable for applications which are oriented towards storage of the entire MVV data, or *non-interactive* delivery over networks, where viewers' potential interaction with the received MVV content does not affect how and what data is delivered. An example application of MVC is *3D Television* (3DTV) which needs to transmit and project all captured views on 3D displays

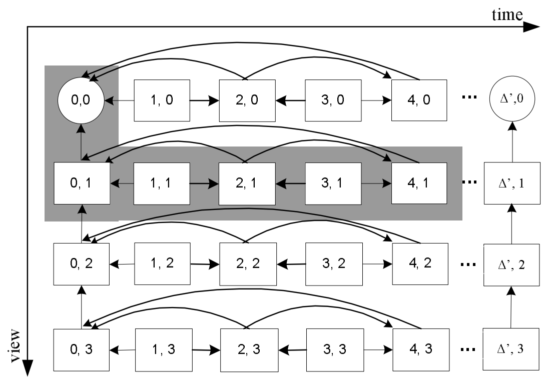


Figure 1.1: Example of MVC structure, where circles and rectangles denote I- and P-frames, respectively. The frames in the shape box represent the ones decoder can access to.

to provide viewers the perception of depth.

1.1.3 Interactive Multiview Video Streaming

One essential aspect of an immersive video experience is the ability for viewers to interact with a remote/virtual environment as if she is there. The viewer may interact naturally via a head-mounded tracking device [4], and an immersive communication system must in response quickly generate the data that corresponds to the observer's request. For instance, if the viewer tilts her head to the right, the viewpoint corresponding to the right-shifted viewing position must be rendered for observing in real-time. If the coded MVV data representing the environment already resides in the viewer's terminal device prior to interaction, then the subset of MVV data corresponding to the viewer's request can be simply fetched from memory, decoded and displayed. However, if the coded MVV data resides remotely in a server, then the transmission of the entire data set before viewer's interaction can be prohibitively expensive in bandwidth and delay.

Hence a more practical communication paradigm is the one where server continuously and reactively sends appropriate coded MVV data in response to a client's periodic requests for data subsets - this is called *interactive multiview video streaming* (IMVS) [5]. More specifically, after MVV data are pre-encoded and stored at the server, IMVS allows a client to periodically select and request switches to different viewpoints, as the requested single-view video is delivered from the server and played back in time uninterruptedly at the client. This is in stark contrast to the aforementioned non-interactive communication where the entire MVV data compressed by MVC is delivered server-to-client before a client interacts with the received data. IMVS has the advantage of reduced bandwidth utilization since only the requested data subset is transmitted. It can be used for a wide range of media modalities, such as interactive light field [6], interactive image browsing [7], flexible video playback [8].

Different from MVC, efficient compression of the sequence of the requested MVV data subset prior to the streaming session is a substantial challenge: the correlation that exists among requested MVV data subset is difficult to exploit because at the coding time, the order and selection of viewpoints chosen by the viewer during streaming time in future is unknown to encoder. Therefore, the existing coding structure used for MVC are not suitable for IMVS. Consider the example in Fig. 1.1 where the frames in the shape box are assumed to be available at the decoder as predictors, in order to switch from frame $F_{2,1}$ of time[1] 2 and view 1 to frame $F_{3,2}$ of next time 3 and neighboring view 2, server would send frames $F_{0,2}$, $F_{2,2}$, $F_{3,2}$ and $F_{4,2}$ to client, but only frame $F_{3,2}$ is displayed. Besides a large resulting transmission rate spike during the view-switch, this also incurs an unwanted overhead in decoding complexity.

Based on the above analysis, we can see that there is an inherent conflict between the view-switch interactivity and compression efficiency for IMVS, *i.e.*, free-navigation flexibility comes at the cost of lower coding efficiency. Meanwhile, due to various processing capability of terminal devices used by different interactive clients, the decoding complexity should be maintained at a low level for a IMVS decoder. Therefore, the destination of research in IMVS is to encode MVV data in such a way that a good compression efficiency is achieved while providing sufficiently flexibility for viewers to freely navigate the MVV data set in her desired order, and decoding tools with low complexity.

1.1.4 3D Representation Format

In the following, we will describe the various representation formats for MVV and discuss the merits and limitations of each in the context of both MVC and IMVS. A comparative analysis on different formats is also provided, based on which the specific formats used for the researches on MVC and IMVS in this thesis are selected respectively.

[1]For ease of discussion, we express time in number of frames for fixed video playback speed.

Texture-based Representation

The first class of MVV representation format relies on texture images of multiple views of the 3D scene, called *N-texture* representation format. It is also the basic representation format for the emerging MVC standard developed by Joint Video Team (JVT) of the ISO/IEC Moving Pictures Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG). One advantage of N-texture representation format is that no 3D geometric description of the scene is required. Therefore, only the texture images of multiple viewpoints should be necessarily encoded, resulting in less transmission bandwidth consumption and lower computational complexity at the encoder. Due to this advantage, we adopt N-texture representation format as the format for our research on MVC throughout the thesis.

While N-texture format would lead to an encoder with low complexity, such a MVV representation involves a high-complexity decoder. This is because as previously mentioned, a MVV system supports the selection of a large number of views (both captured views and virtual views), which makes it impractical to prepare all of these views before transmission. Instead, intermediate views should be interpolated using two neighboring captured views at the decoder via *image-based rendering* (IBR). To achieve high-quality view interpolation, 3D geometry of the scene is necessary, which has to be estimated from transmitted reference views, thereby imposing expensive computations on the decoder. Therefore, N-texture format is not suitable for IMVS which requires low-complexity decoder.

Depth-based Representation

The second class of MVV representation format is based on 3D geometric description of the scene. The 3D geometry is usually described by *depth map* or *depth image*, which specifies the distance between a 3D point in the space and its projection on one camera. The depth image may be extracted from a pair of captured views by solving for stereo correspondence [9] or obtained directly using time-of-flight cameras [10]. Using depth images, virtual views can be synthesized through *depth-based image rendering* (DIBR) techniques. For MVV applications such as 3DTV, it is usually assumed that the scene is observed from a narrow field of views (short baseline distance between cameras in a MVV system). Therefore, a combination of of only one texture plus one depth sequence is sufficient to provide good 3D rendering quality - we call this *1-texture/1-depth* representation format. However, for a scene with abundant 3D geometry, virtual views synthesized using 1-texture/1-depth format

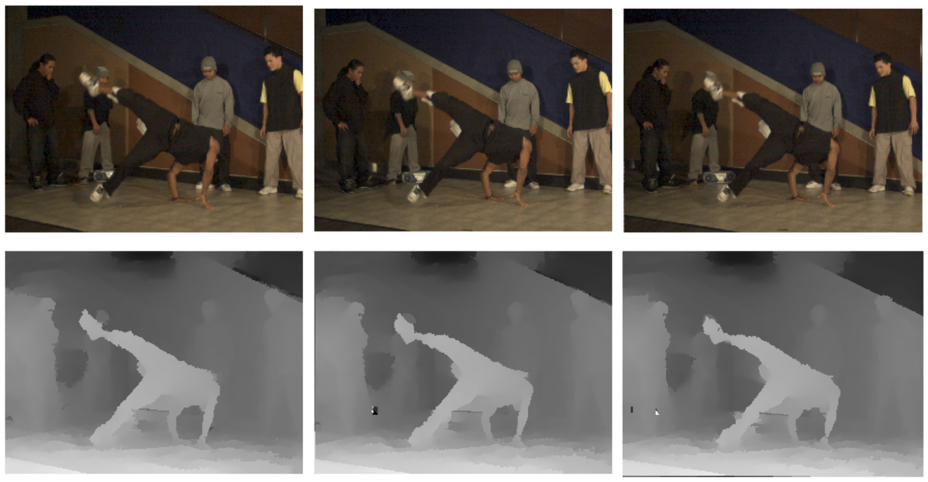


Figure 1.2: N-texture/N-depth representation format

typically show a large occluded region, which severely deteriorates the quality of viewers' 3D perception.

To address the problem of occlusion, it is proposed to combine N-texture format with 1-texture/1-depth format by using one depth image for each texture image, *i.e.*, *N-texture/N-depth* representation format; see Fig. 1.2 for an example. This approach has several advantages. First, as mentioned above, the occlusion problem can be efficiently solved by using multiple reference views which cover all regions observed from a virtual view. Second, compared to N-texture format, N-texture/N-depth format imposes less computational complexity on decoder. This is obvious, since all depth images are encoded and transmitted to reconstruct 3D geometry at decoder, correspondingly, an arbitrary virtual view can be directly synthesized after decoding both texture and depth images. Because of these advantages, in this thesis, we apply N-texture/N-depth format as the 3D representation method for our research on IMVS.

Stimulated by the increasing customer interests in immersive 3D experience, future MVV systems must employ advanced algorithms and technologies to offer higher compression efficiency and better viewer interactivity with MVV data. However, as what we have already analyzed, MVC and IMVS are fundamentally different MVV functionalities, thus the coding tools which are efficient for one problem may not provide a good solution for the other. This prompts us to develop different algorithms to efficiently support MVC and IMVS respectively. Furthermore, as what will be seen in the rest of the thesis, we take into consideration different types of view prediction techniques in our MVC design. Therefore, it is important to study theoretically the RD performance of various MVC algorithms, which can provide important guidelines for the design of practical MVC system.

Correspondingly, the primary goals of this thesis are to develop various compression techniques which could be used for the real MVC and IMVS designs, and obtain a better understanding of the theoretical performance of different MVC schemes. In particular, we

1. develop projective rectification-based view interpolation and extrapolation methods for view synthesis and apply them to MVC.

2. develop a geometric model for our view synthesis algorithms, and present an improved theoretical model to study the RD performance of various MVC schemes.

3. develop an improved depth estimation algorithm and apply it to IMVS.

4. develop three improvement techniques for the existing IMVS schemes to further enrich the functionality of interactivity and implement IMVS in a more realistic network setting.

1.2 Outline and Main Contributions

The focus of this thesis are to develop efficient methods for practical MVC and IMVS designs. In the case of MVC, our research contains two aspects of content. First, we develop advanced view synthesis algorithms to improve the efficiency of inter-view prediction in MVC. Second, we propose an analytical model to study the RD performance of various MVC schemes, which also help explaining the observations obtained from experimental results. In the case of IMVS, after finding several shortcomings in the existing IMVS system, we propose several significant improvements, aiming at a major enrichment of interactive viewing experience and practicality over previous IMVS schemes. The outline and main contributions of the thesis are listed as below.

In Chapter 2, we first provide a brief review of coherent background knowledge that will be extensively used throughout this thesis, including multiview geometry, IBR-based view synthesis, DIBR-based view synthesis and RD performance measures. We then overview the current progresses in MVC from the aspects of both practical designs and theoretical analysis. Finally, we discuss related work in IMVS.

In Chapter 3, we develop projective rectification-based view interpolation and extrapolation methods for view synthesis and apply them to MVC. Compared to most view synthesis methods assuming aligned cameras, our algorithms do not require camera parameters and have little requirement on camera setup, as long as the distance between the cameras is not too far. Moreover, using the proposed view synthesis algorithms, we develop two novel coding structures for MVC. Experimental results show that the proposed view synthesis schemes achieve better performance than existing methods, and lead to improved RD performance than the current MVC standard. An important observation is that although the quality of the extrapolated views is generally lower than that of the interpolated views, the average RD performance of view extrapolation-based MVC across all views can outperform the view interpolation-based MVC as the increase of the number of views, because view extrapolation can be applied to more views than view interpolation. The material of this chapter has appeared in [11, 12, 13].

In Chapter 4, we develop a geometric model for our view synthesis algorithms presented in Chapter 3, and present an improved theoretical model to analyze the RD performances of various MVC schemes. The analysis shows that view rectification can offer additional coding gain over direct view interpolation and view extrapolation-based MVCs, while proposed rectified view interpolation and extrapolation-based MVCs can outperform motion and disparity estimation-based MVC. These verify the experimental observations in Chapter 3. The material of this chapter has appeared in [11, 14, 15].

In Chapter 5, we develop an algorithm to generate a smooth and accurate depth map for view synthesis using 3D geometry and color segmentation. The proposed method presents several improvements over the conventional block-based depth estimation approach. First, a geometric prediction methodology is developed for accurate depth prediction, such that we can largely reduce the complexity of block-based depth matching. In addition, different from the conventional methods which can only estimate depth from a pair of cameras, a robust depth-consistency metric is proposed to synthesize the depth information from multiple views. Moreover, a color segmentation based depth-plane fitting algorithm guarantees the correctness and smoothness in textureless region. Experimental results show that the proposed algorithm can achieve marginal improvement of the synthesized view over the existing depth estimation methods. The material of this chapter has appeared in [16].

In Chapter 6, to further enrich viewers' interactivity with MVV data and implement IMVS in a more realistic network setting, we propose three improvements to existing IMVS schemes. First, using DIBR technique, our system enables free viewpoint switching, *i.e.*, we encode and transmit both texture and depth maps of captured views, allowing a client to select and synthesize any virtual view from an almost continuum of viewpoints between the left-most and right-most captured views. Second, we adopt a more realistic Markovian view-switching model that more accurately captures user behaviors. Third, we consider network delay of a client's view-switching request in practical networks, and optimize the frame structure of texture and depth maps in a network-delay-cognizant manner. We formalize the joint optimization of the frame encoding structure, transmission schedule, and quantization parameters of the texture and depth maps, and propose an iterative algorithm to achieve fast and near-optimal solutions. The convergence of the algorithm is also proved. Experimental results show that the proposed optimized rate allocation method requires significantly less transmission rate than the fixed rate allocation scheme. In addition, with the same storage, the transmission rate of the optimized frame structure can be reduced compared to heuristics-based structures. The material of this chapter has appeared in [17, 18, 19].

Finally, we summarize the research work of this thesis and make the conclusions in Chapter 7. The possible extensions are also discussed.

1.3 Acronyms and Notations

In this section, we summarize the acronyms and some common notations used throughout this thesis.

Table 1.1: List of acronyms

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
MVV	multiview video
FVV	free viewpoint video
3DTV	three-dimensional television
FTV	free viewpoint television
JVT	joint video team
MPEG	moving experts group
VCEG	video coding experts group
MVC	multiview video coding
IMVS	interactive multiview video streaming
JMVC	joint multiview video coding
IBR	image-based rendering

DIBR	depth-based image rendering
VSP	view synthesis prediction
MDE	motion and disparity estimation
RVI	rectification-based view interpolation
DVI	direct view interpolation
RVE	rectification-based view extrapolation
DVE	direct view extrapolation
BDE	block-based depth estimation
HBDE	hierarchical block-based depth estimation
LMMSE	linear minimum mean squared error
SEI	supplemental enhancement information
QP	quantization parameter
RD	rate-distortion
RQ	rate-quantization
DQ	distortion-quantization
GOP	group of picture
FT	fourier transform
KLT	karhunen-lòeve transform
DCT	discrete cosine transform
CABAC	context-adaptive binary arithmetic coding
RANSAC	random sample consensus
SIFT	scale invariant feature transform
DLT	direct linear transform
SP	switching P
DSC	distributed source coding
LDPC	low-density parity check codes
PDF	probability density function
PMF	probability mass function
PSD	power spectral density
MSE	mean-squared-error
PSNR	peak-signal-to-noise ratio
SSIM	structural similarity
SRNL	source residual noise level
TRNL	temporal residual noise level
IVRNL	inter-view residual noise level
VSRNL	view synthesis residual noise level
RTT	round trip time

Table 1.2: List of notations

$\mathcal{R}$	The set of real numbers
$\mathcal{Z}$	The set of integer numbers
$\mathcal{R}^+$	The set of positive real numbers
$\mathcal{Z}^+$	The set of positive integer numbers
$\mathbf{A}$	A boldface letter denotes a matrix
$\mathbf{I}$	The identical matrix
$\mathbf{0}$	The null matrix
$\mathbf{A}^T$	The transpose of a matrix $\mathbf{A}$
$\mathbf{A}^H$	The conjugate of a matrix $\mathbf{A}$
$\mathbf{A}^\dagger$	The conjugate transpose of a matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	The inverse of a matrix $\mathbf{A}$
$\|\mathbf{A}\|$	The determinant of a matrix $\mathbf{A}$
$E(x)$	the expected value of a random variable x

Chapter 2

Review

To facilitate the understanding of our contributions, this chapter reviews some important concepts and current progresses related to the work in this thesis. We begin with a brief introduction of some background knowledge that are extensively used throughout the thesis. We then discuss the related work in both MVC and IMVS as the motivational background material.

2.1 Background Knowledge

2.1.1 Epipolar Geometry

The *epipolar geometry* is the intrinsic projective geometry between two viewpoints. It is independent of scene structures, and only replies on the cameras' internal parameters and relative pose. Suppose a 3D point $\mathbf{X}$ [1] is projected through two camera centers $\mathbf{C}_1$ and $\mathbf{C}_2$ onto two camera planes I_1 and I_2 at pixel positions $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively (see Fig. 2.1). It is obvious that 3D points $\mathbf{X}$, $\mathbf{C}_1$, $\mathbf{C}_2$ and the projection points $\mathbf{x}_1$ and $\mathbf{x}_2$ are located within the same plane π which is called *epipolar plane*. Further, the epipolar plane π constrains the range of searching correspondences in two camera planes. More specifically, given projection point $\mathbf{x}_1$ in I_1, its correspondence point $\mathbf{x}_2$ has to lie on the intersection between the plane π and the camera plane I_2. The intersection of the two planes corresponds to a line $\mathbf{l}_2$ known as *epipolar line*. Therefore, the search of correspondences can now be limited along

[1]In homogeneous coordinate expression, 3D and 2D points are represented by 4×1 and 3×1 column vectors respectively.

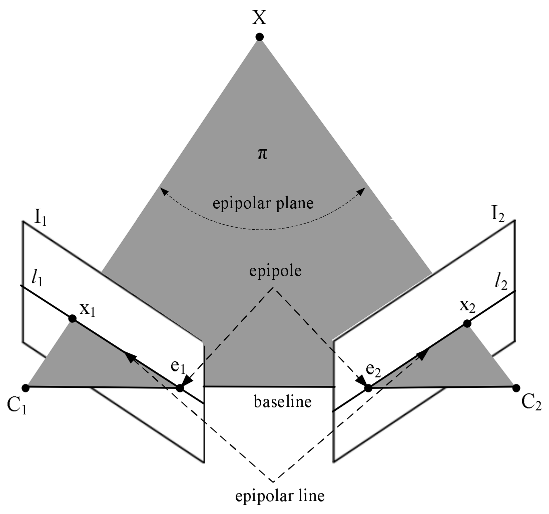


Figure 2.1: Epipolar geometry between two views with an indication of terminologies.

the epipolar line instead of in the entire camera plane.

The epipolar geometry is often described by a 3×3 *fundamental matrix* $\mathbf{F}$ of rank 2, which associates two projection points $\mathbf{x}_1$ and $\mathbf{x}_2$ by the relation

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0 \tag{2.1}$$

Equation (2.1) is a linear function of the entries of $\mathbf{F}$. If enough correspondence points between two views are known, various algorithms can be used to calculate $\mathbf{F}$, such as the 7-point, the 8-point or the least square algorithm [20, 21]. In addition, as the point $\mathbf{x}_2$ belongs to the epipolar line $\mathbf{l}_2$, *i.e.*, $\mathbf{x}_2^T \mathbf{l}_2 = 0$, we can thus have

$$\mathbf{l}_2 = \mathbf{F} \mathbf{x}_1 \tag{2.2}$$

We now introduce some terminologies related to epipolar geometry, which will be employed further in this thesis.

- The *epipolar plane* is the plane determined by a 3D point and the two camera centers.
- The *baseline* is the line connecting two camera centers
- The *epipole* is the point of intersection of the baseline with one camera plane. Equivalently, the epipole is the image in one view of the camera center of the other view.
- The *epipolar line* is the intersection of an epipolar plane with one camera plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right camera plane in epipolar lines, and defines the correspondence between two cameras.

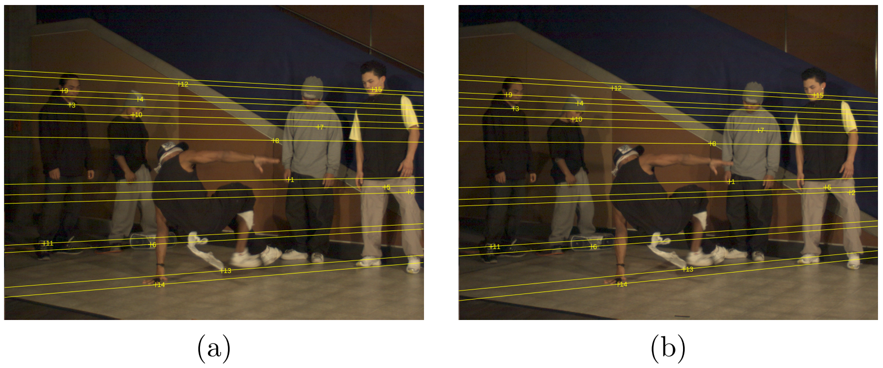


Figure 2.2: A pair of images with superimposed correspondences and their epipolar lines. (a) Left view; (b) Right view.

An example of two views with several pairs of correspondences along with the corresponding epipolar lines is shown in Fig. 2.2.

2.1.2 IBR-based View Synthesis

Image-based rendering (IBR) refers to techniques and representations that allow 3D objects to be virtualized in a realistic way without using 3D geometry of the scene. In IBR-based view synthesis, an intermediate virtual view can be synthesized from images at two neighboring captured views via *view interpolation*. View interpolation has been extensively studied in computer vision and computer graphics, which first calculates a *disparity map*, and then renders an interpolated view. A diagram of procedures in view interpolation is shown in Fig. 2.3, where view I_n is synthesized using two adjacent views I_{n-1} and I_{n+1}, left and right to the synthesized view. Because most view interpolation approaches are designed for stereo vision, they assume aligned camera setup, *i.e.*, the two reference cameras are parallel and only differ from each other by a small horizontal shift.

Disparity Estimation

Since 3D geometry is not used for IBR-based view synthesis, a disparity map needs to be estimated for the 3D geometric description, which indicates the horizontal distance between each pair of correspondence points in two reference views (see Fig. 2.3(a)).

As one classical problem in computer vision, disparity estimation, or *stereo matching*, currently still remains active and a lot of algorithms have been proposed in recent decades. For instance, a disparity map was calculated in [22] by a measure of pixel dissimilarity

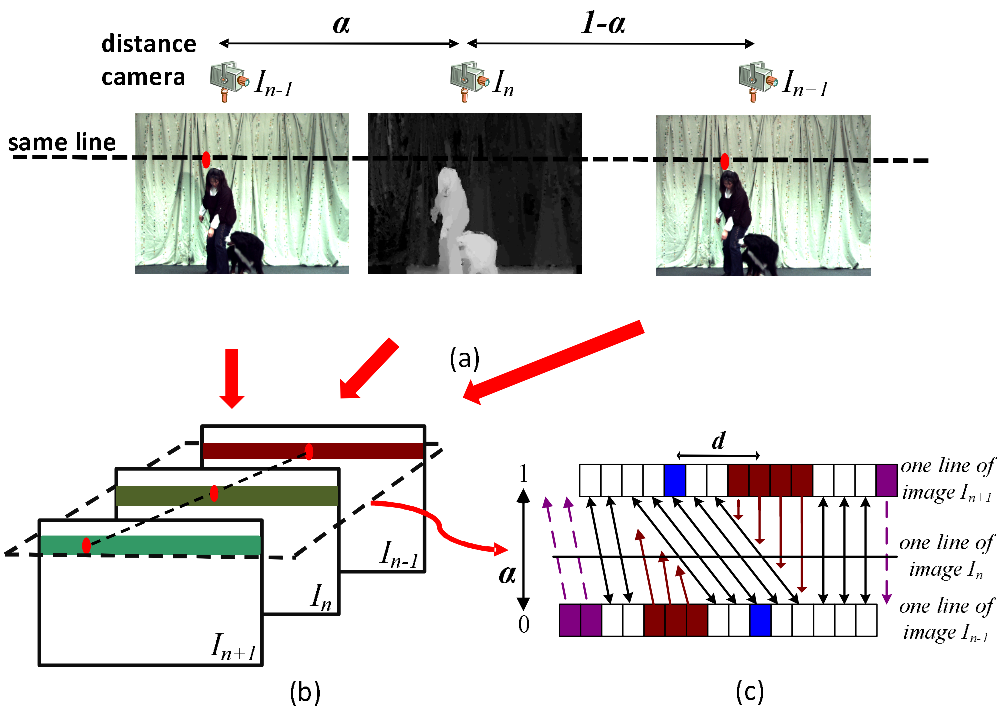


Figure 2.3: Procedures of IBR-based view interpolation of the intermediate view I_n from the left and right reference views I_{n-1} and I_{n+1}. (a) Disparity estimation; (b) View interpolation; (c) Details of interpolating pixels of one line.

that was insensitive to image sampling, and one dimensional (1D) dynamic programming was also applied to accelerate the speed of the algorithm. But, independent processing of different scan-lines led to the inconsistency between scan-lines in the disparity map. In [23], the problem was addressed by computing the disparity from the top left pixel to the bottom right pixel under an energy function that took the smoothness of disparity transition into account. Although the algorithm was further improved in [24] with belief propagation, the complexity of the algorithm was increased. Graph cuts based matching algorithm was adopted in [25], which achieved more accurate disparity estimation, but they cannot handle occlusions well, because the two images were treated asymmetrically, and no constraint was imposed to ensure that a pixel corresponded to at most one pixel in the other image. To handle occlusions, two disparity maps were calculated in [26] from the left and right pictures using the novel graph-cut algorithm.

View Interpolation

Once disparity map is obtained, view interpolation can be applied to create the synthesized view, as shown in Fig. 2.3(b). Let $I_{n-1}(x, y)$, $I_n(x, y)$ and $I_{n+1}(x, y)$ be the pixel values of views I_{n-1}, I_n and I_{n+1} at position (x, y), respectively, and $d(x, y)$ the disparity of I_{n+1} with respective to I_{n-1}. Depending on correspondence condition, the intermediate view can be interpolated by considering the following three cases.

- *Pixels that are visible in both reference views.* As shown by the white squares in Fig. 2.3(c), each of these pixels have a disparity value assigned. Therefore, the positions of the pixels in the intermediate view can be easily derived from simply scaling the disparity values by the intermediate view position $\alpha \in (0, 1)$. And, the pixel values of the intermediate pixels in the synthesized view can also be interpolated from the correspondences in the left and right reference views. This can be mathematically written as

$$I_n(x - \alpha d(x, y), y) = (1 - \alpha)I_{n-1}(x, y) + \alpha I_{n+1}(x - d(x, y), y) \tag{2.3}$$

- *Pixels that are occluded from on reference view.* These pixels have no disparity value assigned, since they are explicitly detected as occlusions (see brown squares in Fig. 2.3(c)). However, as seen in Fig. 2.3(c), the occlusion pixels in the left reference view (view I_{n-1}) are occluded by objects at the right side, and the occlusion pixels in the right reference view (view I_{n+1}) are occluded by objects at the left side. Assuming that occluded pixels are parallel to the image plane, we can extend the disparities of the background into the occluded areas (see brown arrows in Fig. 2.3(c)). Further, the pixels in the synthesized view are directly copied from the corresponding pixels in the reference view. For instance, to interpolate intermediate pixels in the synthesized view I_n using occlusion pixels in the left view I_{n-1}, the disparity of the available pixel left to the occluded area is used, *i.e.*,

$$I_n(x - \alpha d(x_l, y), y) = I_{n-1}(x, y) \tag{2.4}$$

  where x_l is the first visible pixel left to the occlusion area in view I_{n-1}.

- *Pixels whose correspondences are outside the valid region of the other reference view.* For these pixels, no disparity was available due to the restricted image area of one

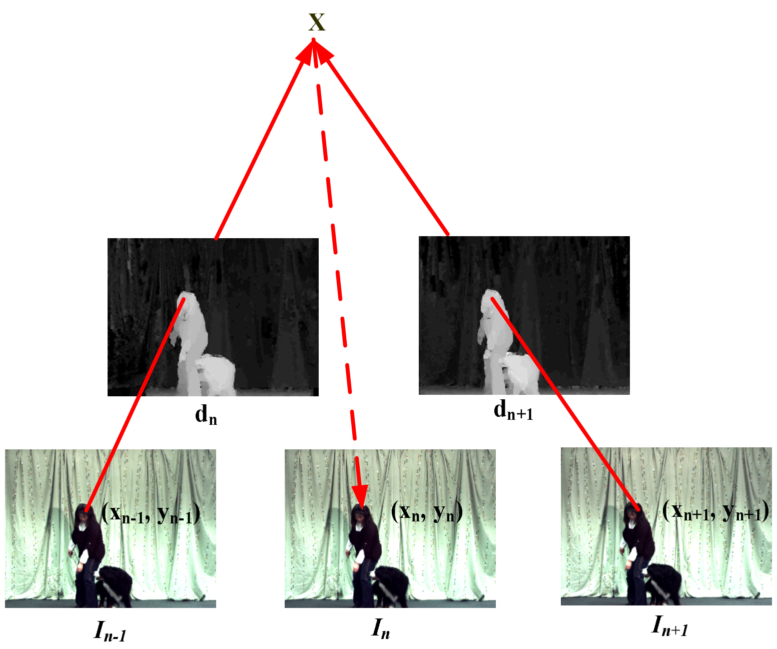


Figure 2.4: DIBR-based view synthesis

viewpoint (see purple squares in Fig. 2.3(c)). Similar to occluded pixels, we extend the disparity of border pixel into undefined region, and pixel values are copied accordingly. For instance, if the correspondence of $I_{n-1}(x, y)$ is outside view I_{n+1}, the interpolated pixel is then taken as

$$I_n(x - \alpha d(x_r, y), y) = I_{n-1}(x, y) \tag{2.5}$$

where x_r is the pixel in view I_{n-1}, which is just right to the region whose correspondences are invalid in view I_{n+1}.

2.1.3 DIBR-based View Synthesis

Depth-image-based rendering (DIBR) is the process of synthesizing virtual views of a scene from captured images and associated per-pixel depth information [27, 28, 29, 30]. Conceptually, this view generation can be implemented by the following two steps. At first, the original pixel points of the captured view are projected into the 3D space, using the respective depth data. Then, those 3D points are re-projected into the image plane of the virtual view. And, this concatenation of 2D-to-3D projection and 3D-to-2D projection is usually called *3D warping*.

Using the same camera setup in Fig. 2.3, Fig. 2.4 shows the procedures of DIBR-based view synthesis, where an arbitrary 3D space point $\mathbf{X}$ is assumed to be projected to (x_{n-1}, y_{n-1}), (x_{n+1}, y_{n+1}) and (x_n, y_n) in the two reference views I_{n-1} and I_{n+1}, and synthesized view I_n, respectively. For simplicity, let us consider 3D warping between views I_{n-1} and I_n, with the two perspective projection equations [20] as

$$\begin{aligned} d_{n-1}\mathbf{m}_{n-1} &= \mathbf{A}_{n-1}\mathbf{R}_{n-1}^{-1}\left(\mathbf{X} - \mathbf{T}_{n-1}\right) && (2.6) \\ d_n\mathbf{m}_n &= \mathbf{A}_n\mathbf{R}_n^{-1}\left(\mathbf{X} - \mathbf{T}_n\right) && (2.7) \end{aligned}$$

where $\mathbf{m}_{n-1} = (x_{n-1}, y_{n-1}, 1)^T$ and $\mathbf{m}_n = (x_n, y_n, 1)^T$ are two projection points in homogeneous notation. d_{n-1} and d_n are the corresponding depth values. The matrices $\mathbf{R}_{n-1}$ and $\mathbf{R}_n$, $\mathbf{T}_{n-1}$ and $\mathbf{T}_n$ specify the respective rotation and translation of views I_{n-1} and I_n from the origin of the world coordinate, and matrices $\mathbf{A}_{n-1}$ and $\mathbf{A}_n$ denote the intrinsic parameters of two views.

Rearranging (2.6) gives an affine representation of 3D point $\mathbf{X}$ from its projection in I_{n-1} as

$$\mathbf{X} = d_{n-1}\mathbf{R}_{n-1}\mathbf{A}_{n-1}^{-1}\mathbf{m}_{n-1} + \mathbf{T}_{n-1} \tag{2.8}$$

Substituting (2.8) into (2.7) then leads to the depth-dependent relation between the projections of the same 3D point in two views:

$$d_n\mathbf{m}_n = \mathbf{A}_n\mathbf{R}_n^{-1}\left(d_{n-1}\mathbf{R}_{n-1}\mathbf{A}_{n-1}^{-1}\mathbf{m}_{n-1} + \mathbf{T}_{n-1} - \mathbf{T}_n\right) \tag{2.9}$$

(2.9) is a powerful formula which describes how to generate an arbitrary view from a known reference view. More specifically, if the depth values of all the points of a 3D scene are known for every pixel of the reference view I_{n-1}, the virtual view can be rendered by projecting all pixels in view I_{n-1} to the virtual viewpoint I_n using (2.9).

As partial regions of one virtual view could be invisible in the reference view, two adjacent views left and right to the virtual view are used for DIBR, as shown in Fig. 2.4, by applying (2.9) to both views I_{n-1} and I_{n+1}. And, the final synthesized view is generated by merging the projected views from the two reference views into one [31].

In spite of two reference view used for DIBR, the resultant virtual view could still contain small regions which are invisible from both two references, due to occlusion. In this case, the missing areas are usually filled by image impainting methods from neighboring projected pixels [32].

2.1.4 RD Performance Measures

In the information-theoretical aspect, upon knowing the *power spectral density* (PSD) function of an image, it is possible to derive the corresponding RD performance of encoding the image. The RD function of a 2D stationary Gaussian random process v with PSD function $\Phi_{vv}(\omega_x, \omega_y)$ is given in parametric form, with rate

$$R(\theta) = \frac{1}{8\pi^2} \int_{\omega_x=-\pi}^{\pi} \int_{\omega_y=-\pi}^{\pi} \max \left[0, \log_2 \frac{\Phi_{vv}(\omega_x, \omega_y)}{\theta}\right] d\omega_x d\omega_y \tag{2.10}$$

and distortion

$$D(\theta) = \frac{1}{4\pi^2} \int_{\omega_x=-\pi}^{\pi} \int_{\omega_y=-\pi}^{\pi} \min \left[\theta, \Phi_{vv}(\omega_x, \omega_y)\right] d\omega_x d\omega_y \tag{2.11}$$

where $\theta \in \mathbb{R}^+$ is the parameter controlling to produce different RD points[33]. In the case of predictive coding, residue signal $e = v - v'$ should be used instead of v, where v' is the prediction signal.

Thus, the RD performance of encoding any stationary Gaussian signal can be determined by (2.10) and (2.11). However, at high bit rate, the energy of the quantization error asymptotically tends to zero, correspondingly, the effect of quantization error on RD performance can be neglected. In [34, 35], a relative performance measure, called *rate difference*, is used to compare different coding schemes at high rate, which is shown by

$$\begin{aligned} \Delta R &= R_e(\theta) - R_v(\theta) \\ &= \frac{1}{8\pi^2} \int_{\omega_x=-\pi}^{\pi} \int_{\omega_y=-\pi}^{\pi} \log_2 \frac{\Phi_{ee}(\omega_x, \omega_y)}{\Phi_{vv}(\omega_x, \omega_y)} d\omega_x d\omega_y \end{aligned} \tag{2.12}$$

which can be achieved by substituting (2.10) into the first line of (2.12) by assuming θ to be a very small value. Note that ΔR represents the bit saving per sample from predictively encoding the residue signal e instead of independently encoding the original signal v, which therefore is negative. In Chapter 4, rate difference will be used as the performance measure for the evaluation of different MVC algorithms.

2.2 Related Works on MVC

The different coding techniques that are developed for MVC are reviewed in this section. This includes methods which make use of existing 2D video coding, as well as methods with disparity estimation. Then, coding techniques based on view synthesis prediction (VSP) are also covered with a review of algorithms specific to view interpolation and DIBR-based view synthesis. We finally summarize the current progress on theoretical MVC analysis.

2.2.1 2D Video Coding-based MVC

Simulcast of Multiview

The most straightforward mean to compress multiview video is to encode each view independently of the other, *e.g.*, using state-of-the-art video encoder H.264/AVC [36]. This solution, also known as *simulcast*, achieve the minimum of computation and decoding delay since dependencies between different views are not exploited.

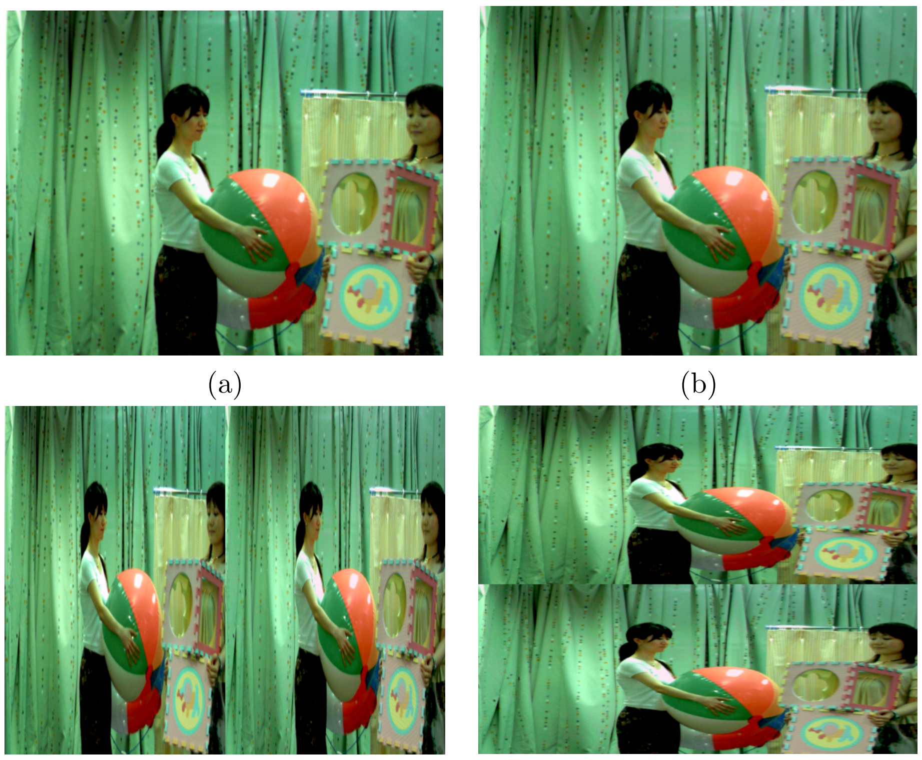
The main drawback of simulcast solution is that coding efficiency is not maximized because redundancy between views, *i.e.*, inter-view redundancy, is not considered. However, in the case of stereo coding, prior studies on asymmetrical coding where one of the views is encoded with less quality, show that the substantial bit rate saving for the second view could be achieved. Therefore, one of the views can be low-pass filtered, more coarsely quantized than the other view [37]. However, viewers could suffer from eye fatigue when viewing asymmetrically coded video for a long period due to unequal watching quality to each eye. Correspondingly, it has been proposed in [38] to periodically switch the asymmetrical coding quality between the left and right eyes. But, further researches on how asymmetric coding affects human visual perception are still needed.

Frame-compatible Coding

To facilitate the introduction of stereoscopic services through the existing equipment, frame-compatible format is proposed, where the stereo signal is essentially a multiplex of two views into a single sequence. Usually, the left and right views are sub-sampled and interleaved into a single frame.

There are various options of both sub-sampling and interleaving. As shown in Fig. 2.5, the two views could be filtered and decimated horizontally or vertically and stored in a side-by-side, top-and-bottom formats. Temporal format is also possible, where the left and right views are interleaved as frames. In this way, frame rate of each view is reduced so that the amount of data is equivalent to that of a single view.

Frame-compatible format can work seamlessly within existing video decoders. In an



(a) (b) (c) (d)

Figure 2.5: Full resolution and frame-compatible representation of stereoscopic videos. (a) Full resolution left view; (b) Full resolution right view; (c) Side-by-side; (d) Top-and-bottom.

effort to better facilitate its adoption, the H.264/AVC standard introduces a new Supplemental Enhancement Information (SEI) message which enables signaling the sampling relationship between two views. By detecting the SEI message, decoder can immediately identify the format and perform accordingly, such as scaling, denoising, based on the frame-compatible format specified.

Due to minimal changes, frame-compatible coding can make stereo services quickly deployed in the market. However, the obvious drawback of this method is that spatial or temporal resolution would be lost, due to spatial or temporal sub-sampling. However, the impact of frame-compatible coding on 3D perception may be limited and acceptable for initial services.

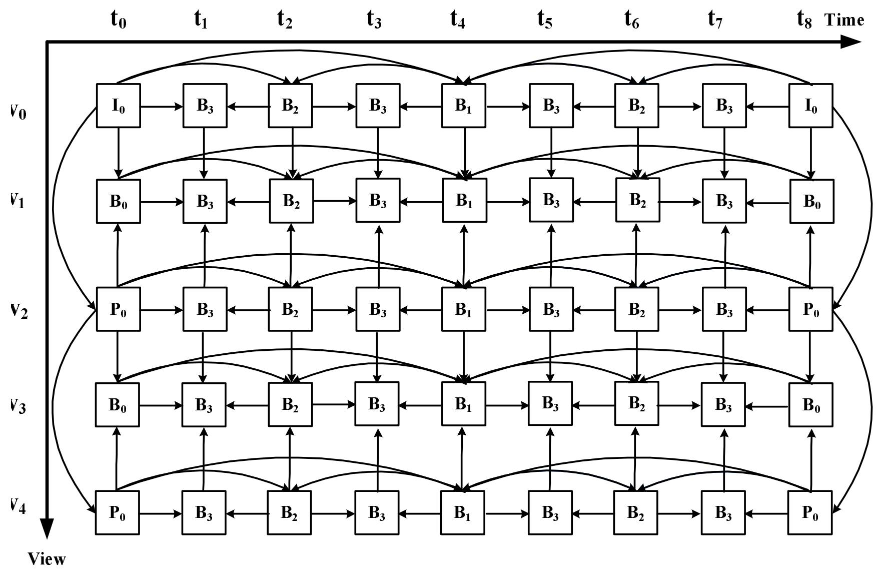


Figure 2.6: Typical hierarchical B structure for JMVC

2.2.2 Motion and Disparity Estimation-based MVC

To improve coding efficiency of MVC, both temporal and inter-view redundancy should be exploited. In this way, pictures are not only predicted from temporal reference pictures, but also from inter-view reference pictures, through motion and disparity estimation (MDE).

The concept of inter-view prediction using MDE-based prediction, was first proposed in [39] and subsequently adopted in amendment of MPEG-2. The concept of group of GOP for MDE-based prediction was introduced in [40], which allowed a picture to refer to decoded pictures of other views even at different time instants. In [3, 41], various modified hierarchical B structures were developed for MDE-based prediction. Most recently, the H.264/AVC standard was modified based one of those structures to support joint multiview video coding (JMVC) software, which used the hierarchical B structure in the temporal direction and the I-B-P disparity prediction structure in the inter-view direction, as shown in Fig. 2.6. In JMVC, inter-view prediction was enabled through flexible reference picture management, where decoded pictures were made available in the reference picture lists. Moreover, since coding decisions were adaptively made on the block level, a block in a particular view could be predicted by a temporal reference, while another block in the same view could be predicted by an inter-view reference. To reduce the complexity of finding the best matching, the multiview geometry was employed in [42] to predict the disparity values, but only multiview image coding was considered.

It has been shown that MVC with disparity estimation-based prediction can give significant improvement over simulcast [3]. A comprehensive set of results for MVC over a broad range of test material was presented in [43], showing that an average of 20% reduction in bit rate relative to the total simulcast bit rate can be achieved with equal reconstructive quality of each view.

2.2.3 VSP-based MVC

Albeit disparity estimation-based MVC can offer significant coding gain over simulcast, the translational inter-view motion assumed by the disparity estimation method could not accurately represent the geometric relationships between different cameras; therefore, this method is not always efficient. For example, larger disparities than the search window size can frequently occur, due to different depths of an object in different views. In addition, effects such as rotation and zooming are difficult to be modeled as pure translational motion.

To enhance the efficiency of inter-view prediction, VSP can be applied, where a synthesized view for a target view is created, using the geometry relationship between different views. The synthesized view is then used as an additional reference to predictively encode the target view. Depending on the view synthesis methods (IBR or DIBR) used, VSP-based MVC can be classified into two categories, *i.e.*, depth-based MVC and view interpolation-based MVC.

Depth-based MVC

In depth-based MVC, synthesized views are created using the corresponding depth maps via DIBR. Therefore, the crucial issue of depth-based MVC is the compression of depth map, *i.e.*, coding tools used for depth data which yield high compression efficiency. Here, different characteristics of depth in comparison to video data must be considered. A depth signal is composed of homogeneous areas inside objects and sharp transitions along boundaries between objects. In this sense, typical video compression algorithms which are designed to preserve low-frequency information and blur images at high compression rate, are not very suitable for depth map coding. In addition, as a depth value can be mapped to a shift value of a texture sample from original reference view, coding errors in depth maps result in wrong pixel shifts in synthesized view, especially on objects' boundaries. Therefore, depth compression algorithms should preserve depth edges better than video coding methods.

However, the initial attempts on depth map coding still used conventional video coding schemes, such as H.264/AVC or JMVC, to encode depth [44]. But, these schemes aimed at not only the RD optimization on depth map coding, but also the quality of synthesized views. In order to increase efficiency of depth map coding, down-sampling before depth encoding was introduced in [45]. After decoding, an up-sampling process was applied, followed by edge refinement based on the objects' contour to preserve boundaries in depth map. A platelet coding method was proposed for depth map coding in [46], where occurrences of foreground and background boundaries were analyzed and approximated by a simple linear function. Finally, each block with boundaries contained two areas, one representing foreground depth and the other representing background depth.

Although depth-based MVC can generate synthesized views with an improved quality relative to that of disparity estimation-based MVC, it leads to a considerable increase of the bandwidth for delivering MVV data, due to the overheads for encoding and transmitting the depth map. It was shown in [47] that an independent transmission of 8-bit depth increased 10% bandwidth consumption on top of texture data coding.

View Interpolation-based MVC

Besides depth-based prediction, view interpolation can also be used for MVC. As illustrated in Sec. 2.1.2, view interpolation-based MVC needs two neighboring views to synthesize a virtual reference for predictively encoding the target view, where disparity map is first calculated and then each pixel in an intermediate view position is interpolated from a pair of correspondences in the left and right views. As decoded frames of both two neighboring views are used for disparity estimation at encoder, the exactly same disparity map can be reproduced at decoder. Thus, no extra bits are sent on transmitting disparity map for view interpolation-based MVC.

View interpolation-based MVC was first used for MVC in [48], which interpolated frames at given time instants and view positions and use them as reference. In addition, one color correction approach was proposed to correct luminance and chrominance values for compensating the variance between different capturing cameras and enhancing the efficiency of view interpolation-based prediction. However, this approach was restricted to aligned camera setup, where all cameras differ from each others only by horizontal shifts.

To deal with more general camera setups, a rectification-based view interpolation (RVI) method was proposed in [49]. It first rectified the two views using the projective rectification method in [20, 21]. This involved the calculation of the fundamental matrix between two views and re-sampling of them such that they have horizontal and matched epipolar lines. A modified version of the disparity estimation method in [22] was then used to create the interpolated view before mapping back to the original domain. The algorithm did not require camera parameters, and had little requirement on camera setup, therefore was suitable for MVV systems with unaligned cameras and unknown camera parameters. In Chapter 3, by modifying the methods in [21, 49], we develop an improved RVI method and apply it to MVC.

The aforementioned view interpolation-based MVC methods deal with view synthesis from a left view and a right view. If these methods are used in MVC, VSP can only benefit half of the views. To overcome this limitation, in Chapter 3, we also develop a rectification-based view extrapolation (RVE) algorithm using two left views or two right views. Hence, VSP can be applied to the coding of all views after the first two. Our results show that although the average quality of the extrapolated views is lower than that of the interpolated views, the overall RD performance of all views of RVE-based MVC can outperform that of RVI-based MVC, as the increase of the number of coded views.

2.2.4 Theoretical Analysis of MVC

Another important topic of MVC is the theoretical RD analysis of different MVC algorithms. Such an analysis can provide important guidelines for the design of practical MVC systems. The RD analysis can be achieved by generalizing that of the traditional 2D single view video coding. The key problem is how to model the inter-view correlations and the underlying inter-view prediction algorithm.

The theory of the RD analysis of motion compensation-based single view video coding was established by Girod [34, 50, 51]. It was generalized to wavelet based video coding in [52] and light field coding in [53], where the impacts of the statistical properties of multiple light field images, the accuracy of the disparity and the transform coding on the compression efficiency were studied. In [54], the RD analysis of multiview image coding with texture-based and model-aided methods was presented, using the same model as in [53]. Recently, these theories were generalized to MVC in [35], where the RD efficiency of MDE-based MVC was investigated by assuming a pair of translatory shifts to model the effects of motion and disparity estimation.

However, the RD analysis of VSP-based MVC has not been reported in the literature. In fact, even the mathematical models of view synthesis algorithms have not been well established. A couple of important progresses toward this direction were obtained recently. In [55], a model was proposed to describe the relationship between the accuracy of disparity and the quality of the interpolated view, based on the framework in [50, 53, 54]. A pre-filter method was also proposed to improve the view interpolation quality. However, the model for the disparity error was oversimplified, and only parallel cameras were considered. In [56], a similar model for view interpolation was used to analyze the theoretical RD performance of a multiview image coding scheme based on view sub-sampling. However, MVC is not considered in both [55] and [56].

In Chapter 4, we develop a more accurate geometric model than that in [55]. Our model enables the study of the impact of projective rectification on the quality of the interpolated or extrapolated view when unaligned cameras are used. To the best of our knowledge, this is the first attempt to quantify the improvement of the projective rectification in view synthesis.

Another contribution in Chapter 4 is that we develop an improved RD model to study the performances of different practical MVC schemes, *e.g.*, MDE-based JMVC and VSP-based schemes. Compared to the models in [35], our model characterizes practical MVC schemes more accurately. Simulation results of this model agree well with the experimental results of various MVC schemes discussed in Chapter 3.

2.3 Related Works on IMVS

In this section, we divide our discussion on related IMVS work into two parts. We first discuss some related works on multiview image/video streaming, especially emphasizing their difference from IMVS. Then, we review current progresses in IMVS.

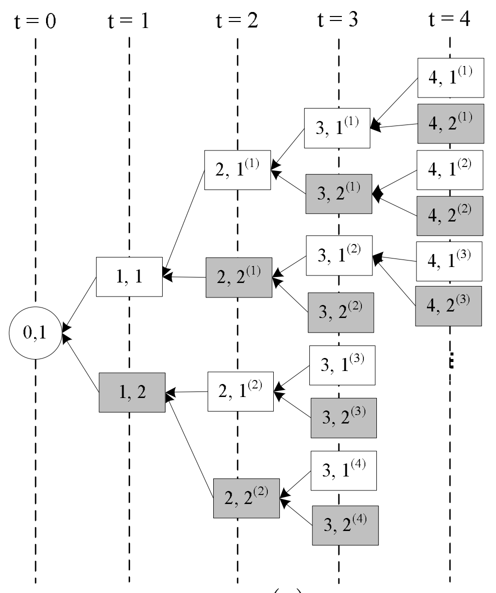
2.3.1 Multiview Image/Video Streaming

In the case of *light field* [57], where a subset of a densely sampled 2D array of images is used to interpolated desired views via IBR, the applications on interactive light field streaming had been extensively investigated [6, 58, 59]. Those works were motivated by the large size of the original image set, so that the transmission of the entire set before view navigation will create prohibitively large delay to viewers. To exploit the spatial correlations between neighboring views, [6] and [58] used switching P (SP)-frame and distributed source coding (DSC)-frame respectively to accommodate different coding paths. First, as what will be mentioned below, it can be noted that both SP-frame and DSC-frame are included in the abstraction of *merge frame* or *M-frame*. Further, since redundant P-frames are not considered in [6, 58], unlike the methods in IMVS, they provide no mechanism to systematically lower transmission cost by making use of extra storage, if available.

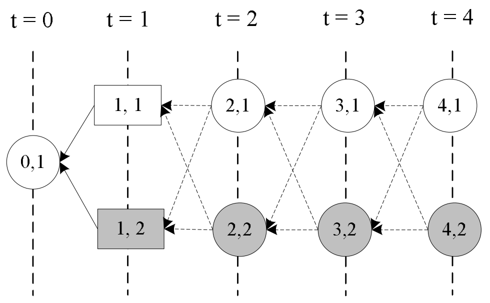
In [60, 61], it was assumed that each coding block of one image was encoded as INTRA, INTER or SKIP as in H.263 [62]. Then, for a requested INTER coding block to be correctly decoded, all blocks in its dependency path that were not already in the client buffer had to be transmitted, incurring a penalty on both transmission rate and decoding complexity. The notion of transmitting and decoding multiple blocks before displaying a single one block was referred as *rerouting* in the scenario of IMVS [63]. The results in [63] showed that rerouting was only able to provide marginal performance gain for IMVS when view-switch period Δ is reasonably large and when redundant P-frames are used.

In [4, 64], a two-layer approach was proposed for streaming MVV data, where coarse and fine quality layers of several views were grouped and pre-encoded. During actual streaming, a subset of views of low quality plus two views of high quality, carefully selected based on user's behavioral prediction, would then be sent to the client. All transmitted views were subsequently decoded, and the highest quality views that matched the user's at-the-moment desired views were displayed. While the intended application is similar to that of IMVS, IMVS is different in that it focuses on the optimal tradeoff among transmission rate, storage and view synthesis distortion using combinations of redundant P-frames and M-frames to construct frame structures.

Another similar work to IMVS is [65], which developed three separate frame structures to support three types of interactivity: view switching, frozen moment and view sweeping. While the authors recognized the importance of a "proper tradeoff among flexibility (interactivity), latency and bandwidth cost", no explicit optimization was performed to find the best tradeoffs of these quantities in one structure.



(a)



(b)

Figure 2.7: Two extreme examples of frame structure to enable view-switching for two views (white and grey) for $\Delta = 1$. I-, P- and M-frames are represented by circles, rectangles and diamonds, respectively. (a) P-frames only at switching points; (b) I-frames only at switching points.

2.3.2 Previous Works in IMVS

Frame Structure Optimization

Most of previous IMVS works focus on frame structure optimization [63, 66, 67]. The goal is to design frame structures at encoding time to trade off expected IMVS transmission rate and storage required to store the structure, without knowing the exact view trajectory a client will take at stream time. To see intuitively the tradeoff involved, consider the following two extreme examples. For simplicity, we assume that a client can request view-switch every $\Delta = 1$, but restrict allowable switches to only neighboring views, *i.e*, only clients observing views k's, $j-1 \leq k \leq j+1$, at time $i-1$ can switch to view j at time i. To encode frame $F_{i,j}$, since temporal playback is not interrupted, at time i one of the previous frames $F_{i-1,k}$'s (for at most three different views k) will be available at the decoder. Thus, one way to support view-switching is to differentially encode one P-frame $P_{i,j}$ for each possible decoded frame $F_{i-1,k}$ in the decoder buffer. We call this approach *redundant P-frames*—redundant in that an original picture $F^o_{i,j}$ is represented by multiple coded versions $P_{i,j}$'s. An example structure to allow view-switching between two views is shown in Fig. 2.7(a) where only P-frames $P_{i,j}$'s are encoded at view-switching points, each using a predictor $F_{i-1,k}$ of previous instant. As shown in Fig. 2.7(a), this approach will increase the number of decoding paths at each switching instant by a factor of two, resulting in a tree structure of size $O(2^N)$ if there are N switching instants between two I-frames. So although this approach would lead to a structure with minimum transmission cost (only bandwidth-efficient P-frames are used), the size of the coding structure is impractically large.

At the other extreme, one can construct a single coded version of the original picture $F^o_{i,j}$ for all possible decoder states, *i.e.*, M-frame, that can be correctly decoded no matter which $F_{i-1,k}$ is in the decoder buffer. Obviously, an independently coded I-frame would fit the M-frame reconstruction constraint. Fig. 2.7(b) shows an example of frame structure only using I-frames at switching points. Hence, a structure that uses M-frames exclusively at all view-switching points has high transmission rate but small storage cost (since each original picture is represented by a single coded version).

In our earlier IMVS works, we posed the IMVS problem as a combinatorial optimization in [5], proved its NP-hardness, and provided two heuristics-based algorithms to find good frame structures for IMVS. A more thorough and analytical treatment of the same problem was given in [63], which could generate a global optimized frame structure with exponential running time, using only I- and P-frames in the structure. In addition, several strategies to reduce the algorithm complexity were proposed for the practical applications. Preliminary results of using I-, P- and DSC frames [68] in an IMVS optimized structure were presented in [66]. [67] was a generalization of [66] where the optimization was posed as a search for the best combination of I-, P- and generalized M-frames.

Implementation of M-frame

In addition to I-frame, more generally, one can conceive other implementations of M-frame that exploit correlation between the set of possible predictors $F_{i-1,k}$'s and the target $F^o_{i,j}$ for coding gain. Example implementations of M-frames include SP-frame in H.264 [69] and different DSC-frames [68, 70][2]. In general, different implementations of M-frames induce different tradeoffs between storage cost and transmission rate [67]. However, any implementation of M-frame must necessarily have larger transmission rate than a P-frame, since by

[2]In the context of DSC, "predictor" frames are used as side information for decoding.

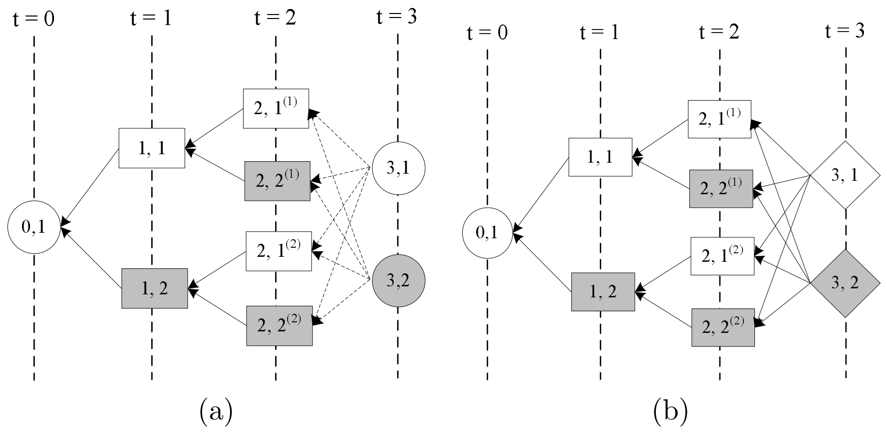


Figure 2.8: Example of frame structure (a) using I- and P-frames only; (b) using I-, P- and DSC-based M-frame for two views. I-, P- and DSC-frames are represented by circles, rectangles and diamonds, respectively.

definition, an M-frame must be encoded under the uncertainty of which one frame in the set of possible predictors $F_{i-1,k}$'s would be available at decoder buffer at stream time.

In the following, the method of using DSC-frames will be conceptually reviewed to achieve identical reconstruction from multiple predictor candidates. In a conventional closed-loop predictive system, the encoder calculates the prediction residue $Z = X - Y$, between source X and predictor Y, and transmits Z to the decoder. DSC approaches the same compression problem by viewing X as an input to a virtual channel with correlation noise Z, and Y as the output of the channel. Therefore, in order to recover X from Y, encoder will send parity information to the decoder. Naturally, this parity information computed entirely from X taking into account the statistics of Z, is independent of a specific Y being observed, and X can be exactly reconstructed as long as sufficient parity information has been transmitted.

This framework can be extended to address the mismatches in multiple decoding paths as follows. We consider N virtual channels, each corresponding to a predictor $F^{(h)}_{i-1,k}$ obtained from instant $i-1$ when we encode a picture $F^o_{i,j}$ as DSC-frame $M_{i,j}$. Each channel is characterized by the correlation noise $Z^{(h)}_k = F^o_{i,j} - F^{(h)}_{i-1,k}$. In order to recover $M_{i,j}$ exactly from any of these channels, the encoder needs to send an amount of parity sufficient for all the channels. That is, the frame size of $M_{i,j}$ depends on the worst correlation between the target and the predictors, i.e.,

$$B_{DSC} = \max_{k,h} H\left(F^{o}_{i,j}|F^{(h)}_{i-1,k}\right) \tag{2.13}$$

By doing so, the same $M_{i,j}$ can be reconstructed no matter which one of previous frames $F^{(h)}_{i-1,k}$'s will be presented at the decoder. Fig. 2.8(b) shows one example frame structure with DSC-frames for view-switching between two views.

Because a single DSC-frame $M_{i,j}$ would be stored in the server regardless of the number of decoding paths, DSC-frames would in general compare favorably to SP frames in terms of storage cost. The storage cost of $F^{0}_{i,j}$ using SP-frames, B_{SP}, depends on the number of replicas, since multiple SP-frames needs to be stored, each for one possible predictor $F^{(h)}_{i-1,k}$. More specifically,

$$B_{SP} = \sum_{k}\sum_{h} H\left(F^{o}_{i,j}|F^{(h)}_{i-1,k}\right) \tag{2.14}$$

DSC-frame is similar to I-frame in the sense that only one single I-frame is necessarily kept in the server for all the possible decoding paths, as shown in Fig. 2.8(a). However, it is obvious that DSC-frame outperforms I-frame in both storage and transmission rate, due to exploration of correlation between the target picture and predictor frames.

Although optimized frame structures of existing IMVS works can achieve optimal tradeoffs between transmission rate and storage, there are several shortcomings. First, the available views for a client to select are limited by the few camera-captured views pre-encoded at server, thus a view-switch could appear abrupt and unnatural to a viewer. Second, the proposed media interaction model that represents a typical client's view-switching behavior is assumed to be statistically independent in time, but it has been shown [65] that viewers exhibit temporal dependencies when switching views. Third, previous structure optimization assumes server-client communication takes place over idealized zero-delay network. In a realistic packet-switched network such as the Internet with non-negligible round trip time (RTT) delay, server's responding upon receipt of each client's requested view will mean each client's requested view-switch will suffer at least one RTT delay, hampering interactivity of the viewing experience. In Chapter 6, we propose three improvement technologies to solve the above problems in existing IMVS works.

Chapter 3

View Interpolation and Extrapolation-based MVC

3.1 Introduction

In this chapter, we first develop an improved RVI method and apply it to VSP-based MVC. As the proposed RVI method synthesizes a virtual view from a left view and a right view, VSP can only be applied to half of the views in RVI-based MVC. To solve this constraint, we then develop a RVE algorithm using two left views or two right views, so that VSP can be applied to the coding of all views after the first two. Experimental results show that these schemes can achieve better RD performance than the current JMVC standard as well as view interpolation/extrapolation-based MVC schemes without using rectification. Another important observation is that although the quality of the RVE-based view synthesis is generally lower than that of the RVI-based view synthesis, the average RD performance of RVE-based MVC across all views can outperform RVI-based MVC as the increase of the number of views, because RVE can be applied to more views than RVI.

The chapter is organized as the follows. In Sec. 3.2, we present the details of the proposed RVI method and its application in MVC. Sec. 3.3 extends the result to view extrapolation and applies it to MVC. Experimental results of the proposed RVI and RVE algorithms are presented in Sec. 3.4. MVC results using our methods are given in Sec. 3.5. The work in this chapter is summarized in Sec. 3.6.

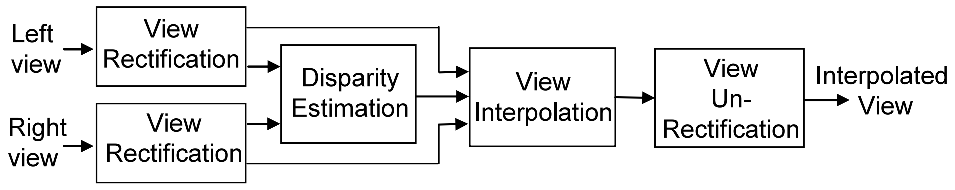


Figure 3.1: Block diagram of the proposed RVI algorithm.

3.2 RVI-based View Synthesis and Application in MVC

In this section, we propose an improved version of the RVI algorithms in [21, 49], and apply it to MVC. In particular, a more robust method is used to rectify the two reference views to reduce their vertical mismatches. A sub-pixel view interpolation is also developed to improve the accuracy of the integer-pixel interpolation in [49]. The proposed method does not require the knowledge of the camera parameters, since all the multiple view geometry can be estimated from the inputs.

3.2.1 The Proposed RVI algorithm

Fig. 3.1 shows the main steps in the proposed RVI algorithm, which are explained below.

Projective View Rectification

To rectify two non-parallel input views, we first estimate the fundamental matrix, which characterizes the epipolar geometry between the two views [20]. The matrix can be obtained without using any camera parameter.

Suppose a point $\mathbf{X}$ in the 3D space is projected to point $\mathbf{x}_l$ in one view. As mentioned in Sec. 2.1.1, its projection correspondence $\mathbf{x}_r$ in the other view lies on the line $\mathbf{F}\mathbf{x}_l$, where $\mathbf{F}$ is the 3×3 rank-2 fundamental matrix with seven degrees of freedom [20]. In addition, $\mathbf{x}_l$ and $\mathbf{x}_r$ satisfy (2.1) which is a linear function of the entries of $\mathbf{F}$. In our method, the 8-point method [20] is used to calculate $\mathbf{F}$ using the correspondences between two views. In addition, the correspondences are selected using corner detection and the random sample consensus (RANSAC) algorithms [20]. The implementation in [71] is modified to calculate $\mathbf{F}$ from the selected point correspondences. Note that other correspondence matching algorithms such as the scale invariant feature transform (SIFT) [72] can also be used to find the point correspondences.

Given $\mathbf{F}$, the epipoles of the two views can be obtained from the left and right null spaces of $\mathbf{F}$. After this, the rectification matrix of each view can be obtained as follows [21, 49]. First, the coordinate origin is translated to the image center via a transform

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where $\mathbf{c} = [c_x, c_y]^T$ is the image center. Suppose the epipole of a view is at $\mathbf{e} = [e_x, e_y, 1]^T$ after the translation. The next step is to rotate the image such that the epipole moves to the x-axis, *i.e.*, its homogeneous coordinate has the format $[v, 0, 1]^T$. The required rotation $\mathbf{R}$ is thus

$$\mathbf{R} = \begin{bmatrix} \alpha e_x & \alpha e_y & 0 \\ -\alpha e_y & \alpha e_x & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

where $\alpha = 1$ if $e_x \geq 0$ and $\alpha = -1$ otherwise.

Given the new epipole position $[v, 0, 1]^T$, the following transformation is applied to map the epipole to infinity.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/v & 0 & 1 \end{bmatrix} \tag{3.3}$$

As a result, the rectification matrix for a view is

$$\mathbf{H} = \mathbf{GRT} \tag{3.4}$$

In [49], the scheme in (3.4) is used to obtain the rectification matrices $\mathbf{H}_l$ and $\mathbf{H}_r$ for the left and right view, respectively, in order to create two parallel views. However, its performance relies mainly on the accuracy of the calculated epipoles. In [21], a more robust and accurate matching transform method is used, where the transformation $\mathbf{H}_l$ for the left view is still obtained by (3.4), but $\mathbf{H}_r$ for the right view is obtained by finding a matching transform that minimizes the mismatch of the two rectified views. However, this method needs to solve the camera matrices, which are not always available.

In this thesis, we optimize the rectification matrix $\mathbf{H}_r$ for the right view by minimizing the distances between a group of rectified corresponding points in the two views, *i.e.*,

$$\arg\min_{\mathbf{H}_r} \sum_i ||\mathbf{H}_l \mathbf{x}_{li} - \mathbf{H}_r \mathbf{x}_{ri}||^2 \tag{3.5}$$

where $\mathbf{x}_{li}$ and $\mathbf{x}_{ri}$ are some of the most accurate point correspondences in the two images, selected by the RANSAC algorithm. The Levenberg-Marquardt algorithm [20] is used to find the optimal solution of $\mathbf{H}_r$, with the initial value given by the method in (3.4). Our experimental results show that using (3.5) can reduce the average vertical mismatch of the two views by as much as 80% compared to the method in [49].

After the rectification, the resolutions of some regions in the rectified views are down-scaled, which can decrease the quality of the interpolated view. The down-scaled factor at a pixel position $(\tilde{x}, \tilde{y})$ in the rectified view is given by [49]

$$m(\tilde{x}, \tilde{y}) = \begin{vmatrix} \frac{\partial \tilde{x}}{\partial x} & \frac{\partial \tilde{x}}{\partial y} \\ \frac{\partial \tilde{y}}{\partial x} & \frac{\partial \tilde{y}}{\partial y} \end{vmatrix} \tag{3.6}$$

To compensate the loss of resolution, the pixel value at position $(\tilde{x}, \tilde{y})$ is extended to the unfilled pixels in a square region around $(\tilde{x}, \tilde{y})$ with a side length of $\sqrt{m(\tilde{x}, \tilde{y})}$.

Disparity Estimation

Since two parallel views are created after rectification, disparity estimation can be performed in 1D. In our method, the disparity estimation method in [26] is adopted, where a smoothness term is introduced into the cost function to favor solutions with small changes between neighbors. The energy cost function for a pixel at (x, y) is defined as:

$$E(x, y) = E_{data}(x, y) + E_{occ}(x, y) + E_{smooth}(x, y) \tag{3.7}$$

where E_{data} results from the intensity differences between corresponding pixels, E_{occ} imposes a penalty for making a pixel as occlusion, and the smooth term E_{smooth} ensures that neighboring pixels have similar disparities. Moreover, an uniqueness constraint is imposed in [26] to deal with occlusions, in which a pixel can correspond to at most one pixel in the other view, *i.e.*, a pixel can only be labeled as either a matching point that corresponds to one pixel, or an occluded point that corresponds to no pixel in the other view.

The disparity estimation in [49] is based on the method in [22], by adding an extra term in the cost function to improve the smoothness of the disparity map. However, our experimental results show that the improvement is not always satisfactory.

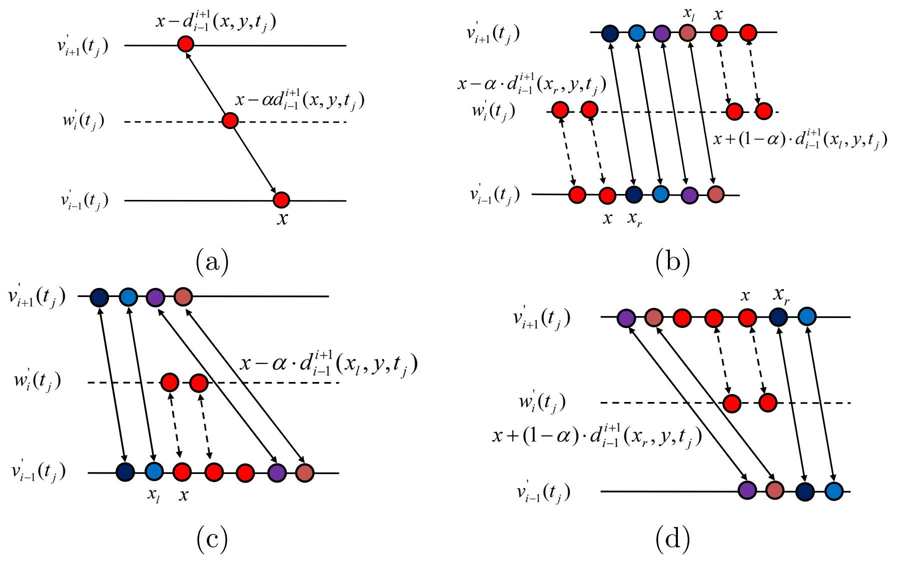

$v'_{i+1}(t_j)$
$w'_i(t_j)$
$v'_{i-1}(t_j)$
$x - d_{i-1}^{i+1}(x,y,t_j)$
$x - \alpha d_{i-1}^{i+1}(x,y,t_j)$
x
(a)
x_l
x
$x - \alpha \cdot d_{i-1}^{i+1}(x_r,y,t_j)$
$x + (1-\alpha) \cdot d_{i-1}^{i+1}(x_l,y,t_j)$
x_r
(b)
$x - \alpha \cdot d_{i-1}^{i+1}(x_l,y,t_j)$
(c)
$x + (1-\alpha) \cdot d_{i-1}^{i+1}(x_r,y,t_j)$
(d)

Figure 3.2: View interpolation. (a) pixels from both $v'_{i-1}(t_j)$ and $v'_{i+1}(t_j)$ are visible; (b) pixels whose correspondences are out of the boundary of $v'_{i-1}(t_j)$ or $v'_{i+1}(t_j)$; (c) Occlusion pixels in $v'_{i-1}(t_j)$; (d) Occlusion pixels in $v'_{i+1}(t_j)$.

Sub-pixel View Interpolation

View interpolation can be performed after disparity estimation. Although two neighboring views are available, there is no guarantee that every pixel in one view has its corresponding pixel in the other view, due to occlusion. Therefore, different cases need to be considered. In addition, in [22, 47], the interpolated coordinates of the pixels in the middle view are directly rounded to integer, which reduces the quality of the interpolated view and creates more occlusion regions. Although an occlusion padding algorithm is performed in [47] to improve the view interpolation, the quality of the synthesized images is still not satisfactory.

In this thesis, we propose a sub-pixel interpolation method, by distributing the contribution of each interpolated pixel with floating-point coordinates to the two nearest horizontal neighbors with integer coordinates.

Let $v_m(t_n)$ be the image of view m at time t_n, $v'_m(t_n)$ the rectified $v_m(t_n)$, and $w'_i(t_j)$ the generated virtual image for view i at time t_j. Also let $v'_m(x,y,t_n)$ and $w'_i(x,y,t_j)$ be the pixel value of $v'_m(t_n)$ and $w'_i(t_j)$ at position (x,y) respectively, and $d_n^m(x,y,t_j)$ the disparity of view m relative to view n at position (x,y) and time t_j. As in [22, 47], we interpolate the middle view by considering three cases.

If a pixel is visible in both views, as shown in Fig. 3.2(a), the corresponding pixel position in the intermediate view can be easily obtained by scaling the disparity value, and the pixel value of the intermediate pixel is interpolated from the correspondences in the left and right views by

$$w'_i\left(x-\alpha d_{i-1}^{i+1}(x,y,t_j),y,t_j\right)=(1-\alpha)v'_{i-1}(x,y,t_j)+\alpha v'_{i+1}\left(x-d_{i-1}^{i+1}(x,y,t_j),y,t_j\right) \quad (3.8)$$

where α is the ratio between the distance from view $i-1$ to view i and that from view $i-1$ to view $i+1$. Note that $x-\alpha d_{i-1}^{i+1}(x,y,t_j)$ generally has floating-point value.

For pixels whose corresponding pixels are out of the valid image area in the other view (Fig. 3.2(b)), we extend the disparity of the border pixel, and the pixel color is copied accordingly. That is, if the correspondence of $v'_{i-1}(x,y,t_j)$ is invalid in $v'_{i+1}(t_j)$, the interpolated pixel is taken as

$$w'_i\left(x-\alpha\cdot d_{i-1}^{i+1}(x_r,y,t_j),y,t_j\right)=v'_{i-1}(x,y,t_j) \quad (3.9)$$

Similarly, if the correspondence of $v'_{i+1}(x,y,t_j)$ is invalid in $v'_{i-1}(t_j)$, the interpolated pixel is

$$w'_i\left(x+(1-\alpha)\cdot d_{i-1}^{i+1}(x_l,y,t_j),y,t_j\right)=v'_{i+1}(x,y,t_j) \quad (3.10)$$

In (3.9) and (3.10), x_r and x_l are the horizontal axis of the first neighbor of $v'_{i-1}(x,y,t_j)$ and $v'_{i+1}(x,y,t_j)$ with valid point correspondence in the other view, as shown in Fig. 3.2(b).

Due to occlusions, some pixels are only seen in one view. Their disparity values are therefore unavailable. In our system, these pixels are detected by the disparity estimation method in [26]. As shown in Fig. 3.2(c-d), the occlusion areas in the left view (view $i-1$) are occluded by the objects at their right side, and the occlusion pixels in the right view (view $i+1$) are occluded by objects at their left side. Therefore, view interpolation can use the disparities of the neighboring background pixels. For view interpolation involving occlusion pixels in $v'_{i-1}(t_j)$, the disparity of the first available pixel to the left is used.

$$w'_i\left(x-\alpha\cdot d_{i-1}^{i+1}(x_l,y,t_j),y,t_j\right)=v'_{i-1}(x,y,t_j) \quad (3.11)$$

For view interpolation involving occlusion pixels in $v'_{i+1}(t_j)$, the disparity of the first available pixel to the right is used.

$$w'_i\left(x+(1-\alpha)\cdot d_{i-1}^{i+1}(x_r,y,t_j),y,t_j\right)=v'_{i+1}(x,y,t_j) \quad (3.12)$$

In (3.11) and (3.12), x_l and x_r are shown in Fig. 3.2(c-d).

Finally, to obtain the interpolated pixels at an integer location (x_0, y_0), we use the weighted combination of all pixels within unit distance from (x_0, y_0), *i.e.*,

$$w_i'(x_0, y_0, t_j) = \text{round}\left(\frac{1}{C(x, x_0)} \sum_{|x-x_0|<1} \gamma(x, x_0) \cdot w_i'(x, y_0, t_j)\right) \tag{3.13}$$

where $C(x, x_0) = \sum_{|x-x_0|<1} \gamma(x, x_0)$, $\gamma(x, x_0) = 1/(|x-x_0|+c_0)$, and c_0 is a constant to prevent overflow, and is set to be 0.1 in our implementation.

Note that if the distances from the left and right view to the target view are equal, the factor α in (3.8) to (3.12) will be 0.5, and the interpolated coordinates will be either integer or half-integer. In this case, the complexity of (3.13) can be simplified.

Projective Un-rectification

Similar to [20], the rectification algorithm above could generate non-rectangular interpolated images. Therefore, the last step of the RVI method is to back-project the intermediate view to the original coordinates at the same position. To do so, we first locate the positions of the four corners from the interpolated image $w_i'(t_j)$, denoted $\mathbf{x}_i', i = 1, \ldots, 4$. Our goal is to find an 3×3 un-rectification matrix $\mathbf{B}$ that minimizes the mapping error from these points to the four corners of the unrectified image $w_i(t_j)$, *i.e.*,

$$\arg\min_{\mathbf{B}} \sum_{i=1,\ldots,4} \|\mathbf{B}\mathbf{x}_i' - \mathbf{x}_i\|^2, \tag{3.14}$$

where $\mathbf{x}_i$ are homogeneous coordinates of the four corners in $w_i(t_j)$. The direct linear transform (DLT) method in [20] can be applied to convert (3.14) into a constrained least square problem

$$\arg\min_{\mathbf{b}} \|\mathbf{A}\mathbf{b}\|, \qquad \text{s.t.} \quad \|\mathbf{b}\| = 1, \tag{3.15}$$

where $\mathbf{b} = [\mathbf{b}_1\, \mathbf{b}_2\, \mathbf{b}_3]^T$ ($\mathbf{b}_i$ is the ith row of $\mathbf{B}$), *i.e.*, the vectorized version of $\mathbf{B}$. Matrix $\mathbf{A}$ is an 8×9 matrix, and each pair of corner correspondences contributes to two rows of $\mathbf{A}$. The optimal solution to (3.15) is the unit singular vector that corresponds to the smallest singular value of $\mathbf{A}$.

3.2.2 RVI-based MVC

In this part, we apply our RVI method to H.264-based MVC, by modifying the MDE-based JMVC software [73], which uses hierarchical B structure in the temporal direction and I-B-P

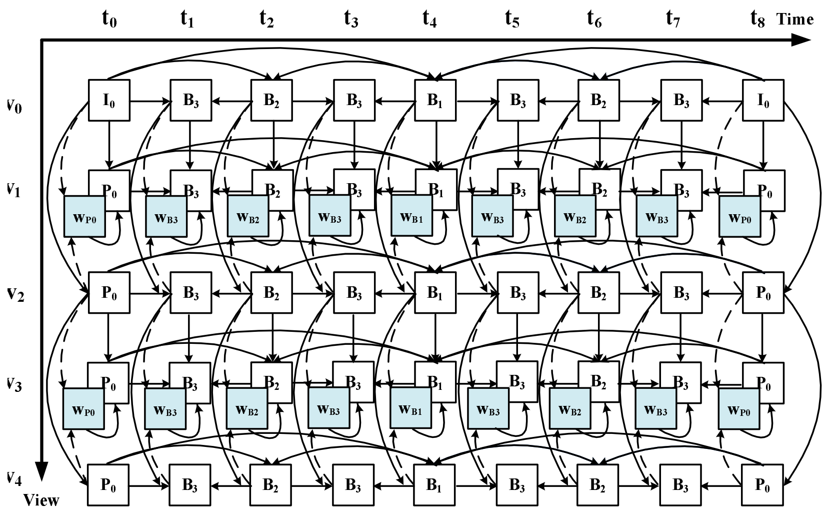

Time
View
t0 t1 t2 t3 t4 t5 t6 t7 t8
v0 v1 v2 v3 v4
I0 B3 B2 B3 B1 B3 B2 B3 I0
P0 B3 B2 B3 B1 B3 B2 B3 P0
wP0 wB3 wB2 wB3 wB1 wB3 wB2 wB3 wP0

Figure 3.3: The proposed RVI-based MVC schemes.

prediction structure in the inter-view direction.

The coding structure of our RVI-based MVC is illustrated in Fig. 3.3 for a system with five views and a group of pictures (GOP) size of 8. The coding of the even-indexed views is identical to the even-indexed views in the JMVC. That is, v_0 is coded using hierarchical B structure in the temporal direction. Other even-indexed views are coded by hierarchical B structure in the temporal direction, as well as disparity-compensated inter-view prediction using the previously reconstructed even-indexed view as reference.

For the odd-indexed views v_{2k+1}, in addition to temporal B references, two inter-view reference pictures are used in our method. The first is a synthesized frame $w_{2k+1}(t_j)$ generated by the proposed RVI method. The second is the left view. The encoder then uses R-D optimization to find the best coding mode for each block, by treating the synthesized view as an additional reference picture. The synthesized views can be generated at the decoder using the reconstructed reference views, thus no additional bits need to be sent to the decoder.

It should be mentioned that the frames of v_{2k+1} are coded as B pictures in the inter-view direction in the JMVC, using the left view and the right view as references. Therefore our scheme has the same number of inter-view references as the JMVC. However, since the quality of our view interpolation-based prediction is usually better than that of the disparity compensation, the proposed RVI-based MVC scheme can achieve a better coding efficiency than JMVC, as shown in Sec. 3.5.

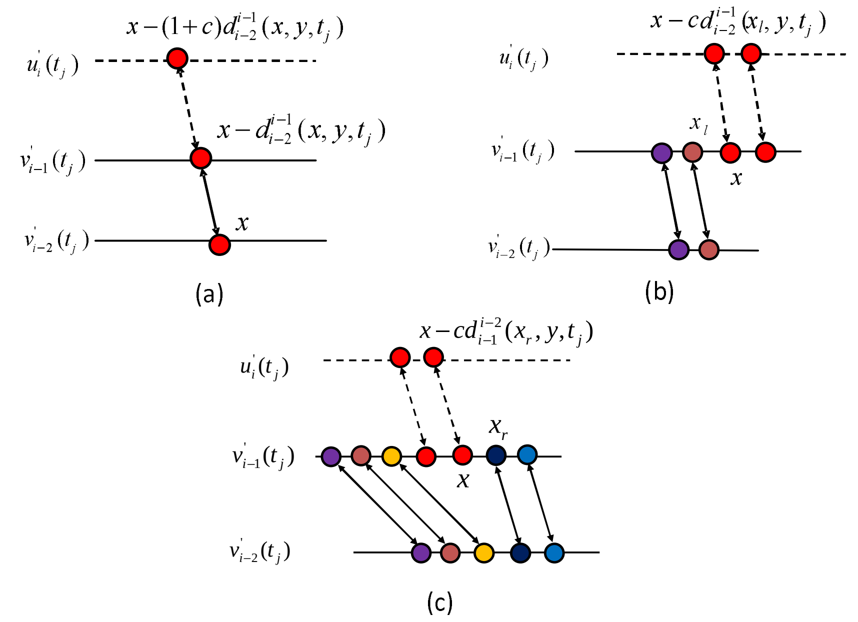


Figure 3.4: View extrapolation. (a) pixels from both $v'_{i-2}(t_j)$ and $v'_{i-1}(t_j)$ are visible; (b) pixels from $v'_{i-2}(t_j)$ is out of the boundary; (c) only pixel from $v'_{i-1}(t_j)$ is available.

3.3 RVE-based View Synthesis and Application in MVC

View interpolation requires a left view and a right view. To apply it to MVC, VSP can only be applied to half views in order to get satisfactory performance. In this section we generalize the RVI method to get a RVE algorithm using two left views or two right views. We then apply the RVE method to MVC to encode all views after the first two views.

3.3.1 The Proposed RVE Algorithm

In this thesis, we assume that the view extrapolation algorithm uses two left views to synthesize a right view. Similar to the view interpolation algorithm in Sec. 3.2, the extrapolation algorithm first performs projective rectification and disparity estimation to the two left views. After that, instead of interpolating the disparity to find the corresponding pixel locations in the middle view, the algorithm extrapolates the disparity and estimates the pixel locations in the right view. The final step of un-rectification is still similar to the view interpolation method. The disparity extrapolation is described below, since it is the only different step.

Using the same notations as in Sec. 3.2.1, two frames from the two previous views,

$v_{i-2}(t_j)$ and $v_{i-1}(t_j)$, are used to extrapolate a frame for view i. Let $v'_{i-2}(t_j)$, $v'_{i-1}(t_j)$ and $u'_i(t_j)$ be the rectified frames of $v_{i-2}(t_j)$, $v_{i-1}(t_j)$ and the synthesized view i at t_j, respectively.

If the horizontal camera distance between $u'_i(t_j)$ and $v'_{i-1}(t_j)$ is c times of that between $v'_{i-2}(t_j)$ and $v'_{i-1}(t_j)$, we assume their disparities have the same scaling factor, *i.e.*,

$$d^{i}_{i-1}(x, y, t_j) = c \cdot d^{i-1}_{i-2}(x, y, t_j). \tag{3.16}$$

The following three cases need to be handled.

If a pixel is visible in both $v'_{i-2}(t_j)$ and $v'_{i-1}(t_j)$, as shown in Fig. 3.4(a), we extrapolate their disparity, and the synthesized pixel in $u'_i(t_j)$ is the average of the pixel pair. That is,

$$u'_i\left(x - (1+c)d^{i-1}_{i-2}(x, y, t_j),\, y, t_j\right) = \frac{1}{2}\left[v'_{i-2}(x, y, t_j) + v'_{i-1}(x - d^{i-1}_{i-2}(x, y, t_j),\, y, t_j)\right] \tag{3.17}$$

For pixels whose correspondences are out of the valid region of $v'_{i-2}(t_j)$, as shown in Fig. 3.4(b), we scale the disparity of the first left pixel $(x_l,\, y)$ with valid point correspondence,

$$u'_i\left(x - c \cdot d^{i-2}_{i-1}(x_l, y, t_j), y, t_j\right) = v'_{i-1}(x, y, t_j) \tag{3.18}$$

If a pixel at (x, y) is only visible in $v'_{i-1}(t_j)$, as shown in Fig. 3.4(c), it is also assumed to be visible in the extrapolated view, and the first available disparity to the right of this pixel, with coordinate (x_r, y), is used as the disparity of this pixel. This disparity is then scaled by c to find the position of the corresponding pixel in the target view, *i.e.*,

$$u'_i\left(x - c \cdot d^{i-2}_{i-1}(x_r, y, t_j),\, y, t_j\right) = v'_{i-1}(x, y, t_j) \tag{3.19}$$

If a pixel is only visible in $v'_{i-2}(t_j)$ but not in $v'_{i-1}(t_j)$, it is assumed to be invisible in the extrapolated view as well. Therefore no operation is needed.

The locations of the extrapolated pixels are not integer in general. To find the pixels at integer locations, the sub-pixel method in (3.13) can be used. However, if the factor c in (3.16) to (3.19) is integer, there is no need for sub-pixel view extrapolation, as all coordinates involved are integer.

After the extrapolation operations above, there could still be some holes in the extrapolated view, because some parts of $v_i(t_j)$ do not appear in $v_{i-2}(t_j)$ and $v_{i-1}(t_j)$. Conventional occlusion handling techniques, such as extrapolating neighboring background pixels, are often ineffective in this case because view extrapolation tends to create larger occluded regions

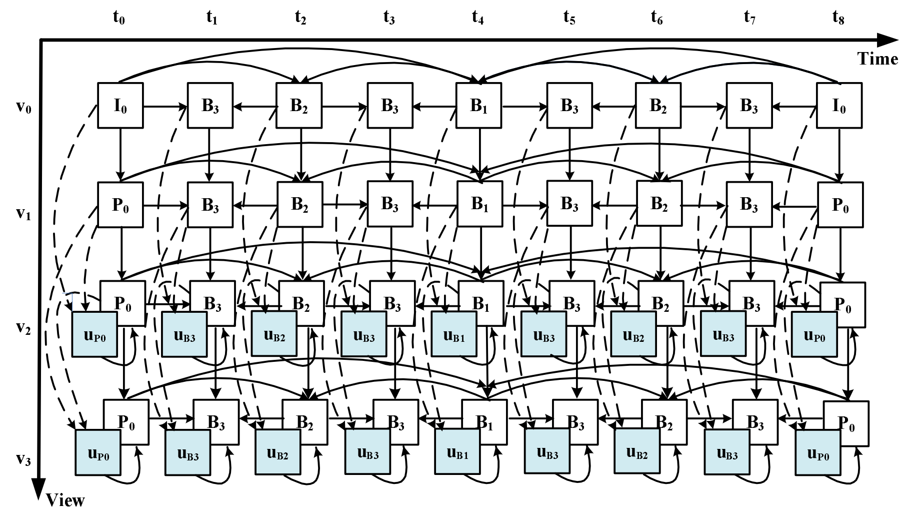

t0 t1 t2 t3 t4 t5 t6 t7 t8
Time
View
v0 v1 v2 v3
I0 B3 B2 B3 B1 B3 B2 B3 I0
P0 B3 B2 B3 B1 B3 B2 B3 P0
uP0 uB3 uB2 uB3 uB1 uB3 uB2 uB3 uP0

Figure 3.5: The proposed RVE-based MVC schemes.

than view interpolation. Therefore, we introduce a new technique for handling occlusions, where the occluded pixel in $u_i'(x,y,t_j)$ is interpolated from pixels in the two previously decoded frames in the temporal direction by

$$u_i'(x,y,t_j) = \frac{1}{2}\left[v_i'(x,y,t_{j-2}) + v_i'(x,y,t_{j-1})\right]. \tag{3.20}$$

In addition, the synthesis bias correction method in [12] is performed after view un-rectification to reduce the biases originated from the view extrapolation as well as the illumination variations between camera views.

3.3.2 RVE-based MVC

The proposed view extrapolation method can be applied to MVC to encode all views after the first two views. The coding structure is shown in Fig. 3.5 for a system with four views and GOP size of 8. Similar to Fig. 3.3, the temporal prediction scheme is based on the hierarchical B structure. The anchor frames in the first two views are either intra-coded or inter-coded using the previous view. After the first two views, each non-anchor frame in the subsequent views has four references: two references from the temporal prediction, one from the decoded previous view, and one from view extrapolation.

We will show later that although the quality of view extrapolation is often lower than that of view interpolation, the degradation could be compensated by the improved coding efficiency when view extrapolation is applied to more views.

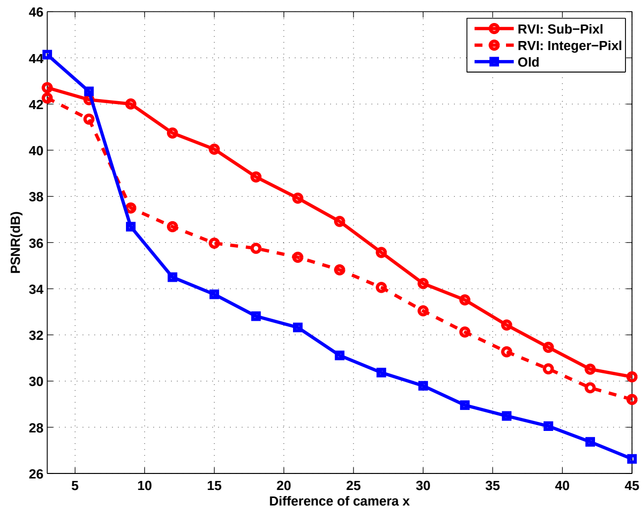


Figure 3.6: Performance comparison of various view interpolation algorithms, for `Xmas`.

3.4 Experiment Results of RVI and RVE-based View Synthesis

We first compare our view synthesis algorithms with the method in [48], using the test sequence `Xmas`, where 101 views $I_m, m = 0, 1, \ldots, 100$ of a fixed scene are taken by moving a camera to different positions along a straight line. The maximum disparity between the two boundary cameras is 70 pixels. Since the cameras are parallel, the rectification and un-rectification steps in our method are not needed. The performance gap is caused by the different disparity estimation and view synthesis algorithms of the two methods.

In Fig. 3.6, we calculate the peak-signal-to-noise ratio (PSNR) between the original and the interpolated images of a specific view, using a left reference view and a right reference view with index difference $2x$. For each given x, the PSNRs of the interpolated results for view 45 to view 55 are measured and the average is used as the corresponding PSNR, as in [48]. The result of the method in [48] is denoted as "Old". The results of both the integer-pixel and sub-pixel versions of the proposed RVI method are plotted.

It can be observed that our method significantly outperforms [48] when $x > 6$, or when the disparity between the left and right views is greater than 8.4 pixels, which is the case in most systems. The gain is up to about 6 dB at $x = 15$. In addition, the quality of the sub-pixel view interpolation is better than that of integer-pixel interpolation in all cases with a maximum improvement of 4.6 dB at $x = 9$. Therefore our method can be useful for

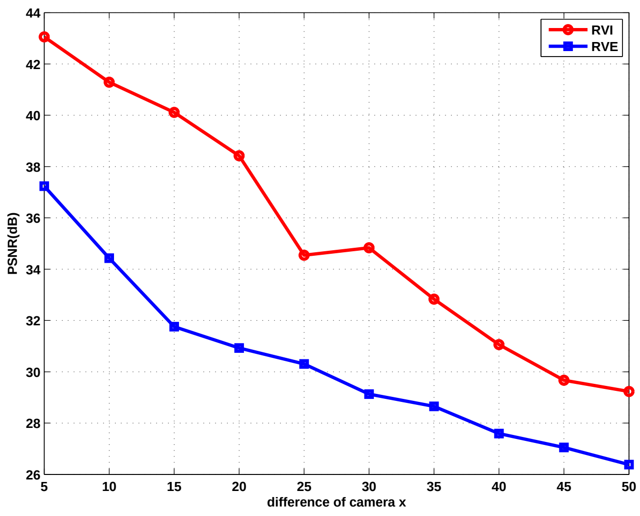


Figure 3.7: Performance comparison between RVI and RVE methods, for `Xmas`.

applications such as free viewpoint videos.

Fig. 3.7 compare the performances of our view interpolation and extrapolation methods. The view interpolation result is obtained by using view 50 as the target view, and view $50 - x$ and view $50 + x$ as references. The view extrapolation result is obtained by treating view 100 as the target and view $100 - x$ and view $100 - 2x$ as the references. As shown in the figure, the quality of view extrapolation is much lower than the view interpolation, with gap ranging from 2.6 to 8.3 dB. However, when we increase the distance between two reference cameras, the gain of view interpolation over view extrapolation begins to drop, as disparity estimation error would dominate the view synthesis distortion for a large camera baseline.

Fig. 3.8 shows the performances of our methods for non-parallel cameras, using view 2 of the `Breakdancers` sequence, which has a 1D arc camera setup. To synthesize view 2, the original view 0 and view 1 are used as references for view extrapolation, whereas view 1 and view 3 are used for view interpolation. Four cases of our methods are tested, namely, RVI, RVE, direct view interpolation (DVI), and direct view extrapolation (DVE), where DVI and DVE are the same as RVI and RVE except that the rectification step is disabled.

It can be seen that RVI outperforms RVE for all the frames of the synthesized view, with an average improvement of 2.8 dB. When the rectification is disabled, the result of DVI and DVE is about 5.8 dB and 4.1 lower than RVI and RVE respectively, and the gap between DVI and DVE is smaller than that between RVI and RVE. This shows the importance of

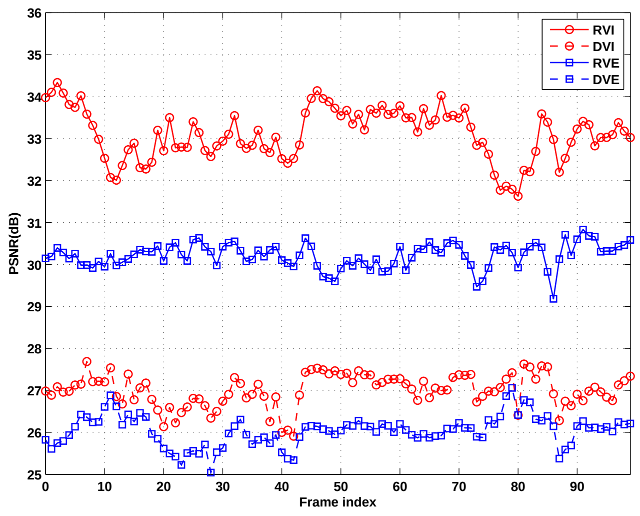


Figure 3.8: Performance comparison of various view interpolation and view extrapolation methods, for `Breakdancers`.

rectification for non-parallel cameras.

Fig. 3.9 shows an interpolated image of `Breakdancers` sequence before and after projective un-rectification. It can be seen that the non-rectangular interpolated frame can be corrected after un-rectification.

3.5 Experimental Results of MVC

We next compare the performances of the proposed RVI and RVE-based MVC schemes with those of the JMVC [73] and the simulcast method. Six MVV data sets, `Breakdancers`, `Ballet` [74], `Uli` [75], `Aquarium`, `Rena`, and `Akko & Kayo` [76] are tested. The first four sequences have 1D arc camera setup, whereas the last two use parallel cameras, for which the RVI and RVE-based MVC reduces to DVI and DVE-based MVC. Our methods are implemented based on the JMVC 3.0 software with 8-frame GOP, 96-pixel motion search window, and CABAC entropy encoding. The RD optimization and loop filter are enabled. The rate control is turned off.

We first show in Table 3.1 the average coding gains of the proposed MVC schemes and the JMVC over the simulcast method, when only view 2 of each sequence is encoded. The coding gains are obtained by averaging the PSNR improvements over the simulcast at all tested bit rates.

To achieve a fair comparison, we first independently encode view 0, view 1 and view

Table 3.1: Average single view coding gain over simulcast

Sequence	Average Coding Gains (dB)				
	JMVC	RVI	DVI	RVE	DVE
Breakdancers	1.50	1.99	1.29	1.66	1.26
Aquarium	1.65	2.09	1.40	1.95	1.32
Ballet	1.33	1.58	0.99	1.48	0.96
Uli	0.08	0.16	0.05	0.15	0.03
Rena	2.76	3.63	3.63	3.24	3.24
Akko&Kayo	3.04	3.68	3.68	3.31	3.31

3. To encode view 2 with the JMVC method, we use the decoded view 1 and view 3 as the inter-view references. To encode view 2 with the RVI or RVE-based MVCs, we use the decoded view 1 and view 3 to interpolate view 2, and use the decoded view 0 and view 1 to extrapolate view 2, respectively. The synthesized view is then used as another inter-view reference for view 2 in our methods, in addition to the decoded view 1. Therefore the JMVC and the RVI and RVE-based MVCs have the same number of reference frames in this single-view coding test. The only difference of our methods from JMVC is that one of the inter-view reference of JMVC is replaced by the synthesized view.

As shown in Table 3.1, both RVI and RVE-based MVCs outperform MDE-based JMVC. Moreover, RVI-based MVC is better than RVE-based MVC for all sequences. When the rectification step is disabled in arc camera setup, both DVI and DVE-based MVCs perform worse than JMVC, highlighting the importance of rectification for non-parallel cameras.

Table 3.1 also shows that the gain of the rectification-based approach is less for sequences Ballet and Uli. The reason for the Ballet case is that the objects in the sequence are quite close to the camera, leading to large occluded areas. The lack of effectiveness for Uli sequence is because it has larger camera distance and more color distortion between cameras.

Fig. 3.10 depicts some RD curves for view 2. The RVI-based MVC can be up to 1 dB better than RVE-based MVC at low rates. When the rate increases, the gain of our method over JMVC starts to diminish, because the prediction residual signal produces most of the output bits. The same phenomenon also exists in other MVC algorithms [42, 48, 77, 78].

We next encode all views of a sequence with JMVC and our methods, using the structure of JMVC, Fig. 3.3 and Fig. 3.5, respectively. We then calculate the average PSNR gains of these schemes over the simulcast when different numbers of views are used in computing the average PSNR. The results are reported in Fig. 3.11, which provides a better performance measure than the single-view coding in Table 3.1 and Fig. 3.10. It also reveals the efficiencies of different methods for each view.

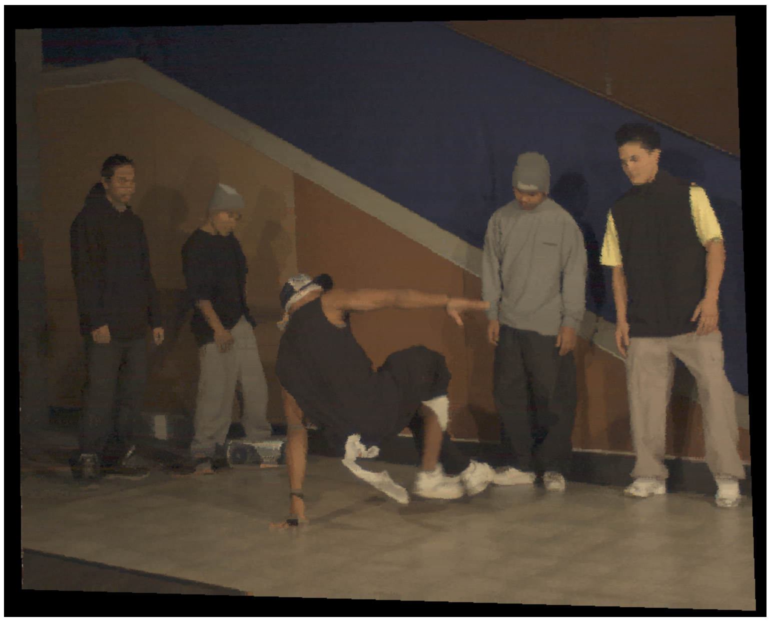
It can be seen in Fig. 3.11 that all MVC schemes have the same performance for view 0, which is always coded by the simulcast method. When two views are used to calculate the average coding gain, view interpolation-based MVCs achieve better results than extrapolation-based MVCs, because view 1 in Fig. 3.3 has one more reference than view 1 in Fig. 3.5. However, when more than two views are compared, view extrapolation-based MVCs start to outperform interpolation-based MVCs, because RVE/DVE is applied to all views after the first two, whereas RVI/DVI can only be applied to half of the views. In addition, the average coding gains of JMVC and RVI-based MVC over simulcast increase for each odd-indexed view while decrease for each even-indexed view, because odd-indexed views in Fig. 2.6 and Fig. 3.3 have one more reference than even-indexed views for inter-view prediction.

Fig. 3.11 also shows that the rectification generates additional coding gains to both view interpolation and extrapolation methods for non-parallel camera setup. These results will be verified by the theoretical analyses in Chapter 4.

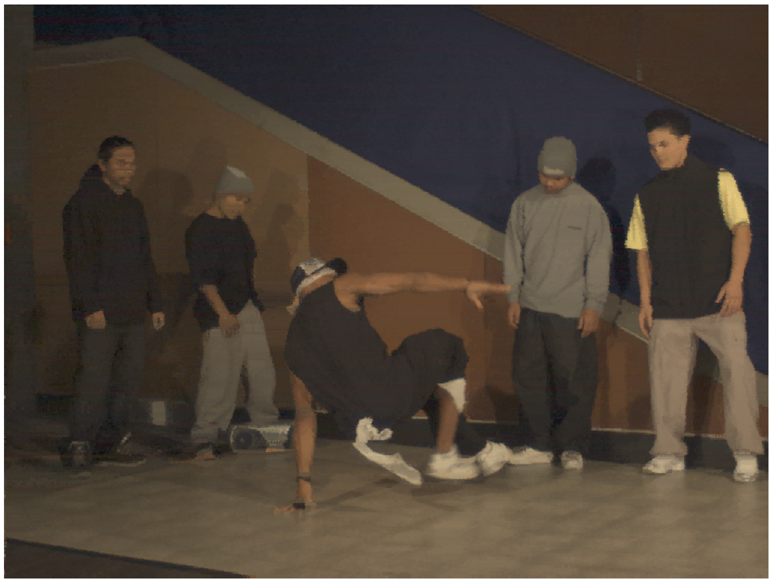
Although the proposed algorithms improve the performance of view synthesis and MVC, they increase the complexity of the system. Experimental results show that the view rectification, disparity estimation, sub-pixel view interpolation and view un-rectification in Sec. 3.2.1 increase the encoding complexity by about 30%, 200%, 10% and 10%, respectively. Since disparity estimation is the most expensive operation, the complexity can be reduced by using faster disparity estimation methods. The scheme can also benefit from the rapid development of hardware design such as GPU computing. For example, a GPU-based real-time view synthesis system is reported in [79], and an OpenGL-based hardware acceleration is used in [49] to achieve real-time view interpolation.

3.6 Summary

In this chapter, we develop projective rectification-based view interpolation and extrapolation methods and apply them to MVC. Experimental results show that the proposed view synthesis schemes achieve better performance than existing methods, and lead to improved RD performance than the current JMVC standard. An important observation is that although the quality of the extrapolated views is generally lower than that of the interpolated views, the average RD performance of view extrapolation-based MVC across all views can outperform the view interpolation-based MVC as the increase of the number of views, because view extrapolation can be applied to more views than view interpolation.

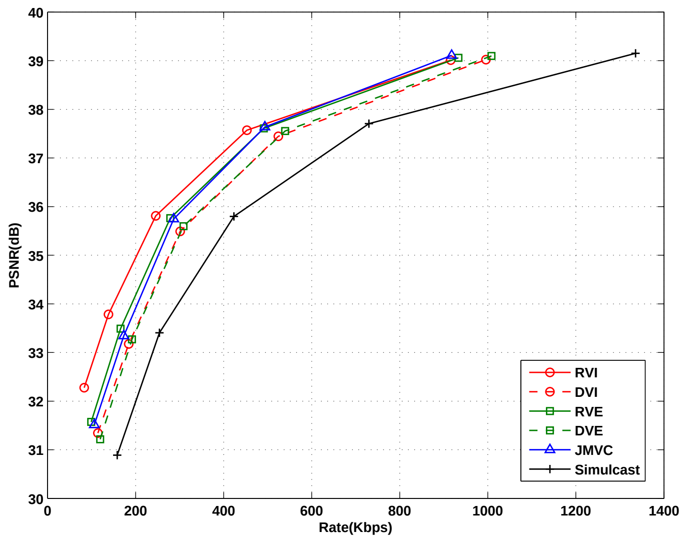


(a)

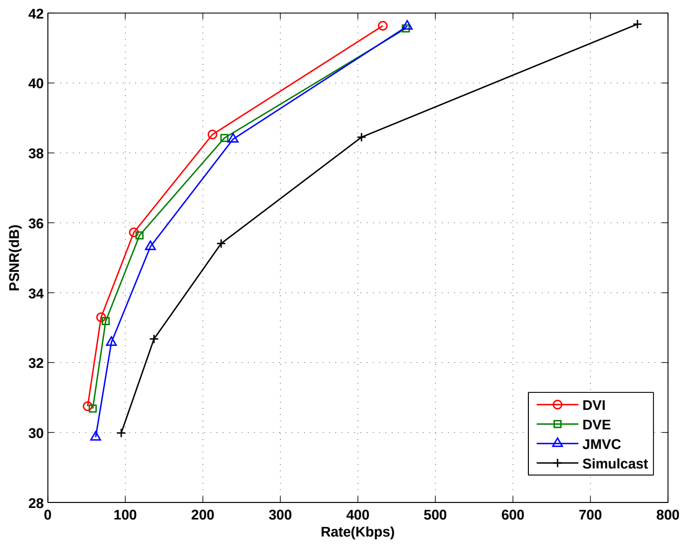


(b)

Figure 3.9: Examples of interpolated images of `Breakdancers`. (a) Before un-rectification. (b) After un-rectification (PSNR: 33.96 dB).

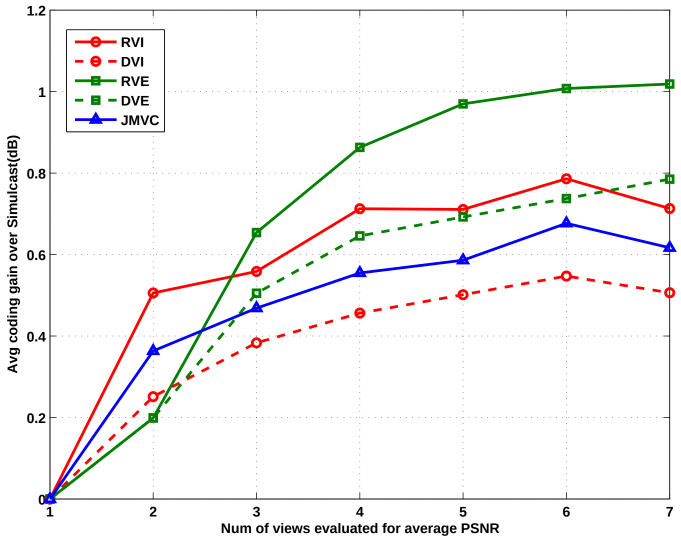


(a)

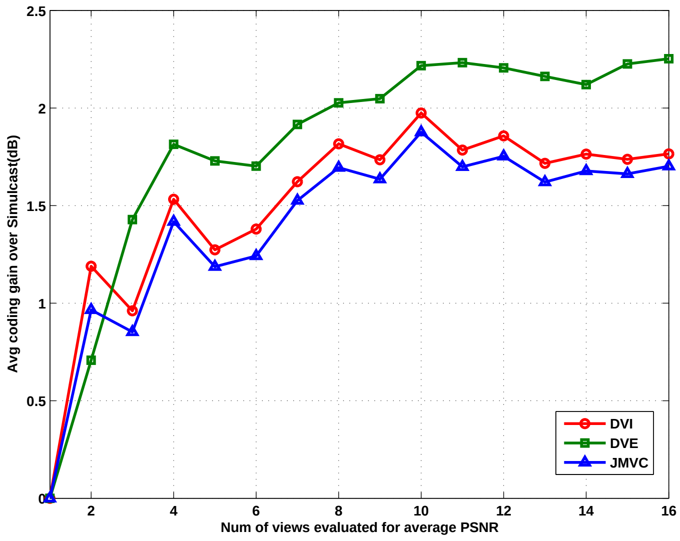


(b)

Figure 3.10: RD curves of encoding one view. (a) `Breakdancers`. (b) `Rena`.



(a)



(b)

Figure 3.11: Average coding gain across different views over the simulcast method. (a) `Breakdancers` at 550 kbps. (b) `Rena` at 200 kbps.

Chapter 4

Theoretical RD Analysis of MVC

4.1 Introduction

In Chapter 3, we develop RVI and RVE-based view synthesis algorithms and show experimentally the coding gain of the proposed methods over the existing DVI, DVE and MDE-based view prediction methods when applying them to MVC. In [55], a theoretical model was developed for DVI-based view synthesis, where the effect of disparity estimation error on the quality of the view interpolation was analyzed and a pre-filter method was proposed to improve the view interpolation quality. However, the model in [55] assumed all cameras were aligned in a straight line and had parallel views.

In this chapter, we propose a more accurate geometrical model than that in [55], which allows more flexible camera setup in terms of positions and directions. Furthermore, this model enables us to derive the theoretical performance gain of projective rectification on the quality of the interpolated or extrapolated view when unaligned cameras are used. To the best of our knowledge, this is the first attempt to quantify the improvement of the rectification step to view synthesis. On the other hand, we also develop an improved RD model to analyze the performances of different practical MVC schemes, such as MDE-based JMVC, RVI/RVE-based MVC and DVI/DVE-based MVC. Our model is a generalization of that in [35], with the consideration of modeling different VSP methods. Simulation results of this model verify the experimental results of various MVC schemes discussed in Chapter 3.

The chapter is organized as follows. In Sec. 4.2, we develop a geometric model to analyze the impacts of the camera orientation, camera distance, and disparity error on the quality of RVI and RVE. In Sec. 4.3, we develop an improved RD model for the comparison of

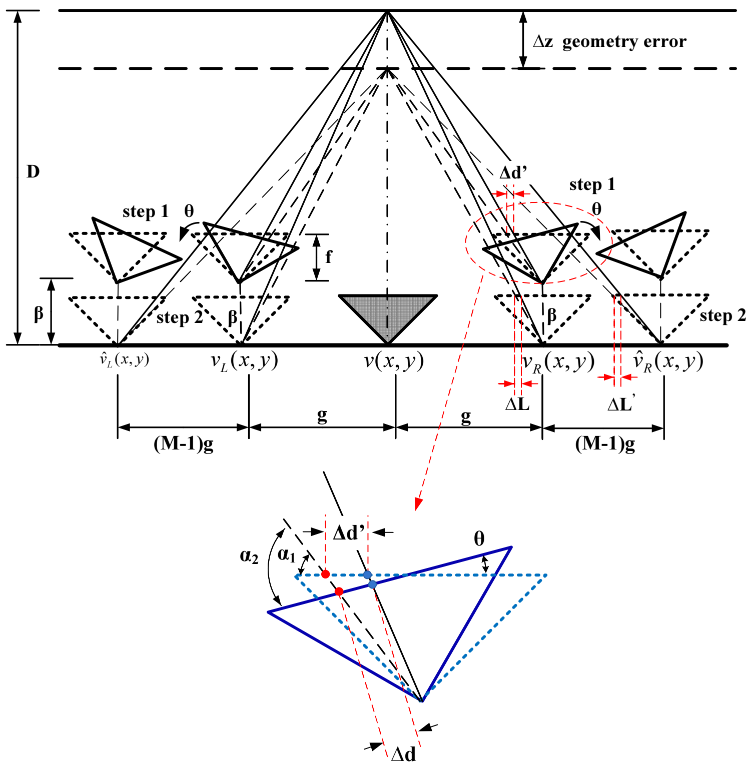


Figure 4.1: Model of rectification-based view synthesis.

practical MVC schemes. Simulation results are given in Sec. 4.4. The work in this chapter is summarized in Sec. 4.5.

4.2 Geometric Models of RVI and RVE-based View Synthesis

In this section, we first develop a geometric model to theoretically analyze the performance of RVI-based view synthesis, especially derive the performance gain of projective rectification on previous DVI method. Then, the same model is extended to study RVE-based view synthesis. The models discussed in this section will be used in Sec. 4.3 to study the RD performance of RVI and RVE-based MVC.

4.2.1 Geometric Model of RVI

Fig. 4.1 shows the proposed geometric model for RVI, which is a generalization of that in [55]. In Fig. 4.1, v_L and v_R are the left and right views, whose orientations are not parallel,

v is the new view to be synthesized, and $\hat{v}_L$ and $\hat{v}_R$ are another possible pair of left and right views.

As in [53, 55], the scene is modeled as a planar surface, and a geometry error of Δz is assumed in estimating the position of the planar surface. As shown in 4.1, the geometry error will lead to a disparity error Δd in the left and right views, and in general $\Delta z \neq \Delta d$. This model is more accurate than that in [55], where no distinction is made between the geometry error of the scene and the disparity error of the images, *i.e.*, it simply assumes $\Delta z = \Delta d$.

We next find the relationship between the average disparity errors before and after the rectification, which shows that the rectification can reduce the disparity error. For simplicity, we assume v is parallel to the planar surface. The un-rectification step is thus not needed. We also assume that the left and right views are symmetric with respect to the middle view.

With these assumptions, the rectification reduces to two steps. First, the left and right views are rotated by angle θ so that they are parallel to the middle view. This step captures the essence of the transform in (3.3). Secondly, the rotated views are shifted by β to be in the same line as the middle view.

The enlarged portion in Fig. 4.1 illustrates the details of the first step, where $\Delta d'$ is the disparity error after rotation. We assume that the scene surface is far away from the camera centers. Thus, using the geometric relationship in Fig. 4.1, the disparity errors Δd and $\Delta d'$ caused by the geometry error of a point on the scene surface is approximately related by

$$\frac{\Delta d'}{\Delta d} \approx \frac{sin\alpha_2}{sin\alpha_1} = \frac{sin(\alpha_1+\theta)}{sin\alpha_1} = cos\theta + sin\theta \cdot \frac{cos\alpha_1}{sin\alpha_1} \tag{4.1}$$

where the angles α_1 and α_2 are defined in Fig. 4.1.

The result above depends on the pixel location. To simplify the model, we use the average value of $\Delta d'/\Delta d$ of all pixels, denoted by $\overline{\Delta d'}/\overline{\Delta d}$. Assume α_1 is uniformly distributed in $[\frac{\pi-\alpha_0}{2}, \frac{\pi+\alpha_0}{2}]$, where α_0 is the field of view, we have

$$\frac{\overline{\Delta d'}}{\overline{\Delta d}} \approx \frac{1}{\alpha_0}\int_{\frac{\pi-\alpha_0}{2}}^{\frac{\pi+\alpha_0}{2}} \left(cos\theta + sin\theta \cdot \frac{cos\alpha_1}{sin\alpha_1}\right) d\alpha_1 = cos\theta \tag{4.2}$$

After shifting the left and right views to the same line as the middle view, the disparity error changes from $\Delta d'$ to ΔL. Let D be the distance from the planar surface to the camera center of the middle view. If $D >> \beta$, which is usually the case, we have $\Delta L \approx \Delta d'$. To see this, let f be the camera focal length, and g the horizontal distance between the left or right view and the middle view. We can get from Fig. 4.1 that

$$\Delta L = g \cdot \left(\frac{D-f}{D} - \frac{D-f-\Delta z}{D-\Delta z} \right) \tag{4.3}$$

$$\Delta d' = g \cdot \left(\frac{D-f-\beta}{D-\beta} - \frac{D-f-\Delta z-\beta}{D-\Delta z-\beta} \right) \tag{4.4}$$

Hence the impact of the shift is negligible when $D >> \beta$.

As a result, the average disparity error $\overline{\Delta L}$ after the rectification is related to the original average disparity error by

$$\overline{\Delta L} \approx \overline{\Delta d} \cdot cos\theta \tag{4.5}$$

Therefore, the rectification can reduce the disparity error by a factor of $cos\theta$ on average.

Although the derivation of this model is based on the simple planar surface model and the average step in (4.2), we will show later that the model agrees reasonably well with the experimental results in both view synthesis and MVC.

In a MVC system, instead of using the nearest neighboring views, sometimes it is necessary to use other cameras to interpolate a middle view, as in the hierarchical B structure. In Fig. 4.1, this can be studied by using a pair of left and right views $\hat{v}_L$ and $\hat{v}_R$ with distance Mg from the middle view as the references for interpolation.

To study the impact of the camera distance on the disparity error, note from (4.3) that the rectified disparity error is proportional to the distance between two cameras; hence if we denote $\Delta L'$ as the rectified disparity error for $\hat{v}_L$, we have

$$\Delta L' = Mg \left(\frac{D-f}{D} - \frac{D-f-\Delta z}{D-\Delta z} \right) = M\Delta L \tag{4.6}$$

Its average is thus $\overline{\Delta L'} \approx M\overline{\Delta d}cos\theta$, which is M times of that of the nearest camera. This is approximately true even if the cameras at distance Mg have different angles from the nearest cameras. It also simplifies the subsequent derivation, since we only need to consider the angle θ of the nearest cameras.

Let $2d$ be the true disparity of the planar surface between rectified left view $\hat{v}'_L$ and right view $\hat{v}'_R$. When the disparity error above is considered, the disparity becomes $2d + 2\overline{\Delta L'}$. The interpolated middle view is the average of the disparity-compensated rectified left and right views, thus

$$\begin{aligned} w(x,y) &= 1/2 \left\{ \hat{v}'_L \left(x + d + \overline{\Delta L'}, y \right) + \hat{v}'_R \left(x - d - \overline{\Delta L'}, y \right) \right\} \\ &= 1/2 \left\{ v \left(x + \overline{\Delta L'}, y \right) + v \left(x - \overline{\Delta L'}, y \right) \right\} \\ &= 1/2 \left\{ v \left(x + Mcos\theta\overline{\Delta d}, y \right) + v \left(x - Mcos\theta\overline{\Delta d}, y \right) \right\} \end{aligned}$$

The fourier transform (FT) of the interpolated view is thus

$$W(\omega_x, \omega_y) = H_1(\omega_x) \cdot V(\omega_x, \omega_y) \tag{4.7}$$

where

$$H_1(\omega_x) = cos\left(Mcos\theta\,\overline{\Delta d}\,\omega_x\right) \tag{4.8}$$

(4.7) is a powerful formula that captures the impacts of the disparity error, camera orientation, and camera distance on the quality of the interpolated view. When $M = 1$ and $\theta = 0$, this expression reduces to that in [55] for parallel views. Our model thus can be used to study view interpolation with flexible camera orientations.

4.2.2 Geometric Model of RVE

The setup in Fig. 4.1 can be modified to derive a model for view extrapolation. Without loss of generality, we use two left views, such as $\hat{v}_L$ and v_L, to extrapolate the target right view v. If the true disparity of the rectified left view v'_L with respect to the target view is d, the disparity of $\hat{v}'_L$ will be Md. Using the disparity errors in (4.5) and (4.6), the extrapolated view can be obtained by the following average of the disparity-compensated rectified v_L and $\hat{v}_L$.

$$\begin{aligned} u(x,y) &= 1/2\left\{v'_L\left(x + d + \overline{\Delta L}, y\right) + \hat{v}'_L\left(x + Md + \overline{\Delta L'}, y\right)\right\} \\ &= 1/2\left\{v\left(x + \overline{\Delta L}, y\right) + v\left(x + \overline{\Delta L'}, y\right)\right\} \\ &= 1/2\left\{v\left(x + cos\theta\overline{\Delta d}, y\right) + v\left(x + Mcos\theta\overline{\Delta d}, y\right)\right\} \end{aligned}$$

The FT of the extrapolated view is

$$U(\omega_x, \omega_y) = H_2(\omega_x) \cdot V(\omega_x, \omega_y) \tag{4.9}$$

where

$$H_2(\omega_x) = \frac{1}{2}\left(e^{jcos\theta\overline{\Delta d}\,\omega_x} + e^{jMcos\theta\overline{\Delta d}\,\omega_x}\right) \tag{4.10}$$

When $M = 2$, it corresponds to the proposed view extrapolation based MVC configuration in Fig. 3.5, where the two closest left views are used for extrapolation.

4.3 An Improved RD Model for MVC

In this section, by modifying the theoretical framework in [34, 53, 54], we first derive the bit rate difference over the simulcast scheme when encoding one frame of the multiview sequences with different MVC methods, including the proposed RVI and RVE-based schemes,

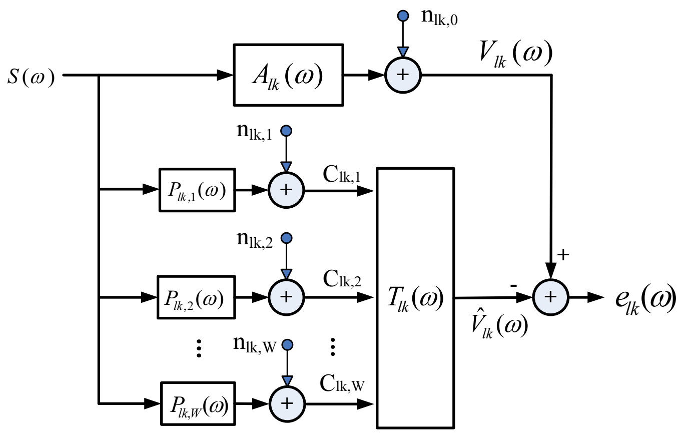


Figure 4.2: The theoretical model of encoding one frame.

and MDE-based JMVC. We then obtain the average bit rate difference when encoding all frames of the multiview system, by considering the different prediction configurations of different frames.

We denote L and K to be the number of views and the number of frames per GOP, respectively. For simplicity purpose, we assume L is odd, and $K = 2^B$, where B is the levels of hierarchical B-frames in each GOP. The k-th frame in a GOP of the l-th view is denoted as v_{lk}. We use $i_k \in [0, \ldots, B]$ to represent the temporal hierarchy of v_{lk}, where $i_k = 0$ corresponds to the temporal intra-coded frames.

Note that the mathematical model in [35, 52] aims to find the upper bound of the rate difference of MVC. It therefore assumes all motion and disparity-compensated frames in the L views and K frames are jointly encoded by a global Karhunen-Lòeve Transform (KLT), which is not the case in practical systems. The model developed here considers the coding of each frame. Therefore it matches the practical codecs more accurately, and can be used to compare the performances of different MVC schemes. It also allows us to study the impact of view rectification and other prediction methods on the coding efficiency of the entire MVC system.

4.3.1 RD Model for the Coding of One Frame

To study the theoretical RD performances of encoding one frame using different MVC schemes, the statistical model in Fig. 4.2 is used, which is modified from those in [34, 53, 54]. Since predictive coding is used in practical MVC, we use the linear minimum mean squared error (LMMSE) theory to find the prediction residual of each frame. The RD theory is then used to obtain the bit rate difference of different methods over the simulcast scheme, where each view is coded independently using hierarchical B structure.

The first branch in Fig. 4.2 represents our source model for the MVV data. Similar to [35], we assume that all frames of the MVV system are generated from a common root frame s by some translations or shifts. More precisely, v_{lk} is obtained from s by two shifts, $\mathbf{\Gamma}_l = [\Gamma_{lx}, \Gamma_{ly}]^T$ and $\mathbf{\Upsilon}_k = [\Upsilon_{kx}, \Upsilon_{ky}]^T$, and an additive white Gaussian noise $n_{lk,0}$, where $\mathbf{\Gamma}_l$ is caused by the different camera viewpoints between s and v_{lk}, $\mathbf{\Upsilon}_k$ is caused by their different observation times, and $n_{lk,0}$ represents the components of v_{lk} that cannot be modeled by the simple translation. We also assume that $n_{lk,0}$ is independent of s. As a result, let

$$A_{lk}(\omega) = e^{-j\omega^T \mathbf{\Gamma}_l} e^{-j\omega^T \mathbf{\Upsilon}_k} \tag{4.11}$$

where $\omega = [\omega_x, \omega_y]^T$, the FT of frame v_{lk} can be written as

$$V_{lk}(\omega) = A_{lk}(\omega) S(\omega) + N_{lk,0}(\omega) \tag{4.12}$$

where $S(\omega)$ and $N_{lk,0}(\omega)$ are the FT of the root frame s and the Gaussian noise $n_{lk,0}$, respectively.

We next model the predictive coding with multiple reference frames. As in [34, 35, 53, 54], we assume that v_{lk} is predicted from a vector of up to W compensated references $\mathbf{c}_{lk} = [c_{lk,1}, c_{lk,2}, \ldots, c_{lk,W}]^T$, as shown in Fig. 4.2. These references can come from motion compensation in the temporal direction, disparity compensation in the inter-view direction, or VSP from the inter-view direction. We assume that each reference image $c_{lk,i}$ is related to the root frame s by a transfer function $P_{lk,i}(\omega)$ and an additive white Gaussian noise $n_{lk,i}$ that is independent of s. $P_{lk,i}(\omega)$ models the different prediction methods with limited accuracy, and $n_{lk,i}$ captures the components that cannot be described by the transfer functions.

The expressions of $P_{lk,i}(\omega)$ are given in Appendix 4.A, which shows that there are only five distinctive cases of $P_{lk,i}(\omega)$ for the MVC schemes considered in this thesis. As in [35], the disparity-compensated reference is modeled as the root signal s shifted by a disparity error $\mathbf{\Xi} = [\Xi_x, \Xi_y]^T$. The motion-compensated reference is modeled as s shifted by a displacement error $\mathbf{\Theta}_{f,i_k} = [\Theta_{f,i_k x}, \Theta_{f,i_k y}]^T$ or $\mathbf{\Theta}_{b,i_k} = [\Theta_{b,i_k x}, \Theta_{b,i_k y}]^T$, where $\mathbf{\Theta}_f$ and $\mathbf{\Theta}_b$ indicate the displacement error of the forward and backward motion-compensated reference, respectively. When a reference is obtained by the proposed RVI or RVE, $P_{lk,i}(\omega)$ becomes $H_1(\omega)$ or $H_2(\omega)$ in (4.8) and (4.10).

We assume the components in $\overline{\Delta d}$, $\boldsymbol{\Xi}$, $\boldsymbol{\Theta}_{f,i_k}$ and $\boldsymbol{\Theta}_{b,i_k}$ are independent zero-mean Gaussian random variables with variances σ_d^2, σ_Ξ^2, $\sigma_{\Theta_{i_k}}^2$ and $\sigma_{\Theta_{i_k}}^2$, respectively. In [34, 35], $\sigma_{\Theta_{i_k}}^2$ is assumed to be identical in all temporal hierarchies, which is not true in practices, as the prediction accuracy usually decreases as the increase of the reference frame distance. To model this effect, we assume the variance of the displacement error is related to the B-frame hierarchy by

$$\sigma_{\Theta_{i_k}}^2 = 2^{2(B-i_k)}\sigma_\Theta^2, \quad i_k = 1, \ldots, B \tag{4.13}$$

where σ_Θ^2 is the variance of the displacement error of the B frames in the finest hierarchy.

Given $P_{lk,i}(\omega)$, assuming that the noises are uncorrelated with each others, it is easy to show that the LMMSE filter in Fig. 4.2 to estimate frame v_{lk} is [34]

$$\mathbf{T}_{lk}(\omega) = A_{lk}(\omega)\boldsymbol{\Phi}_{\mathbf{c}s}^\dagger(\omega)\boldsymbol{\Phi}_{\mathbf{cc}}^{-1}(\omega) \tag{4.14}$$

where $\boldsymbol{\Phi}_{\mathbf{c}s}(\omega)$ and $\boldsymbol{\Phi}_{\mathbf{cc}}(\omega)$ are the cross spectral density functions between $\mathbf{c}_{lk}$ and s, and the PSD of $\mathbf{c}_{lk}$ respectively, and $\dagger$ is the conjugate transpose. Due to space limitation, we drop the subscript lk in $\boldsymbol{\Phi}_{\mathbf{c}s}(\omega)$ and $\boldsymbol{\Phi}_{\mathbf{cc}}(\omega)$. Let

$$\mathbf{F}_{lk}(\omega) = [P_{lk,1}(\omega), P_{lk,2}(\omega), \ldots, P_{lk,W}(\omega)]^T \tag{4.15}$$

we then have

$$\begin{aligned} \boldsymbol{\Phi}_{\mathbf{c}s} &= \mathbf{F}_{lk}\Phi_{ss} \\ \boldsymbol{\Phi}_{\mathbf{cc}} &= \mathbf{F}_{lk}\Phi_{ss}\mathbf{F}_{lk}^\dagger + \boldsymbol{\Phi}_{\mathbf{nn}} = \Phi_{ss}\mathbf{F}_{lk}\mathbf{F}_{lk}^\dagger + \boldsymbol{\Phi}_{\mathbf{nn}} \end{aligned} \tag{4.16}$$

where the variable ω is not shown due to space limitation, and $\boldsymbol{\Phi}_{\mathbf{nn}}$ is the PSD of the noise vector $[n_{lk,1}, n_{lk,2}, \ldots, n_{lk,W}]$.

Note from (4.11) that $A_{lk}(\omega)A_{lk}^*(\omega) = 1$. Therefore it can be shown that with the LMMSE filter in (4.14), the PSD of the corresponding prediction error $e_{lk}(\omega)$ is

$$\Phi_{e_{lk}e_{lk}} = \Phi_{ss} - \boldsymbol{\Phi}_{\mathbf{c}s}^\dagger\boldsymbol{\Phi}_{\mathbf{cc}}^{-1}\boldsymbol{\Phi}_{\mathbf{c}s} + \Phi_{n_0n_0} \tag{4.17}$$

where $\Phi_{n_0n_0}$ is the PSD of the noise $n_{lk,0}$.

The transfer function $\mathbf{F}_{lk}(\omega)$ depends on the coding structure. As shown in Appendix 4.A, $\mathbf{F}_{lk}(\omega)$ is determined by random variables such as $\overline{\Delta d}$, $\boldsymbol{\Xi}$, $\boldsymbol{\Theta}_{f,i_k}$ and $\boldsymbol{\Theta}_{b,i_k}$. As in [34, 35, 53, 54], we are interested in the expected values of (4.16):

$$\begin{aligned} \boldsymbol{\Phi}_{\mathbf{cs}} &= E\left\{\mathbf{F}_{lk}\right\}\Phi_{ss} \\ \boldsymbol{\Phi}_{\mathbf{cc}} &= \Phi_{ss}E\left\{\mathbf{F}_{lk}\mathbf{F}_{lk}^\dagger\right\} + \boldsymbol{\Phi}_{\mathbf{nn}} \end{aligned} \tag{4.18}$$

Given Φ_{ss}, $\Phi_{n_0n_0}$ and $\mathbf{\Phi_{nn}}$, $\Phi_{e_{lk}e_{lk}}$ is determined by $E\left\{\mathbf{F}_{lk}\right\}$ and $E\left\{\mathbf{F}_{lk}\mathbf{F}_{lk}^{\dagger}\right\}$, which are given in Appendix 4.A for all frames in four MVC schemes, namely, RVI-based MVC, RVE-based MVC, MDE-based JMVC, and simulcast.

Once $\Phi_{e_{lk}e_{lk}}$ in (4.17) is known for all MVC schemes, the RD theory can be used to calculate the bit rate difference between different MVC schemes [34, 35, 50, 51, 52, 53, 54]. In this thesis, we use the simulcast coding as the reference scheme. The bit rate difference between a given MVC scheme and the simulcast method for the coding of the frame v_{lk} is thus

$$\Delta R_{lk} = \frac{1}{8\pi^2}\int_{\omega_x}\int_{\omega_y}\log_2\frac{\Phi_{e_{lk}e_{lk}}(\omega_x,\omega_y)}{\Phi_{e_{lk}e_{lk},0}(\omega_x,\omega_y)}d\omega_x d\omega_y \tag{4.19}$$

where $\Phi_{e_{lk}e_{lk},0}(\omega_x,\omega_y)$ is the PSD of the residue signal e_{lk} in the simulcast approach.

4.3.2 Average Coding Gain of the Entire Multiview System

The bit rate difference over the simulcast in (4.19) is for one frame. In this part, we compute the average bit rate difference for all frames and all views of a GOP.

Since the first view is coded by the same simulcast method in all MVC schemes, this view does not contribute anything to the overall bit rate difference. As discussed in Appendix 4.A, the coding of other views can be divided into five cases.

We first derive the average coding gain of the RVI-based MVC. It can be seen from Fig. 3.3 that except for view v_0, all even-indexed views are coded with MDE-based prediction as in JMVC, whereas RVI is applied to all odd-indexed views. For stationary signals, the coding efficiency of all odd (even)-indexed views should be the same, so the average bit rate difference of RVI-based MVC over simulcast is

$$\Delta R_{RVI} = \frac{1}{LK}\left(\frac{L-1}{2}\sum_{k=0}^{K-1}\Delta R_k^{RVI} + \frac{L-1}{2}\sum_{k=0}^{K-1}\Delta R_k^{JMVC,P}\right) \tag{4.20}$$

where ΔR_k^{RVI} and $\Delta R_k^{JMVC,P}$ is the rate difference of a RVI-coded frame in RVI-based MVC and a P frame in JMVC, respectively. They can be calculated by (4.19) with (4.25)-(4.26) and (4.28)-(4.29) from Appendix 4.A, respectively. Note that their values depend on the B-frame hierarchy of frame k.

For the RVE-based MVC, as shown in Fig. 3.5, view v_1 is coded by MDE-based prediction with one inter-view reference. All other views are coded with RVE. Therefore, the average bit rate difference over simulcast is

$$\Delta R_{RVE} = \frac{1}{LK}\left((L-2)\sum_{k=0}^{K-1}\Delta R_k^{RVE} + \sum_{k=0}^{K-1}\Delta R_k^{MDE,P}\right) \tag{4.21}$$

where ΔR_k^{RVE} and $\Delta R_k^{MDE,P}$ are the rate difference of a RVE-coded frame in all RVE-coded views and a MDE-coded frame in view v_1, which can be derived from (4.19) using (4.25)-(4.27) from Appendix 4.A, respectively.

In the JMVC, the frames in all the even-indexed views except for view v_0 are inter-view coded as P frame using the reference from the last even-indexed view, and all odd-indexed views are inter-view coded as B frames, using the neighboring left and right views. Therefore, the average rate difference of the JMVC over simulcast is

$$\Delta R_{JMVC} = \frac{1}{LK}\left(\frac{L-1}{2}\sum_{k=0}^{K-1}\Delta R_k^{JMVC,B} + \frac{L-1}{2}\sum_{k=0}^{K-1}\Delta R_k^{JMVC,P}\right) \tag{4.22}$$

where $\Delta R_k^{JMVC,B}$ is the rate difference of a B frame in the odd-indexed views of JMVC, which can be obtained by (4.19) with (4.30) and (4.31) from Appendix 4.A.

4.4 Simulation Results

In this section, we use the RD model in Sec. 4.3 and the average rate difference to analyze the theoretical performance of RVI/RVE-based MVC and JMVC, and explain the experimental results in Chapter 3. In Fig. 4.2, let c_1 be the RVI or RVE-based reference, c_2 the inter-view disparity compensated reference, and c_3 and c_4 the temporal motion compensated references. The noises in the figure are denoted by n_0 to n_4.

As in [51, 52, 53, 54, 55], we assume the PSD of the root image s is

$$\Phi_{ss}(\omega_x, \omega_y) = \frac{2\pi}{\omega_0^2}\left(1 + \frac{\omega_x^2+\omega_y^2}{\omega_0^2}\right)^{-\frac{3}{2}}, \quad \omega_0 = -ln(\rho) \tag{4.23}$$

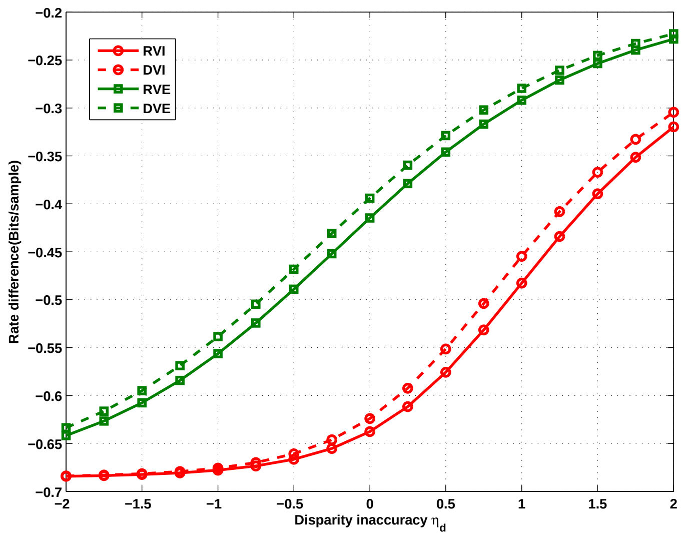
where $\rho = 0.93$ is the correlation between adjacent pixels in the image. Note that the original signal in (4.23) is normalized with the variance $\sigma_s^2 = 1$ [51].

The camera orientation angle θ is set to be 25°, which corresponds to the optimal planar camera placement for stereo surface reconstruction [80]. The motion estimation inaccuracy $\log_2\left(\sqrt{12}\sigma_\Theta\right)$ and inter-view disparity estimation inaccuracy $\log_2\left(\sqrt{12}\sigma_\Xi\right)$ are chosen to be 0, which corresponds to integer-pixel motion and disparity estimations [35]. The source residual noise level (SRNL) $10\log_{10}\left(\sigma_{n0}^2\right)$, the temporal residual noise level (TRNL) $10\log_{10}\left(\sigma_{n_3}^2\right)$ and $10\log_{10}\left(\sigma_{n_4}^2\right)$ are set as -20 dB, which is typical for natural videos [34]. The inter-view residual noise level (IVRNL) is usually higher than temporal noise, so $10\log_{10}\left(\sigma_{n_2}^2\right)$ is chosen as -15 dB.

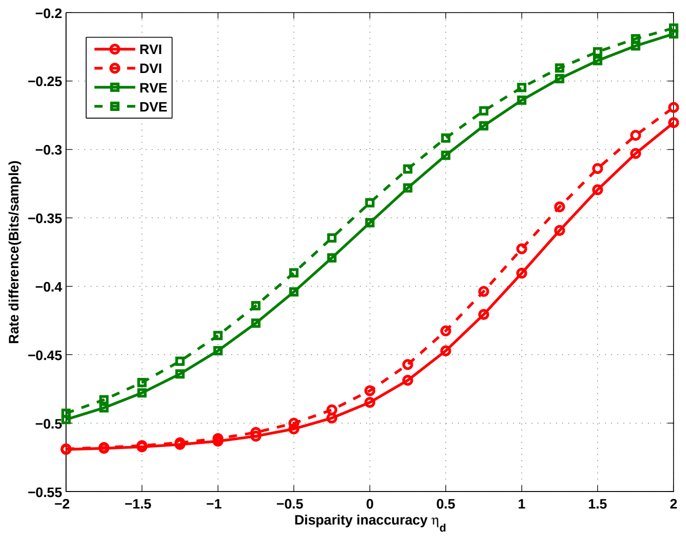
Fig. 4.3 shows the effect of the VSP disparity inaccuracy, *i.e.*, $\eta_d \triangleq \log_2\left(\sqrt{12}\sigma_d\right)$, on the rate difference of coding one view using RVI/DVI-based MVC, and RVE/DVE-based MVC, with different view synthesis residual noise level (VSRNL) $10\log_{10}\left(\sigma_{n_1}^2\right)$. It shows that view interpolation-based MVCs save more bits than extrapolation, and the gain can be more than 0.2 bits/sample. The rectification process yields additional coding gain over direct view synthesis, although the improvement is limited and becomes negligible as the decrease of η_d. These theoretical results agree well with the experiments in Fig. 3.10 for single view coding.

Fig. 4.4 depicts the average rate difference of various MVC schemes across all views for different numbers of views L, with $\eta_d = 0$ (integer-pixel VSP disparity accuracy) and different VSRNL. As shown in Fig. 4.4, both RVI and RVE-based methods outperform JMVC. The rate saving can be improved by all methods when more viewpoints are encoded, but most of the improvements are obtained when the number of views is less than 10. Similar to Fig. 4.3, the view rectification offers additional gain in both RVI and RVE, and the gain is larger in RVE. More importantly, although RVI-based MVC outperforms RVE-based MVC when L is very small, the RVE-based MVC can achieve better performance than RVI-based MVC as the increase of the number of views, because RVE can be applied to more viewpoints, instead of only half of the views in the RVI-based MVC. This verifies the experimental results in Fig. 3.11.

Fig. 4.5 shows the average rate difference of various MVC schemes for different temporal GOP size K when $L = 3$ and $L = 9$ respectively, with $\eta_d = 0$ and VSRNL of -20 dB. It is shown that smaller GOP size K provides larger gain than larger GOP size, and the gain becomes negligible as K increases. This result agrees with the theoretical analysis in [35]. Moreover, the bit rate saving of RVI-based MVC is better than that of RVE-based MVC when $L = 3$, but RVE-based MVC outperforms RVI-based MVC when $L = 9$. This verifies the result obtained from Fig. 4.4. That is, the RVE-based MVC is more beneficial to the overall coding gain when the number of encoded views is increased.

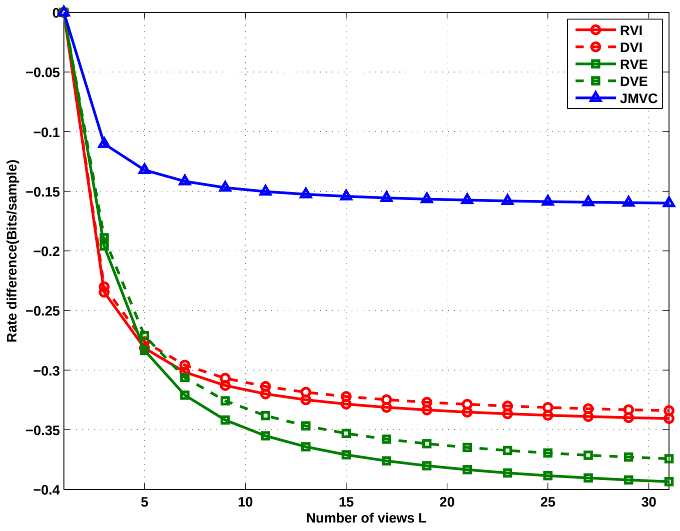


(a)

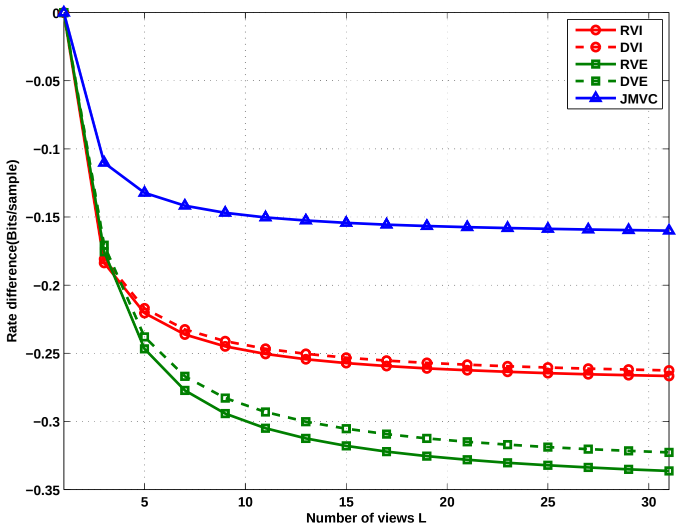


(b)

Figure 4.3: Rate difference of single view coding versus VSP disparity inaccuracy η_d. The VSRNL is -25 dB in (a) and -20 dB in (b).

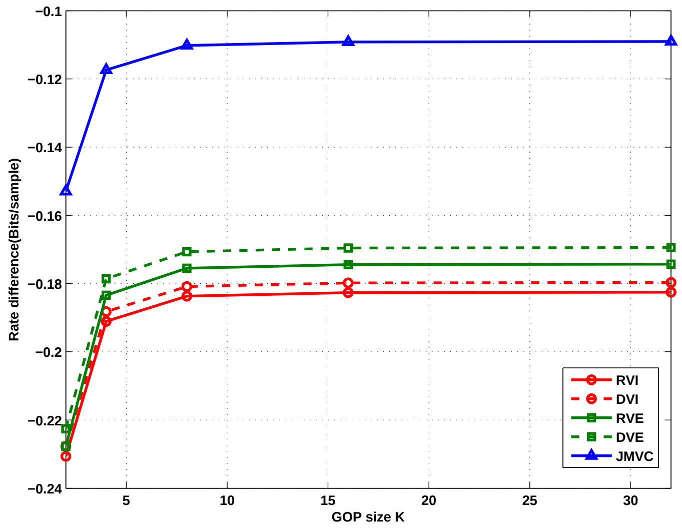


(a)

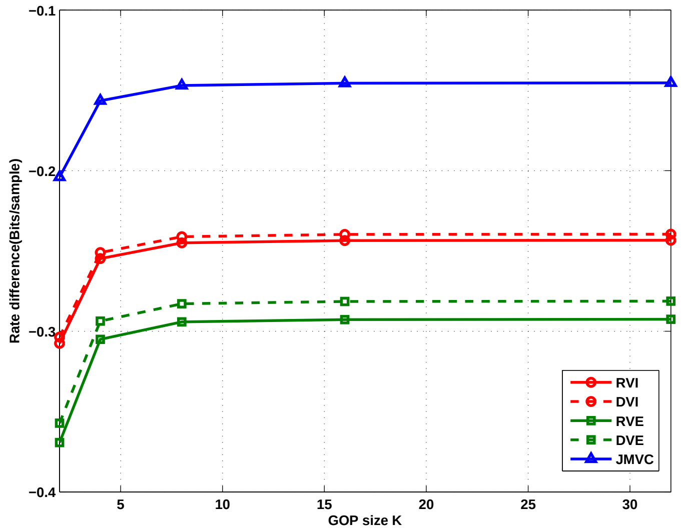


(b)

Figure 4.4: Rate difference of all views with different MVC schemes and different number of views L. The VSRNL is -25 dB in (a) and -20 dB in (b).



(a)



(b)

Figure 4.5: Average rate difference of different MVC schemes with different GOP size K. The number of views is $L = 3$ in (a) and $L = 9$ in (b).

4.5 Summary

In this chapter, we develop a geometric model to study the performance of RVI and RVE-based view synthesis algorithms, In addition, we propose an improved model to analyze the RD performances of various practical MVC schemes, including RVI/DVI-based MVC, RVE/DVE-based MVC and MDE-based JMVC as well as simulcast. Simulation results obtained from those theoretical models help us explain the experimental results in Chapter 3.

4.A The LMMSE Filters in Various MVC Schemes

In this appendix, we derive the expressions of the terms $E\left\{\mathbf{F}_{lk}\right\}$ and $E\left\{\mathbf{F}_{lk}\mathbf{F}_{lk}^{\dagger}\right\}$ in (4.18) for all the frames in RVI/RVE-based MVC, MDE-based JMVC and simulcast.

Recall that view 0 is simulcast-coded in all four schemes. For the coding of other views, by inspecting the prediction structures of the four MVC schemes, we can see that there are only five different cases, *i.e.*, the VSP-coded views in RVI-based MVC, the VSP-coded views in RVE-based MVC, view v_1 in RVE-based MVC, the even-indexed views in RVI-based MVC and JMVC, and the odd-indexed views in the JMVC. The details of these cases are given below.

4.A.1 Case 1 and Case 2

For VSP-coded views in RVI/RVE-based MVC, as in Fig. 3.3 and Fig. 3.5, every frame is predicted from four references: the reference from the proposed RVI/RVE, the inter-view reference from the left view, and two temporal references in the hierarchical B structure. Therefore, $F_{lk}(\omega)$ in (4.15) for RVI/RVE-coded frame v_{lk} with the i_k-th temporal hierarchy is obtained as

$$\mathbf{F}_{lk}^{(m)} = \left[H_m(\omega), e^{-j\omega^T\mathbf{\Xi}}, e^{-j\omega^T\mathbf{\Theta}_{f,i_k}}, e^{-j\omega^T\mathbf{\Theta}_{b,i_k}}\right] \tag{4.24}$$

where the disparity error $\mathbf{\Xi}$ and displacement errors $\mathbf{\Theta}_{f,i_k}$ and $\mathbf{\Theta}_{b,i_k}$ are defined in Sec. 4.3.1. $m = 1$ and $m = 2$ correspond to RVI-coded frames and RVE-coded frames, respectively, and $H_1(\omega)$ and $H_2(\omega)$ are defined in (4.8) and (4.10).

By our models and the property of the characteristic function of Gaussian random variables, the expectation of (4.24) is

$$E\left\{\mathbf{F}_{lk}^{(m)}\right\} = \left[\varphi_m(\omega),\ \psi(\omega),\ \phi_{i_k}(\omega),\ \phi_{i_k}(\omega)\right]^T \tag{4.25}$$

for $m = 1, 2$, where

$$\begin{aligned}
\varphi_1(\omega) &= E\left\{H_1(\omega)\right\} = e^{-\frac{1}{2}M^2 cos^2\theta\omega_x^2\sigma_d^2} \\
\varphi_2(\omega) &= E\left\{H_2(\omega)\right\} = \frac{1}{2}(e^{-\frac{1}{2}cos^2\theta\omega_x^2\sigma_d^2} + e^{-\frac{1}{2}M^2 cos^2\theta\omega_x^2\sigma_d^2}) \\
\psi(\omega) &= e^{-\frac{1}{2}\omega^T\omega\sigma_\Xi^2},\ \phi_{i_k}(\omega) = e^{-\frac{1}{2}\omega^T\omega\sigma_{\Theta_{i_k}}^2}
\end{aligned}$$

In addition,

$$E\left\{\mathbf{F}_{lk}^{(m)}\mathbf{F}_{lk}^{(m)\dagger}\right\} = \\
\begin{bmatrix} G_{m,1}(\omega) & G_{m,2}(\omega) & G_{m,3i_k}(\omega) & G_{m,3i_k}(\omega) \\ G_{m,2}(\omega) & 1 & G_{4i_k}(\omega) & G_{4i_k}(\omega) \\ G_{m,3i_k}(\omega) & G_{4i_k}(\omega) & 1 & G_{5i_k}(\omega) \\ G_{m,3i_k}(\omega) & G_{4i_k}(\omega) & G_{5i_k}(\omega) & 1 \end{bmatrix} \tag{4.26}$$

for $m = 1, 2$, where

$$\begin{aligned}
G_{1,1}(\omega) &= E\left\{\|H_1(\omega)\|^2\right\} = \frac{1}{2}\left(1 + e^{-2M^2 cos^2\theta\omega_x^2\sigma_d^2}\right) \\
G_{2,1}(\omega) &= E\left\{\|H_2(\omega)\|^2\right\} = \frac{1}{2}\left(1 + e^{-\frac{1}{2}(M-1)^2 cos^2\theta\omega_x^2\sigma_d^2}\right) \\
G_{m,2}(\omega) &= \varphi_m(\omega)\,\psi(\omega),\ \ G_{m,3i_k}(\omega) = \varphi_m(\omega)\,\phi_{i_k}(\omega) \\
G_{4i_k}(\omega) &= \psi(\omega)\,\phi_{i_k}(\omega), \qquad G_{5i_k}(\omega) = \phi_{i_k}^2(\omega)
\end{aligned}$$

Note that the calculation of the PSD of the prediction errors for the DVI-coded and DVE-coded frames (defined in Chapter 3) is a special case of this model with $\theta = 0$, *i.e.*, without view rectification.

4.A.2 Case 3

The coding of view v_1 in the RVE-based MVC (Fig. 3.5) uses hierarchical B structure in the temporal direction and the disparity compensated view v_0 in the inter-view direction. Therefore, the transfer function is obtained from (4.25) by omitting the first item, *i.e.*,

$$E\left\{\mathbf{F}_{lk}\right\} = [\psi(\omega),\ \phi_{i_k}(\omega),\ \phi_{i_k}(\omega)]^T \tag{4.27}$$

and the corresponding $E\left\{\mathbf{F}_{lk}\mathbf{F}_{lk}^\dagger\right\}$ is simply the lower-right 3×3 submatrix of (4.26).

4.A.3 Case 4

For even-indexed views in RVI-based MVC (Fig. 3.3) or in JMVC with I-B-P inter-view coding, the prediction functions are similar to (4.27), except that the inter-view reference is the previous even-indexed view instead of the immediate neighboring view. As in (4.13), we assume the variance of the disparity error is increased to four times. Thus,

$$E\left\{\mathbf{F}_{lk}\right\} = \left[\psi^4(\omega),\ \phi_{i_k}(\omega),\ \phi_{i_k}(\omega)\right]^T \tag{4.28}$$

$$E\left\{\mathbf{F}_{lk}\mathbf{F}_{lk}^{\dagger}\right\} = \begin{bmatrix} 1 & G_{6i_k}(\omega) & G_{6i_k}(\omega) \\ G_{6i_k}(\omega) & 1 & G_{5i_k}(\omega) \\ G_{6i_k}(\omega) & G_{5i_k}(\omega) & 1 \end{bmatrix} \tag{4.29}$$

where $G_{6i_k}(\omega) = \psi^4(\omega)\phi_{i_k}(\omega)$.

4.A.4 Case 5

Frames in the odd-indexed views of the JMVC method have two temporal hierarchical B references and two inter-view references from the left and right views. Therefore

$$E\left\{\mathbf{F}_{lk}\right\} = \left[\psi(\omega),\ \psi(\omega),\ \phi_{i_k}(\omega),\ \phi_{i_k}(\omega)\right]^T \tag{4.30}$$

$$E\left\{\mathbf{F}_{lk}\mathbf{F}_{lk}^{\dagger}\right\} = \begin{bmatrix} 1 & G_7(\omega) & G_{4i_k}(\omega) & G_{4i_k}(\omega) \\ G_7(\omega) & 1 & G_{4i_k}(\omega) & G_{4i_k}(\omega) \\ G_{4i_k}(\omega) & G_{4i_k}(\omega) & 1 & G_{5i_k}(\omega) \\ G_{4i_k}(\omega) & G_{4i_k}(\omega) & G_{5i_k}(\omega) & 1 \end{bmatrix} \tag{4.31}$$

where $G_7(\omega) = \psi^2(\omega)$.

Finally, the simulcast scheme independently encodes each view using the hierarchical B structure. Therefore, each frame is predicted from two temporal references. Thus

$$\mathbf{F}_{lk} = \left[\phi_{i_k}(\omega),\ \phi_{i_k}(\omega)\right]^T \tag{4.32}$$

and the corresponding $E\left\{\mathbf{F}_{lk}\mathbf{F}_{lk}^{\dagger}\right\}$ is simply the lower-right 2×2 submatrix of (4.26).

In addition, the first frame of each view in each GOP is encoded in the temporal direction using the intra coding. It is straightforward to extract from (4.25) and (4.26) the corresponding functions by removing the temporal prediction related items.

Chapter 5

Improved Depth Map Estimation

5.1 Introduction

Depth estimation is a key component of processing MVV data, which directly affects the quality of arbitrary view rendering via DIBR. Almost all the traditional schemes are based on stereo matching, which first estimates the disparity map between two cameras, and then calculates the depth value using the given camera parameters and pose information [81, 82]. Since the complexity of the disparity estimation-based depth estimation is quite high, they are difficult to be applied in real-time applications. To solve this problem, a 3D warping-based block depth estimation (BDE) algorithm was proposed to directly extract the optimal depth from multiview images [83]. Although this method yields better synthesis quality than the methods in computer vision, it still involves large amount of search operations, and the resulting depth map is difficult to compress.

In [42], the multiview epipolar geometry was employed to predict the disparity value, which can reduce the complexity of disparity matching in multiview image coding. Inspired by this approach, in this chapter we propose a geometric depth prediction algorithm for depth estimation. To the best of our knowledge, this is the first epipolar geometry-based depth prediction method for depth estimation.

Using the matched block obtained from epipolar geometry, we can find an initial estimation of the depth of the block by the triangulation method and depth projection [20]. To compensate the error in the calculation of epipolar geometry, 3D warping is then applied to refine the predicted depth.

On the other hand, all the aforementioned depth estimation algorithms are developed for the two-view configuration, where the depth map of the target view can only be estimated from one reference view. However, in a MVV system, it might be necessary to estimate the depth from multiple references. Accordingly, we develop a simple scheme to fuse the depth estimations from multiple references, by using a structural similarity (SSIM) [84] and maximum likelihood-based approach.

To facilitate the efficient compression of the depth map, we also propose a depth map smoothing algorithm, using color segmentation and plane fitting. The RANSAC algorithm [20] is used to derive the optimal parameters for each depth plane, which improves the robustness of the algorithm to depth errors.

Experimental results show that compared to the existing BDE-based depth estimation methods, the proposed method not only achieves improved view synthesis quality, but also produces smooth depth maps that require much fewer bits to encode.

The rest of this chapter is organized as follows. In Sec. 5.2, the original BDE algorithm is briefly introduced. In Sec. 5.3, we present the details of the proposed depth estimation algorithm and explain its improvements over previous methods. Experimental results of the proposed method on view synthesis and depth coding are given in Sec. 5.4, followed by summarizing the works of this chapter in Sec. 5.5.

5.2 3D Warping-based Block Depth Estimation

Let $\mathbf{A}_n$, $\mathbf{R}_n$ and $\mathbf{T}_n$ denote the intrinsic matrix, rotation matrix and translation vector of the target view I_n. Consider a block centered at coordinate (x_n, y_n) in I_n. If the depth value of the block is d_n, we can map the block onto a reference view, such as I_{n-1}, by 3D warping. First, using (2.8), the block location is projected into 3D world coordinate by

$$\mathbf{X} = \mathbf{R}_n \mathbf{A}_n^{-1} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} d_n + \mathbf{T}_n \tag{5.1}$$

where $\mathbf{X}$ represents a 3D world coordinate. Then, using (2.6), $\mathbf{X}$ can be re-projected onto the reference view via

$$d_{n-1} \begin{bmatrix} x_{n-1} \\ y_{n-1} \\ 1 \end{bmatrix} = \mathbf{A}_{n-1} \mathbf{R}_{n-1}^{-1} (\mathbf{X} - \mathbf{T}_{n-1}) \tag{5.2}$$

where (x_{n-1}, y_{n-1}) is the projected coordinate in I_{n-1}, and d_{n-1} is the corresponding depth value. If a synthesized view is needed for I_n, the pixel values of the block at (x_n, y_n) will be chosen as $I'_n(x_n, y_n) = I_{n-1}(x_{n-1}, y_{n-1})$.

The best depth of the block at (x_n, y_n) is obtained by the following minimization procedure

$$d_n^* = \arg\min_{d_n} ||I_n(x_n, y_n) - I_{n-1}(x_{n-1}, y_{n-1})|| \tag{5.3}$$

where the minimization is carried out over the set $d_n \in [0, 255]$.

Since the cost function seeks to minimize the prediction error, the BDE method can achieve a good synthesis quality. However, the resulting depth maps are usually very noisy and not suitable for compression. For example, the depth map and the corresponding synthesis result of one frame from the sequence `Akko&Kayo` with 4×4-block-based depth estimation are shown in Fig. 5.1. Due to the lack of smoothness, the conventional codecs, such as H.264/AVC, fail to encode the depth map with high compression efficiency while still maintaining a good synthesis quality [83].

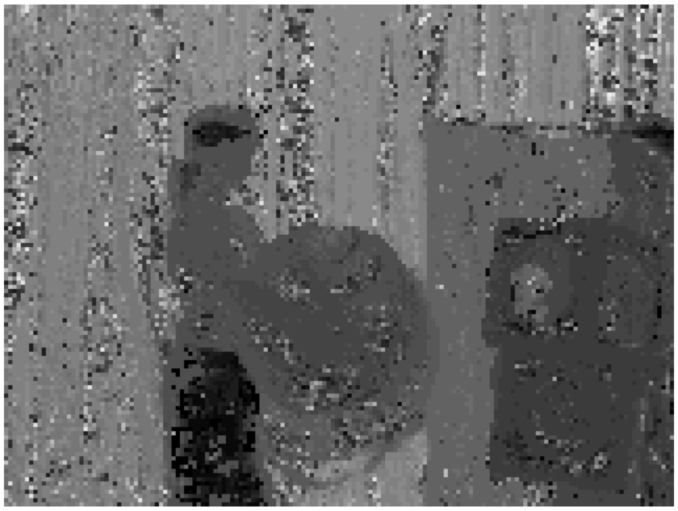
5.3 The Proposed Depth Estimation Algorithm

In this section, we propose a new depth estimation algorithm which can improve the result of the the BDE method in Sec. 5.2.

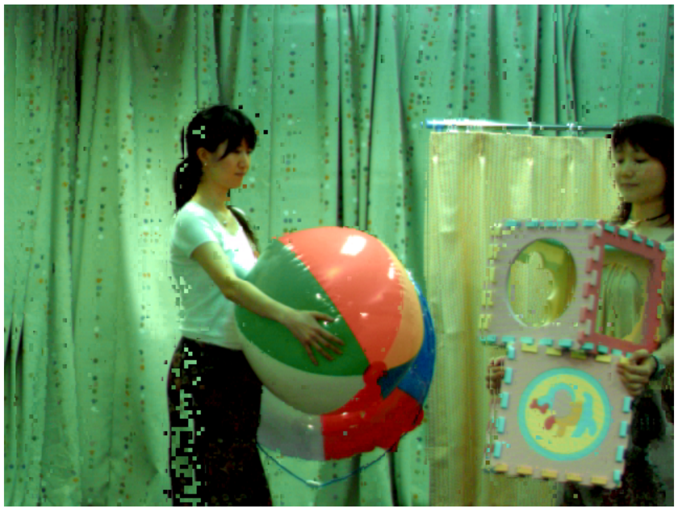
5.3.1 Geometric Depth Prediction

Unlike the original BDE which searches all possible depths to find the optimal depth with the minimum distortion, our proposed epipolar geometry-based method can accurately track the depth variation and reduce the matching complexity. Specifically, assuming the depth map of the target view I_n is to be predicted from a reference view I_{n-1}. With block-based estimation, our approach calculates the depth value for a block in I_n centered at (x_n, y_n) by four steps.

1. We first search for the corresponding block in the reference I_{n-1} using epipolar geometry.
2. We then compute the 3D world coordinate of the corresponding block pair by triangulation method.



(a)



(b)

Figure 5.1: Performance of BDE-based depth estimation. (a) Depth map; (b) View synthesis result with PSNR 26.21 dB.

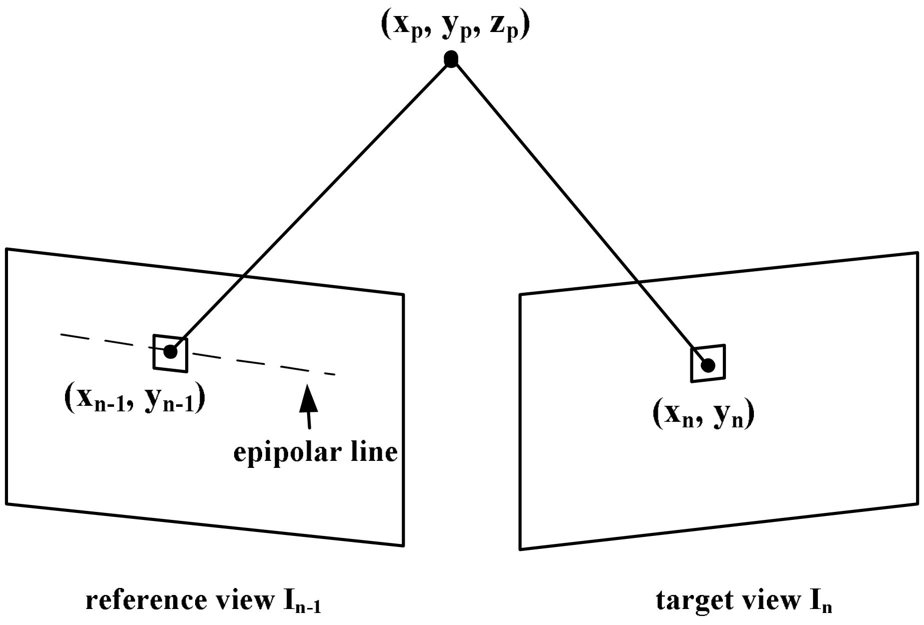


Figure 5.2: Geometric depth prediction.

3. We next project the 3D point back to the target view to calculate the predicted depth.
4. We finally refine the depth value within a small range around the predicted value.

Searching Corresponding Block Pairs

As introduced in Chapter 2.1.1, epipolar geometry is the geometry constraint between a stereo pair of cameras, which states that if a 3D point is projected to one view, its projection point in the other view must lie on its epipolar line. Therefore, as depicted in Fig. 5.2, we can constrain the search range of the corresponding block in I_{n-1} in a 1D segmentation along the epipolar line, which can reduce the searching complexity and block mismatching probability. Experimental results show that the result will be good enough by searching in the range of $[-25, 25]$ around the initial position on the epipolar line. The matching criterion is similar to that in (5.3), with (x_{n-1}, y_{n-1}) as the parameters to be optimized.

Note that due to computation errors, the corresponding block in the reference view is usually not exactly on the epipolar line. To address this problem, a rectangular search window is used in [42]. However, in our scheme, we only use the epipolar geometry-based method to generate an initial estimation of the depth, which will be refined later. Therefore a small 1D search on the epipolar line can be used, which has much lower complexity than the rectangular window search in [42] and the search in (5.3).

3D Coordinate Reconstruction

Once the corresponding block pair is available, the 3D coordinate of the two correspondence blocks in the 3D space can be calculated. Many 3D reconstruction algorithms have been proposed in the last decade. In this thesis, the 3D coordinate of each correspondence pair is recovered by the classic triangulation algorithm [20]. In Fig. 5.2, suppose (x_n, y_n) and (x_{n-1}, y_{n-1}) are the projection locations of the same spatial point (x_p, y_p, z_p) in the target view and reference view respectively. Given the camera matrices of the two views, we can have

$$d_n \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \mathbf{M}_n \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \qquad d_{n-1} \begin{bmatrix} x_{n-1} \\ y_{n-1} \\ 1 \end{bmatrix} = \mathbf{M}_{n-1} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \tag{5.4}$$

where d_n and d_{n-1} are the depths of two blocks, $\mathbf{M}_n$ and $\mathbf{M}_{n-1}$ are two 3×4 projection matrices of the two views, which can be easily derived from the intrinsic matrices, rotation matrices and translation vectors of the cameras [20]. To find (x_p, y_p, z_p), we can obtain the expressions of d_n and d_{n-1} from the third rows of the two equations, and substitute them into the first two rows. This leads to the following linear equations of (x_p, y_p, z_p).

$$\begin{aligned}
m_{14}^{n} - x_n m_{34}^{n} &= \left(x_n m_{31}^{n} - m_{11}^{n}\right) x_p + \left(x_n m_{32}^{n} - m_{12}^{n}\right) y_p + \left(x_n m_{33}^{n} - m_{13}^{n}\right) z_p \\
m_{24}^{n} - y_n m_{34}^{n} &= \left(y_n m_{31}^{n} - m_{21}^{n}\right) x_p + \left(y_n m_{32}^{n} - m_{22}^{n}\right) y_p + \left(y_n m_{33}^{n} - m_{23}^{n}\right) z_p \\
m_{14}^{n-1} - x_{n-1} m_{34}^{n-1} &= \left(x_{n-1} m_{31}^{n-1} - m_{11}^{n-1}\right) x_p + \left(x_{n-1} m_{32}^{n-1} - m_{12}^{n-1}\right) y_p \\
&\quad + \left(x_{n-1} m_{33}^{n-1} - m_{13}^{n-1}\right) z_p \\
m_{24}^{n-1} - y_{n-1} m_{34}^{n-1} &= \left(y_{n-1} m_{31}^{n-1} - m_{21}^{n-1}\right) x_p + \left(y_{n-1} m_{32}^{n-1} - m_{22}^{n-1}\right) y_p \\
&\quad + \left(y_{n-1} m_{33}^{n-1} - m_{23}^{n-1}\right) z_p
\end{aligned} \tag{5.5}$$

where $m_{i,j}^{k}$ is the (i, j)-th entry of the projection matrix M_k. The least-square method can be used to find the 3D position (x_p, y_p, z_p).

Depth Projection

After the second step, we recover the 3D coordinate of each corresponding block pair. Now, our objective is to calculate the depth d_n through the projection of the 3D point to I_n. Specifically, given the projection matrix $\mathbf{M}_n$ of the target view and the 3D coordinate (x_p, y_p, z_p), we can obtain the new projection point (x'_n, y'_n) with the depth value d'_n by (5.4). Note that if there is no any mismatch in the block matching process of the first step, (x'_n, y'_n) should be identical to (x_n, y_n). However, there are always camera noise and color distortion between different cameras, and the estimation error of the least-square method for 3D reconstruction can also deviate (x'_n, y'_n) from (x_n, y_n). So, in order to alleviate the effect of mismatch, the predicted depth value is calculated by averaging the estimations from both x- and y- directions, *i.e.*,

$$\hat{d_n} = \frac{d'_n}{2}\left(\frac{x'_n}{x_n} + \frac{y'_n}{y_n}\right). \tag{5.6}$$

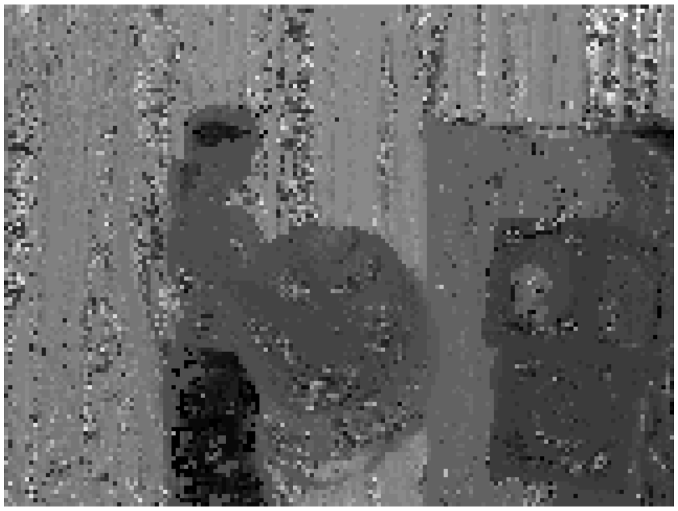
Depth Refinement

The initial depth obtained above is based on the epipolar geometry. To reduce the impact of possible errors in the epipolar geometry calculation, we refine the depth estimation using the 3D warping method in Sec. 5.2, by treating the predicted depth above as the initial value. Since the predicted depth already has reasonable quality, the search window can be much smaller than in (5.3). In this thesis, the search range for the refined depth is $[-10, 10]$.

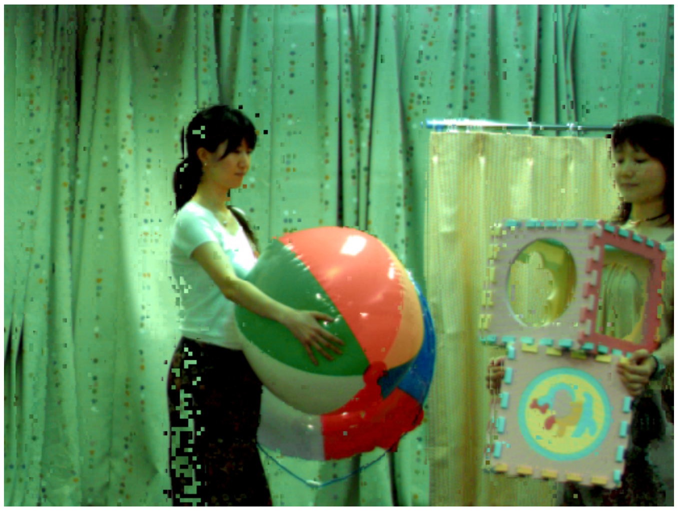
Depth Estimation and View Synthesis Results

Fig. 5.3 shows the depth map and view synthesis results generated by the proposed geometry-based depth estimation algorithm. Notice that the geometry-based method can achieve almost the same quality as that of the original BDE method in Fig. 5.1. However, the complexity of our approach is only about 27.5% of the exhaustive search based BDE method.

It should be mentioned that the complexity is measured by the number of iterations needed by one depth estimation algorithm to find the best match for each block. Also, the complexity of calculating the epipolar geometry is not included, as it can be easily derived from the projection matrices of the views [20]. Moreover, this process can be computed off-line before depth estimation.



(a)



(b)

Figure 5.3: Performance of geometry-based depth estimation. (a) Depth map; (b) View synthesis result with PSNR 26.20 dB.

5.3.2 Fusion of Depth Estimations from Multiple References

Different from the previous stereo setup where the depth map of one view can only be estimated from one reference, in a multiview system, the information from multiple reference views can be utilized for depth estimation. Therefore, it is necessary to develop an effective way to fuse multiple depth estimations.

Suppose M neighboring views are used as references for depth estimation. For each block in the target view, using the geometry-based depth estimation method discussed in Sec. 5.3.1, we can compute M depth values $d_n^k, k = 1, 2, \ldots, M$. We assume that the true depth d_n of the block follows a Gaussian mixture model

$$p(d_n) = \sum_{k=1}^{M} \frac{c_k}{C_0} W(d_n - d_n^k), \tag{5.7}$$

where c_k is the weighting parameter for the k-th estimate, $C_0 = \sum_{i=1}^{M} c_i$, and $W(x) = 1/\sqrt{2\pi}e^{-x^2/2}$.

In our scheme, the weighting parameter c_k is given by SSIM [84], which provides a faithful measure of the similarity of two image blocks, and is defined as

$$\frac{(2\mu_x\mu_y + C_1)(2\delta_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\delta_x^2 + \delta_y^2 + C_2\right)}, \tag{5.8}$$

where μ_x and μ_y are the mean values of the target block and the reference block, δ_x^2 and δ_y^2 are the variances of the two blocks, δ_{xy} is the covariance coefficient of the two blocks, C_1 and C_2 are two parameters to stabilize the division with weak denominator. The SSIM index is between -1 and 1, with 1 achieved when the two blocks are identical.

To find the best depth for the target block, we use the maximum likelihood criterion, *i.e.*, we search the depth value that maximizes the probability in (5.7). In our method, the bisection method is applied to the range $\left[\min(d_n^i), \max(d_n^i)\right], i = 1, 2, \ldots, M$ to find the optimal depth value.

Note that other similarity measures, such as the normalized cross correlation, sum of square difference and mutual information, can also be used as the weighting parameters in (5.7). However, it is found through our experiments that the SSIM achieves the best synthesis quality. In addition, using the two nearest views (one left view and one right view) is sufficient to obtain a satisfactory synthesis quality. Therefore, $M = 2$ is used in the following experiments.

5.3.3 Depth Map Smoothing Using Color Segmentation and Plane Fitting

It can be seen from Fig. 5.3(a) that the depth map created by the proposed geometry-based depth estimation algorithm is still very noisy, especially in the textureless regions, because of the ambiguity involved in block-based matching. The following two important observations can also be made.

- For a region with homogenous color, there is usually no large depth discontinuity.
- depth changes often coincide with color changes. These facts suggest that we can segment the target image into homogenous color regions and then model the depth values in each segment by a parametric surface.

An example of color segmentation-based depth estimation is [85]. Some commonly used parametric depth representations include plan-plus-parallax model [86, 87], 3D plane model [88] and high-order polynomial surface [89]. Due to its low complexity, plan-plus-parallax is the most popular model, and is adopted in this thesis. In addition, the fast color segmentation algorithm in [90] is used in our approach. An example of the color segmentation is given in Fig. 5.4.

Based on the methods described in Sec. 5.3.1 and Sec. 5.3.2, we can obtain a depth value for each pixel in the target view. Using those depth values as initials, a depth plane can be determined for each segment. In this thesis, the following model is used

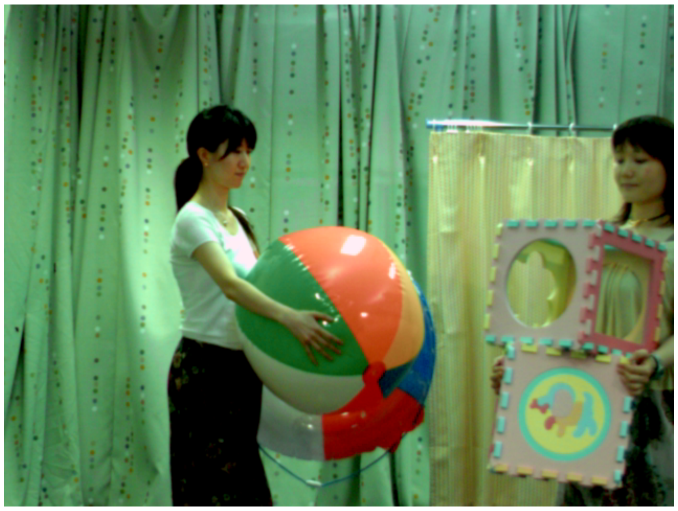
$$ax + by + c = d, \tag{5.9}$$

where (x, y) is an image point, and d is the depth value of the point. Grouping the equations of all pixels in one segment, we get

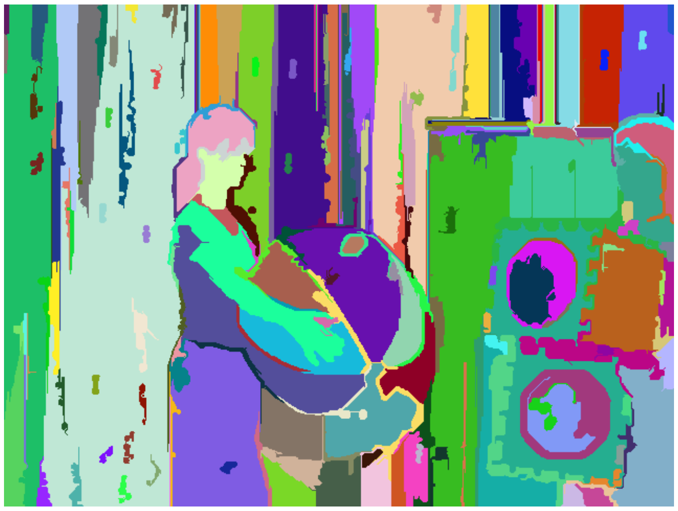
$$\mathbf{A} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{D} \tag{5.10}$$

where $\mathbf{A}$ consists of row vectors $[x, y, 1]$ of all pixels in the segment and $\mathbf{D}$ is the column vector of all the depth values of the segment. The plane parameters a, b and c can then be estimated by the least-square solution.

To reduce the impact of depth estimation noises, the RANSAC scheme is further employed, which iteratively selects a random subset of the original data as hypothetical inliers.



(a)



(b)

Figure 5.4: Example of color segmentation. (a) Original image; (b) Color segmentation image.

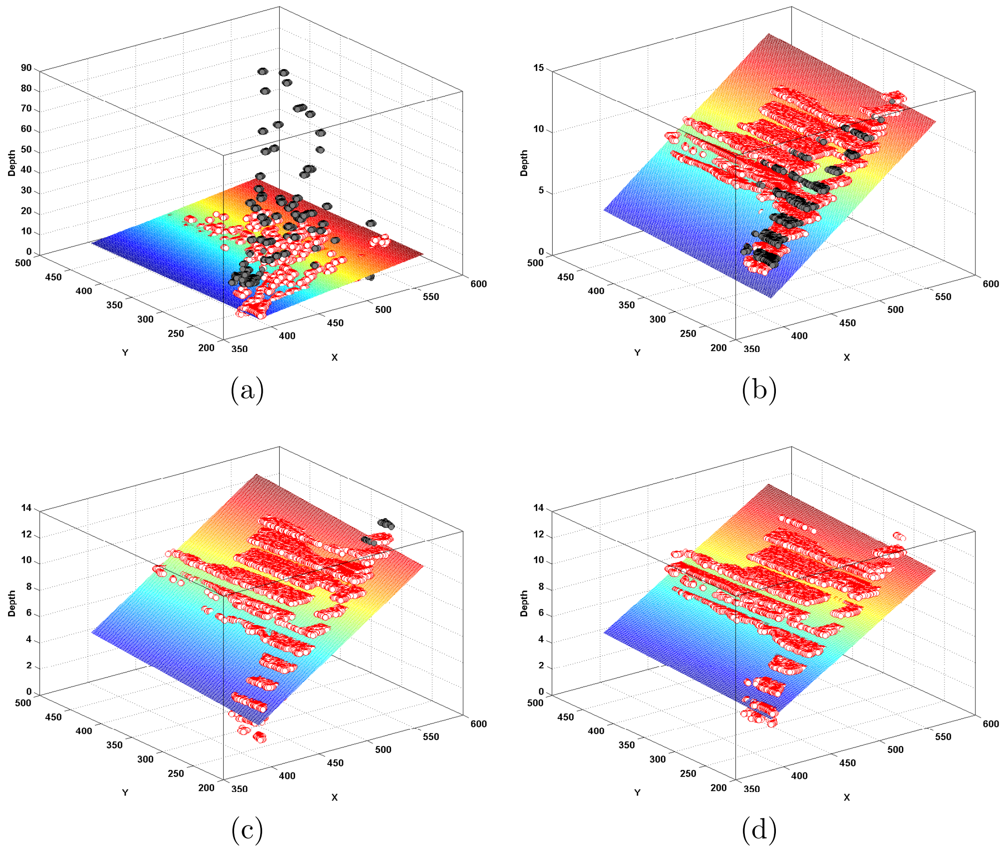
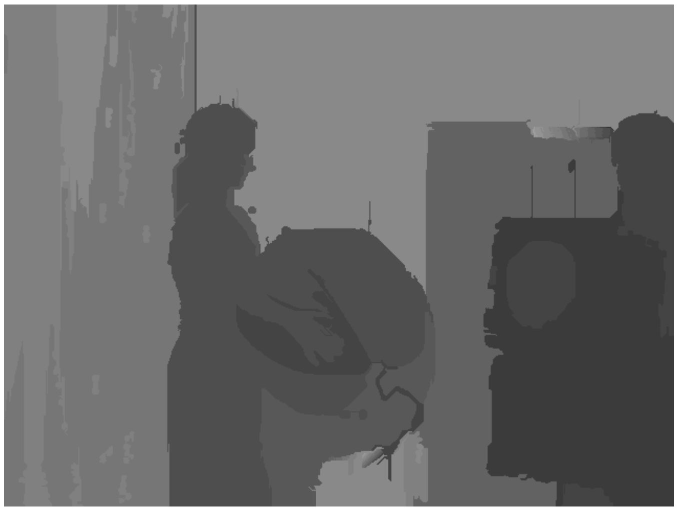


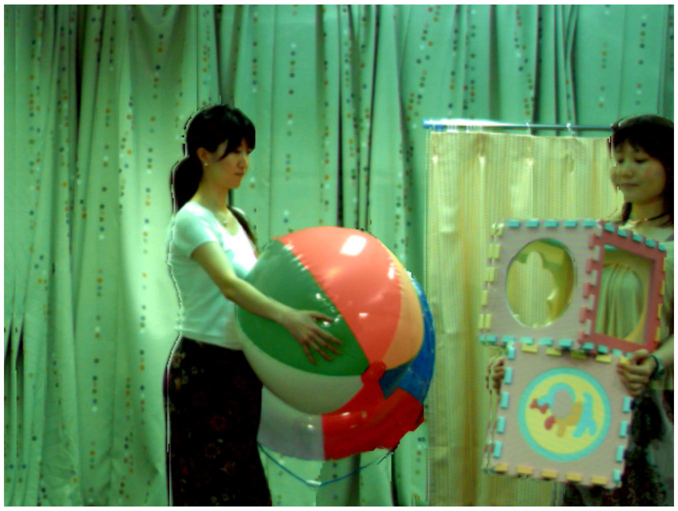
Figure 5.5: RANSAC based plane fitting: (a) Step 1; (b) Step 2; (c) Step 3; (d) Step 4. Inliers are represented by hollow red circles, and outliers are represented by solid dark circles.

The algorithm is illustrated in Fig. 5.5. First, some initial depth values are used to fit a plane. In the next iteration, the depth values within a given distance of the fitted plane are added as new inliers. The plane parameters are updated based on the new inliers. This process iterates until the plane parameter converge or the maximum iteration is reached.

Fig. 5.6 shows the depth map and view synthesis results generated after the proposed depth plane fitting. As shown in Fig. 5.6, this process is particularly useful for smoothing the depth values in the textureless regions where matching ambiguities often occur, while achieves a much better synthesis quality than the previous block-based methods. During the experiment, we found the depth plane parameters can often converge within three iterations, which does not significantly increase the complexity of the proposed algorithm.



(a)



(b)

Figure 5.6: Final result of the proposed depth estimation. (a) Depth map; (b) View synthesis result with PSNR 28.05 dB.

5.4 Experimental Results

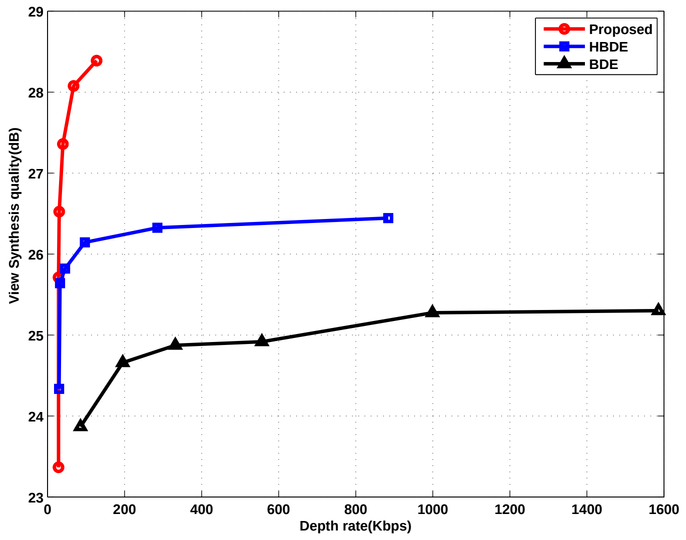
In this section, we demonstrate the performance of the proposed depth estimation method on depth coding and MVC by comparing with two representative methods: the original BDE method [83] and the hierarchical block-based depth estimation (HBDE) method [91]. Two multiview video data set, `Akko&Kayo` [76] and `Vassar` [92], are tested. We implement the three methods in the current MVC reference software JMVC [73]. The typical settings of 8-frame GOP, 32-pixel motion search window, and CABAC entropy encoding are used. The RD optimization and loop filter are enabled, whereas the rate control is turned off.

5.4.1 View Synthesis Results

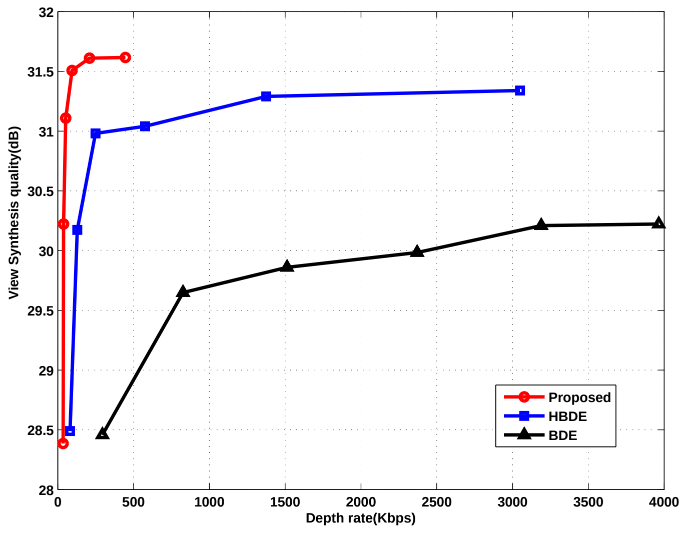
As shown in Fig. 5.6(a), the depth map generated by the proposed method is much smoother than other depth maps. It is therefore expected that our method should achieve better compression efficiency in depth map coding. To verify this, we compare the average synthesis qualities versus the depth coding bit rates of the three methods, where the depth maps are encoded based on the hierarchical B structure of JMVC. Fig. 5.7 shows the results of view 2 of two multiview sequences. To calculate the depth map of view 2, view 1 and view 3 are used as references for our method, while view 1 is used for BDE and HBDE methods. As shown in Fig. 5.7, the proposed method outperforms both BDE and HBDE methods at all bit rates, while the depth maps generated by HBDE can be compressed more efficiently than those of BDE. Specifically, for `Akko&Kayo`, the coding efficiency of the proposed method is up to 4 dB better than BDE and 2 dB better than HBDE. For `Vassar`, the improvement of our method is up to 3 dB over BDE, and 1.5 dB over HBDE. Note that both BDE and HBDE can only improve their synthesis qualities at very high rates, and it will be shown later that the huge bandwidth consumption of depth map coding in both BDE and HBDE, especially at high bit rate, is impractical for real depth-based MVC.

5.4.2 Depth Coding Results

Next, we compare the coding efficiency of the three depth estimation methods. In BDE, HBDE and the proposed method, we treat the synthesized view with depth map and the reconstructed left view as inter-view references and two temporal images derived from hierarchical B structure as temporal references. Note that although the camera matrices $\mathbf{A}_n$, $\mathbf{R}_n$ and $\mathbf{T}_n$ have to be transmitted from encoder to decoder for view synthesis in all the



(a)



(b)

Figure 5.7: View synthesis results. (a) `Akko&Kayo`; (b) `Vassar`.

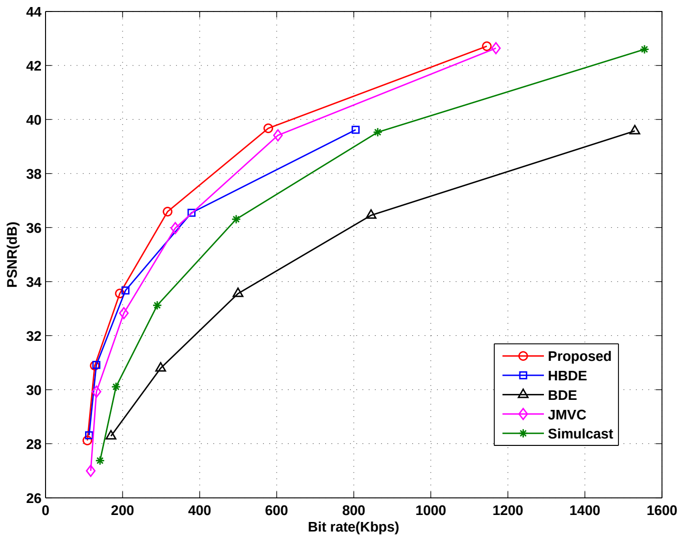
depth-based schemes, the amount of information required to describe these parameters is very small and thus the associated coding overhead is negligible.

For better comparison, the three depth-based schemes are also compared with two other MVC schemes that do not use depth information. The first one is simulcast which uses hierarchical B structure to encode each view independently. The second scheme is the current JMVC method, where the reconstructed left view is used for inter-view prediction. To achieve a fair comparison with the MVC schemes without depth information, the actual bit stream used for the depth compression in the three depth-based MVC schemes are considered when calculating the overall bit rate of the entire MVC system.

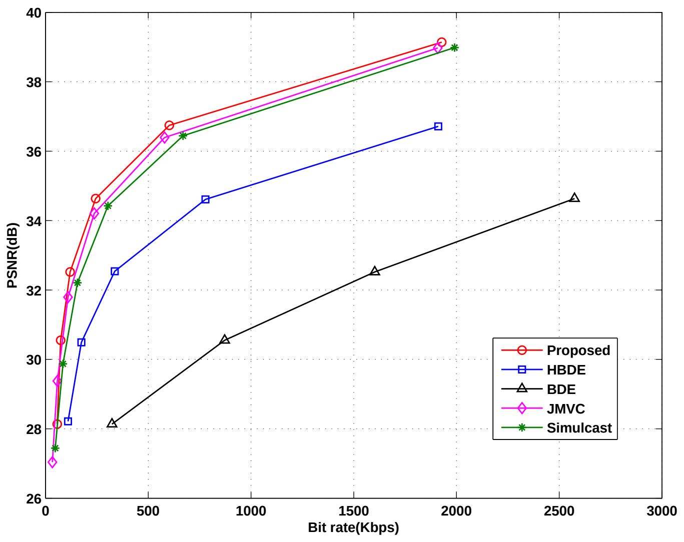
Fig. 5.8 depicts the RD curves of encoding view 2 of two multiview sequences by various schemes. It can be seen that the proposed method outperforms all other depth-based schemes. In addition, when compared with the methods without depth information, our method can also be better than JMVC and Simulcast. Especially, the proposed method can provide improvements over JMVC at all bit rates, with the maximum of 1.2 dB for `Akko&Kayo` and 0.8 dB for `Vassar`. On the other hand, due to the noisy depth maps generated by HBDE method, the coding performance of the HBDE method is worsen than JMVC for most of the bit rates. Although the HBDE method can be better than JMVC for `Akko&Kayo` at low bits, its performance drops quickly at high bit rates, because more bits are used for depth map compression. For BDE method, it can be observed that its performance is always lower than Simulcast. The reason is because the depth map compression in the BDE method can consume almost the same bandwidth as that used for encoding the texture information, which offsets the coding gain by applying the depth-based view synthesis for inter-view prediction. The results in Fig. 5.8 demonstrate the importance of depth map coding to the overall coding efficiency.

5.5 Summary

In this chapter, we develop an improved algorithm to generate a smooth and accurate depth map for DIBR-based view synthesis. For each block in the target view, the algorithm first uses epipolar geometry to find its matched block in the reference view, from which an initial depth is obtained using the triangulation method and depth projection. 3D warping is then applied to refine the depth. In addition, a SSIM and maximum likelihood-based approach is developed to fuse the depth estimations from multiple references. Finally, the depth map



(a)



(b)

Figure 5.8: Depth coding results. (a) `Akko&Kayo`; (b) `Vassar`.

is smoothed via segmentation and plane fitting. Compared to existing 3D warping-based depth estimation, the proposed algorithm can achieve up to 4 dB improvement in view synthesis, while requires much fewer bits to encode the depth map. Experimental results in depth coding show that the proposed method can outperform the H.264-based JMVC software by more than 1 dB.

Chapter 6

Delay-cognizant IMVS with Free View Synthesis

6.1 Introduction

While frame structure optimizations [63, 66, 67] for IMVS have been studied to achieve significant reduction in expected transmission rate over naive frame structures of comparable sizes, there are several shortcomings. First, the available views for a client to select are limited by the few camera-captured views pre-encoded at server, thus a view-switch could appear abrupt and unnatural to a viewer. Second, the proposed media interaction model that represents a typical client's view-switching behavior is assumed to be statistically independent in time, but it has been shown [65] that viewers exhibit temporal dependencies when switching views. Third, previous structure optimization assumes server-client communication takes place over idealized zero-delay network. In a realistic packet-switched network such as the Internet with non-negligible round trip time (RTT) delay, server's responding upon receipt of each client's requested view will mean each client's requested view-switch will suffer at least one RTT delay, hampering interactivity of the viewing experience.

In this chapter, to enrich the IMVS experience and implement it in a realistic network setting, we propose three significant improvements. First, leveraging on the recent advances in DIBR [29, 30] that enable synthesis of a virtual intermediate view between two captured views using depth information, we encode both the texture and depth maps of captured views into a *video-plus-depth* coding format [93], each at the respective optimized quantization parameter (QP) under a given view synthesis distortion constraint, into a frame structure. To enable free-viewpoint view-switching [94], *i.e.*, synthesizing virtual views from an almost continuum of viewpoints between the left-most and right-most captured views, the server transmits texture and depth maps of two nearest captured views to the client. This represents a major improvement in interactive viewing experience over previous IMVS schemes.

Second, given free viewpoint selection is available to clients, we construct a more general media interaction model with finite memory that more accurately captures user behavior in view selection. Third, we consider the non-negligible transmission delay in the design of the frame structure and transmission schedule. As a result, frames in the structure corresponding to possible playback RTT can be additionally transmitted, so that a user can experience zero-delay view-switching.

We formalize the joint optimization of the frame encoding structure, transmission schedule, and QPs of the texture and depth maps, and propose an iterative algorithm to achieve fast and near-optimal solutions. Convergence of the proposed algorithm is also proved. Experimental results show that our proposed rate allocation method reduces transmission rate over fixed texture/depth rate allocation methods by up to 38%. In addition, for the same storage, transmission rate of the frame structure generated by our proposed algorithm can be up to 55% lower than that of I-frame-only structures, and 27% lower than that of the structure without M-frames.

The outline of the chapter is as follows. We first discuss the IMVS system, source model of encoding multiview video, our generalized media interaction model with memory for view-switching and network delay model in Sec. 6.2. In Sec. 6.3, we formulate the problem of finding the optimal frame structure, transmission schedule and QPs for encoding texture and depth maps in a network-delay-cognizant manner. In Sec. 6.4, we develop an iterative optimization algorithm to efficiently find a solution for the proposed IMVS problem. Simulation results and conclusion are given in Sec. 6.5 and Sec. 6.6, respectively.

6.2 System and Media Interaction Model

To facilitate understanding of our contributions in this chapter, we first overview the system model for IMVS. We then describe the source model for coded multiview videos, and DIBR

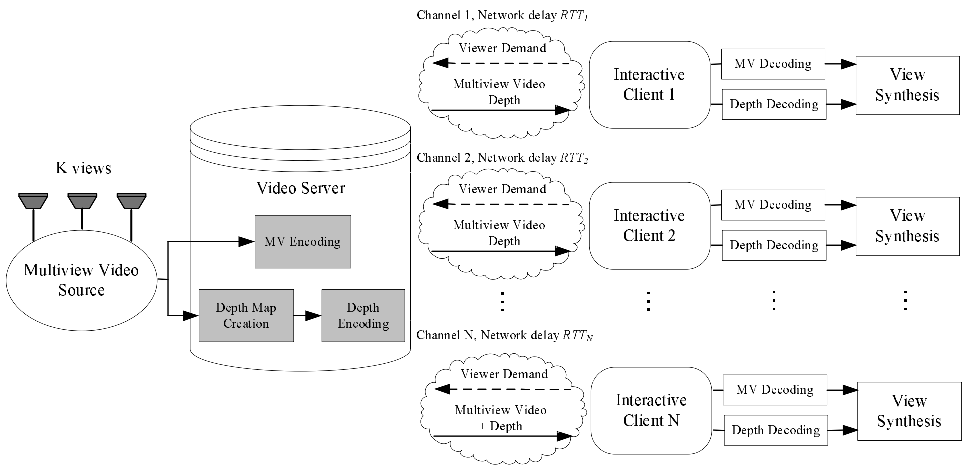


Figure 6.1: Overview of the proposed IMVS system

used for synthesizing virtual views using coded texture and depth maps of neighboring views. We then discuss a general view-switching model of finite memory that captures user's behavior in selecting possibly virtual views. Finally, we discuss our network model that considers the RTT delays between streaming server and clients.

6.2.1 System Model for IMVS

The system model we consider for IMVS is shown in Fig. 6.1, where a *multiview video source* captures time-synchronized videos of a 3D scene from K evenly spaced, horizontally shifted cameras in a 1D array. A *video server* sequentially grabs captured texture and depth maps from the multiview video source[1], and encodes the texture and depth maps separately into the same optimized frame structure $\mathcal{T}$ of I-, P- and M-frames, at their respective optimized QPs. In other words, the same permutation of I-, P- and M-frames used to encode texture maps at one QP, will be used also to encode depth maps using a different QP separately. The video server stores a single data structure $\mathcal{T}$, using which the server can provide IMVS service for multiple clients. Another approach of live encoding an unique view traversal tailor-made for each streaming client's interactivity is computationally prohibitive if the number of clients is large.

A client can request a view-switch every Δ frames, where the requested view can be a captured view or an intermediate virtual view between two captured views. The availability

[1]Depth maps can be estimated from texture maps using stereo-matching algorithms [32], or captured directly using time-of-flight cameras [10].

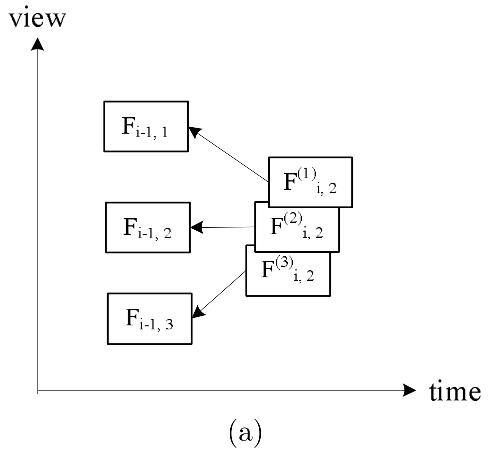
of a large number of virtual views—an almost continuum of views between left-most and right-most captured views—enables finer grain view-switches compared to previous IMVS work [63, 66, 5], where the available views were limited by the number of capturing cameras, and each view-switch was an abrupt jump from one camera view to another. To facilitate synthesis of a virtual view at the client side, the server always transmits both texture and depth maps of the closest left and right captured views. The client then interpolates the requested virtual view using received texture and depth maps via DIBR, as discussed in Chapter 2.1.3. Further, we assume I-frames are inserted every Δ' frames, $\Delta \ll \Delta'$, for all K captured views for some pre-defined level of random access.

Since the same optimized frame structure is used to encode both texture and depth maps of multiview video source, for ease of discussion, we will use the term *picture* to denote both the texture and depth maps of the corresponding captured image, and the term *frame* to denote the specific coded version of texture and depth maps of an image. Further, given view-switch period[2] Δ, we use $F^o_{i,j}$ and $F_{i,j}$ to denote a picture and a frame of view j at *view-switch instant* $i\Delta$, *i.e.*, the time at which a client selects her i-th view-switch location.

6.2.2 Multiview Video Source Model

A picture can be coded as an intra-coded I-frame with no predictor, a differentially coded P-frame with a single predictor, or a conceptual M-frame with multiple predictors known at encoding time. I-frame is used for random access. For view-switching, either redundant P-frames or M-frames are used. Redundant P-frames mean one differentially coded P-frame is constructed for each potential predictor (last frame in a decoding path from which a view-switch is possible). M-frame, on the other hand, has a single frame representation for multiple potential predictors; reconstruction property of M-frame guarantees that the exact same frame can be correctly decoded no matter which one of a set of predictor frames known at encoding time is actually available at the decoder's buffer at stream time. Redundant P-frames offer the lowest transmission rate possible while increasing the storage required as the number of decoding paths multiplies over time. An M-frame has a single frame representation and hence smaller storage, but at a higher transmission rate than P-frame.

[2]In more general case of $\Delta > 1$, a picture $F^o_{i,j}$ represents Δ consecutive pictures of view j from time $i\Delta$ to time $(i+1)\Delta - 1$, and a frame $F_{i,j}$ represents Δ consecutive actual frames of view j, including a carefully chosen I-, P- or M-frame determined by our optimization algorithm followed by $\Delta - 1$ consecutive P-frames predicted from the same view.



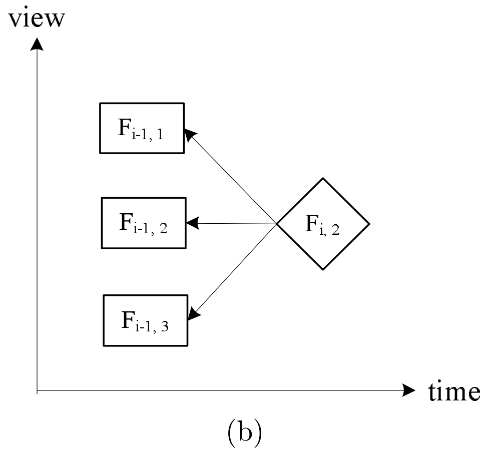


Figure 6.2: Examples of (a) redundant P-frames and (b) M-frame.

Fig. 6.2 is an example to show the tradeoff between transmission rate and storage for redundant P-frames and M-frame from three different predictors. The redundant P-frames in Fig. 6.2(a) need three different coded versions of picture $F^o_{i,2}$, one for each of three different predictors $F_{i-1,1}$, $F_{i-1,2}$ and $F_{i-1,3}$, whereas in Fig. 6.2(b) only one M-frame is needed to get the same coded version no matter which of the three predictors is available at the decoder.

An M-frame can be implemented using one of many available techniques such as SP-frames in H.264 [69] and DSC-frames [68, 70]. In this thesis, we implement an M-frame using DSC [68], due to its demonstrably superior coding performance over SP-frames. We overview the encoding of a DSC frame as follows. First, motion information from each of the predictor frames is encoded. Then, transform coefficients of the motion residuals in discrete cosine transform (DCT) domain from each prediction are compared. Because most significant bits of the transform coefficients are likely to be the same for all residuals, only the least significant bit bit-planes that are different among the residuals need to be encoded using low-density parity check codes (LDPC). By encoding multiple motion information and differing bit-planes, the exact same frame can be recovered no matter which predictor frame is available at decoder's buffer. By exploiting correlation between predictor frames and the target, DSC frame has much smaller size than the independently coded I-frame.

6.2.3 Media Interaction Model

Without loss of generality, we first denote K evenly spaced captured views by $1, \ldots, K$. Between every pair of adjacent captured views i and $i+1$, we in addition define a set of

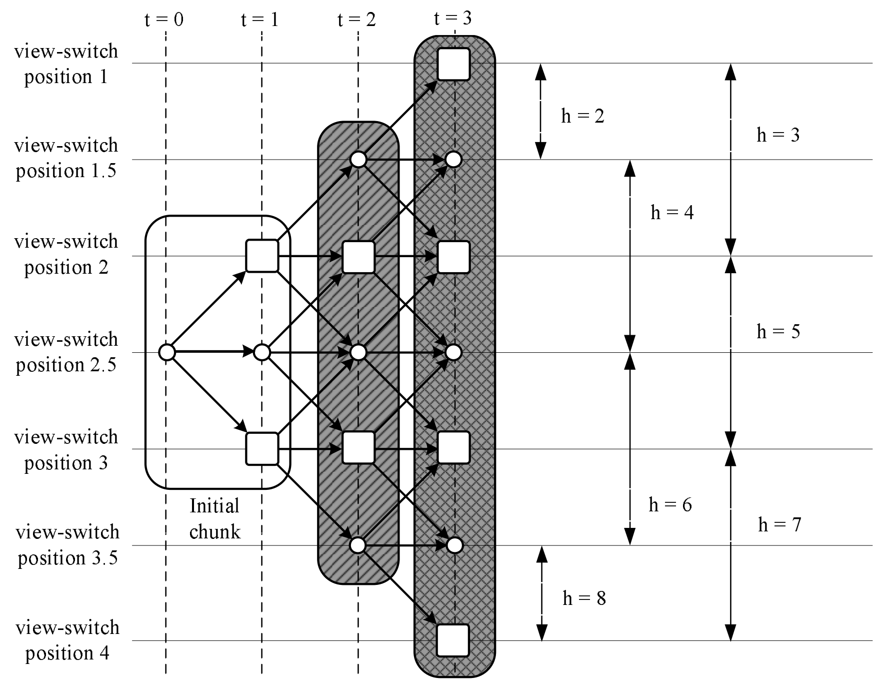


Figure 6.3: Example of progressive view-switch for $K = 4$ captured views (rectangles) with $K' = 1$ intermediate view (circles) between two captured views ($d = 0.5$), view difference bound $L = 1$, initial view $v^0 = 2.5$, view-switching period $\Delta = 1$ and $RTT = \Delta - \epsilon$. View-switch positions in the shadeless box and shaded boxes with different patterns represent the ones covered by the initial chunk and structure slices at time 2 and 3 respectively. Each double-end arrow delimits the range of possible view-switches covered by one structure slice after receiving a view-switch coordinate feedback h from client.

K' evenly spaced virtual view positions that can also be requested by clients, *i.e.*, $i + \frac{j}{K'+1}$, $j \in \{1, \ldots, K'\}$, separated by *view spacing* $d = 1/(K' + 1)$. The total number of views available for client's selection is hence expanded to $K'(K-1)+K$. Fig. 6.3 shows an example of multiview sequence where $K = 4$ and $K' = 1$ ($d = 0.5$). Note that all available discrete *view-switch positions* (virtual and captured) available for client's selection are multiples of d. In the sequel, we will say that a view-switch position $v = kd$, $k \in \mathcal{Z}^+$, $1/d \leq k \leq K/d$, has *view coordinate* k, where k is view-switch position v expressed in multiples of view spacing d.

We design an interaction model to allow a client to periodically request a view-switch every Δ frames from view-switch position v to another view-switch position v', where the difference $|v' - v|$ is no larger than Ld, $L \in \mathcal{Z}^+$, where the pre-defined *View difference bound* L limits the speed of view transition.

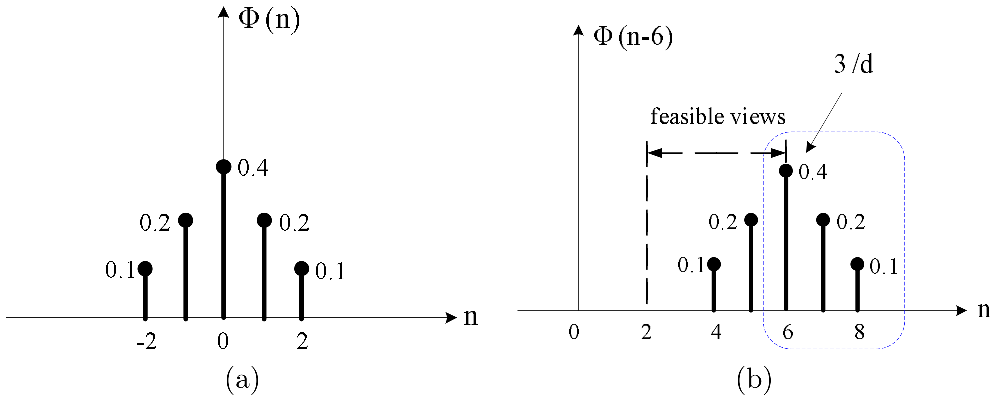


Figure 6.4: Example of view-switch probability function for $K = 3$ captured views with $K' = 1$ intermediate view between two neighboring captured views ($d = 0.5$). (a) original $\Phi(n)$; (b) shifted function $\Phi(n-6)$.

To optimize multiview video frame structure at encoding time without knowledge of clients' eventual chosen view trajectories at stream time, we propose the following probabilistic model to capture the view-switching trend of a typical client. Suppose a client is watching view coordinate k at view-switch instant $i\Delta$, after watching view coordinate k' at instant $(i-1)\Delta$. The probability[3] that she will select view coordinate l at instant $(i+1)\Delta$ is $\Omega_{k',k}(l)$, $l \in \{\max(1/d, k-L), \dots, \min(K/d, k+L)\}$:

$$\Omega_{k',k}(l) = \begin{cases} \Phi(l - (2k - k')), & \max(1/d, k-L) < l < \min(K/d, k+L) \\ \sum_{n=l}^{\infty} \Phi(n - (2k - k')), & l = \min(K/d, k+L) \\ \sum_{n=-\infty}^{l} \Phi(n - (2k - k')), & l = \max(1/d, k-L) \end{cases} \tag{6.1}$$

where $\Phi(n)$ is a symmetric *view-switching probability function* centered at zero; see Fig. 6.4(a) for an example. In words, (6.1) states that the probability $\Omega_{k',k}(l)$ that a client selects view coordinate l depends on both the current view coordinate k *and* previous selected coordinate k'; the probability is the highest at position $k + (k - k')$ where the client continues in view-switch direction $k - k'$. If l is a boundary coordinate, 1 or K, or at the view difference bound $k \pm L$, then the probability $\Omega_{k',k}(l)$ needs to sum over probabilities in view-switching

[3]Given all available view-switch positions (captured and virtual) for client's selection are integer multiples of view spacing d, we can define the view-switch probability function $\Omega_{k',k}(l)$ in discrete domain, where k', k and l are all view coordinates.

probability function $\Phi(n - (2k - k'))$ that fall outside the feasible views as well, as shown in Fig. 6.4(b), where the right-most boundary view is requested given $K = 3$ and $K' = 1$, *i.e.*, $l = 3/d$. This Markovian view-switching model with memory has been shown experimentally to be more realistic [4] than previous memoryless models [63, 66, 67]. However, the proposed view-switching model differs from that in [4] in two aspects. First, in our view-switching model, the current view-switch position is only dependent on the positions that the client selected for the last two view-switches instead of three previous view-switch positions represented by the complex Kalman filter as considered in [4]. Second, as shown in [65], a client is more likely to switch in the direction as the last view-switch. To model this effect, our model uses the information about the span of the last view transition to predict the future view-switch position. This is in contrast to the view-switch model in [4] where this information is not considered.

6.2.4 Network Delay Model

Round trip time (RTT) delay is the time required for a packet to travel from a client to the server and back. In our IMVS scenario, RTT delay specifically represents the minimum server-client interaction delay experienced by a client from the time she sends a view-switch request, to the time the effected video due to the request is received. Here, we assume there are different RTTs between the video server and different clients, though RTT of each server-client pair remains constant (each of server-to-client and client-to-server transmission takes exactly half of RTT) once video streaming starts. In addition, we assume all RTTs do not exceed an upper-bound $RTT_{\max}$. There is much work in the literature in estimating RTT in typical packet-switched networks [95, 96], but is outside the scope of this thesis. We will simply assume the probability density function (PDF) of RTT, $\psi(x)$, is known a priori at video encoding time.

6.3 Problem Formulation

Having described the proposed IMVS system and models in Sec. 6.2, we now formulate the IMVS problem as an optimization problem: given pre-defined storage and distortion constraints, design an optimal frame structure and associated schedule, and select optimal QPs for texture and depth map coding, that minimize the expected server transmission rate, while providing clients with zero-delay view-switching interactivity in IMVS. In Sec. 6.3.1,

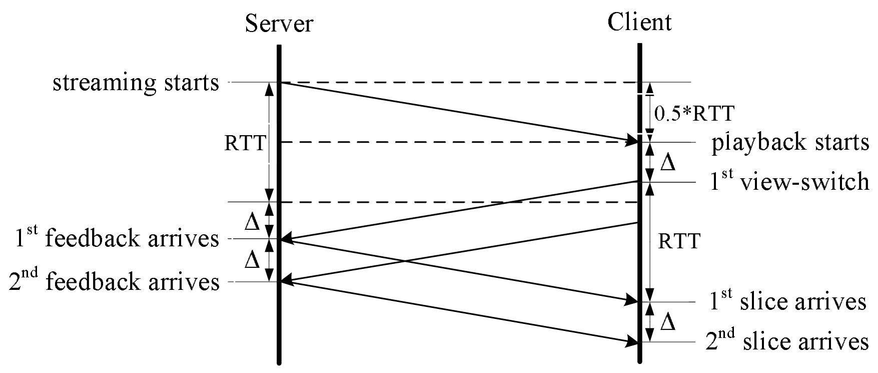


Figure 6.5: Timing diagram during server-client communication.

we first develop a network-delay-cognizant transmission protocol for transmitting frames in a coding structure for IMVS, so that each client can enjoy zero-delay view-switching given her unique server-client RTT. We then provide definitions of optimization variables, search space, constraints and objective in Sec. 6.3.2. Finally, we formally define the IMVS optimization problem in Sec. 6.3.3.

6.3.1 Network-delay-cognizant Transmission Protocol

Previous IMVS works [63, 66, 67] do not properly address the problem of network delay; hence a view-switch request from a client will suffer at least one RTT delay in addition to the system's inherent Δ-frame view-switch interval[4]. In this section, we develop a transmission protocol for network-delay-cognizant view-switching, so that a client can play back the video in time and perceive no additional view-switching delay (beyond the system's Δ-frame view-switch interval), even when RTT is non-negligible. The key idea is to send additional data to cover all possible view-switch positions to be requested by a client one RTT into the future beyond the requested view.

Following the illustration in Fig. 6.5, we first discuss timing events during server-client communication in IMVS system assuming constant transmission delay (as discussed in Sec. 6.2.4). The server first transmits an *initial chunk* of coded multiview data to the client, arriving at the client $\frac{1}{2}RTT$ time later. At time 0[5], the client starts playback, and

[4]View-switch interval Δ for IMVS systems can be set very small (on the order of every 3 to 5 frames), and hence an additional RTT delay on the Internet of up to hundreds of milliseconds can be detrimental to the interactive multiview video experience.

[5]For ease of discussion, we set the origin of time at the point when client starts playback of video.

makes her first view-switch Δ-frame time later. Her first view-switch decision (feedback) is transmitted immediately after the view-switch, and arrives at server at time $\frac{1}{2}RTT + \Delta$. Responding to the client's first feedback, server immediately sends a *structure slice*, arriving at the client $\frac{1}{2}RTT$ time later, or RTT time after the client transmitted her feedback. More generally then, the client sends feedbacks in interval of Δ-frame time, and in response, server sends a structure slice corresponding to each received feedback every Δ-frame time. We assume there are no packet losses during packet transmission.

Notice that from the time the client starts playback to the time the first structure slice is received from server, $\Delta + RTT$ time has elapsed. Therefore, before the arrival of the first structure slice, the number of view-switches, δ, that the initial chunk must enable is

$$\delta = \left\lfloor \frac{\Delta + RTT}{\Delta} \right\rfloor \tag{6.2}$$

For simplicity, we assume that IMVS session starts from a known initial position v^o with view coordinate k^o, *i.e.*, $v^o = k^o d$ and $k^o \in \mathcal{Z}^+$, $1/d \leq k^o \leq K/d$. Given each subsequent view-switch can maximally alter view coordinate by $\pm L$, initial chunk must contain data enabling view-switches to view coordinates $\mathcal{V}_i = \{k \mid \max(1/d, k^o - iL) \leq k \leq \min(K/d, k^o + iL), k \in \mathcal{Z}^+\}$ at view-switch instants $(i\Delta)$'s, where $0 \leq i \leq \delta$.

Because subsequent structure slices arrive every Δ-frame time, each structure slice only needs to enable one more view-switch for the client to continue video playback in time and enjoy zero-delay view-switching. Notice that because each structure slice arrives at the client RTT time after the client sent her view-switch feedback, the view-switch enabled by the structure slice corresponding to the client's feedback sent at instant $t = i\Delta$ is the first view-switch after time $t + RTT$, *i.e.*, view-switch at instant $(i + \delta)\Delta$. In other words, given client's view coordinate selection h at instant $i\Delta$, the span of view-switch coordinates $\mathcal{V}_{i+\delta}$ that a structure slice must cover for the view-switches at instant $(i + \delta)\Delta$, is $\mathcal{V}_{i+\delta} = \{k \mid \max(1/d, h - \delta L) \leq k \leq \min(K/d, h + \delta L), k \in \mathcal{Z}^+\}$.

This protocol—transmitting multiple views for the sake of client's selection of a single view in the future—is in stark contrast with the protocol in [63, 66, 67], where only one single view is transmitted corresponding to each client's request. Fig. 6.3 illustrates a view-switching example for $K = 4$, $K' = 1$, $L = 1$, $v^o = 2.5$, $\Delta = 1$ and $RTT = \Delta - \epsilon$ for small $\epsilon > 0$. The initial chunk contains only enough multiview data to enable $\delta = 1$ view-switch, spanning view-switch coordinates $\mathcal{V}_1 = \{4, 5, 6\}$. If the client first selects view-switch coordinate $h = 4$ at time 1, then the first structure slice must span view-switch coordinates

Table 6.1: Summary of notations for IMVS problem formulation

Notations	Description
K, K'	number of captured views, number of virtual views between two neighboring captured views.
L, v^o	view difference bound, starting view-switch position.
RTT, δ	RTT delay, number of view-switches that a structure slice covers into future after receiving a client feedback.
$\mathcal{T}$, $\mathbf{Q} = [Q_t, Q_d]^T$	frame structure, QPs for encoding texture and depth map.
$B(\mathcal{T}, \mathbf{Q})$, $C(\mathcal{T}, \mathbf{Q})$, $D(\mathbf{Q})$	storage cost, transmission cost, distortion cost of a frame structure $\mathcal{T}$ and QPs $\mathbf{Q}$.
$F^o_{i,j}$, $F_{i,j}$	original picture, coded frame of view-switch instant $i\Delta$ and view j.
$I_{i,j}$, $P_{i,j}$, $M_{i,j}$	I-frame, P-frame, M-frame of view-switch instant $i\Delta$ and view j.
$\Xi_i(\delta)$, $c(\Xi_i(\delta))$	the set of frames, the center view coordinate for decoding at view-switch instant $i\Delta$.
$p(\Xi_i(\delta))$, $q(F_{i,j}, \delta)$	transmission probability of a slice $\Xi_i(\delta)$, a frame $F_{i,j}$, given δ.
$\psi(x)$, $\Psi(\delta)$	probability density function of RTT, probability mass function of δ.
$t^\delta(\mathcal{T}, G(\delta), \mathbf{Q})$	transmission rate of a frame structure $\mathcal{T}$ and QPs $\mathbf{Q}$, given schedule $G(\delta)$.
$D^c_j(Q_t)$, $D^s_k(\mathbf{Q})$	average distortion of frames of captured view j, virtual view k, given QPs $\mathbf{Q}$.

$\mathcal{V}_{1+\delta} = \{3, 4, 5\}$. Instead, if the client first selects view-switch coordinate $h = 6$, then the corresponding slice must span view-switch coordinates $\mathcal{V}_{1+\delta} = \{5, 6, 7\}$.

6.3.2 Definitions for IMVS Optimization

Before formally defining the IMVS optimization problem, we first define optimization variables (frame structure, associated transmission schedules and QPs), and storage, transmission and distortion costs corresponding to a set of variables. See Table 6.1 for a summary of notations.

Redundant Frame Structure

Given a multiview sequence of K captured views with K' intermediate views between each pair of neighboring captured view, One can construct a *redundant frame structure* $\mathcal{T}$, comprised of I-, P- and M-frames, denoted as $I_{i,j}$'s, $P_{i,j}$'s and $M_{i,j}$'s respectively, to represent the captured multiview video frames at view-switch instant $i\Delta$'s and view j's for IMVS. Each frame not located at view-switch instants (not shown in our graphical model) is a P-frame predicted from a frame of the same view and previous time instant. Note that we do not specify if the predictions for P-frames $P_{i,j}$'s a motion- or disparity-compensated; our abstraction only aims to capture the dependencies among frames, and not the particular

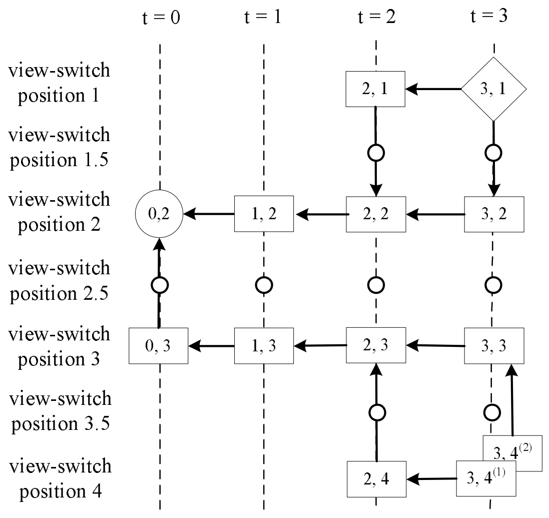


Figure 6.6: Example frame structure for $K = 4$ captured views with $K' = 1$ intermediate view (small circles) between two neighboring captured views (view spacing $d = 0.5$), view difference bound $L = 1$, initial view $v^0 = 2.5$, view-switching period $\Delta = 1$ and $RTT = \Delta - \epsilon$. I-, P- and DSC-frames are represented by large circles, rectangles and diamonds, respectively.

encoding tools used. Note also that while we already discussed one concrete DSC implementation of M-frame in Sec. 6.2.2, our abstraction and subsequent optimization can apply more generally to any implementation of M-frame. Fig. 6.6 shows one example frame structure for multiview sequence in Fig. 6.3.

A frame structure $\mathcal{T}$ forms a *directed acyclic graph* starting with an I-frame if initial view-switch position is a captured view, or an I-frame and a P-frame predicted from the I-frame if initial view-switch position is a virtual view, as starting nodes. In Fig. 6.6, I-frames $I_{0,2}$ and P-frame $P_{0,3}$ are two starting nodes of structure $\mathcal{T}$ for synthesizing virtual view 2.5. $\mathcal{T}$ is redundant in the sense that an original picture $F^o_{i,j}$ can be represented by more than one frame $F_{i,j}$. In Fig. 6.6, original picture $F^o_{3,4}$ is represented by two P-frames, $P^{(1)}_{3,4}$ and $P^{(2)}_{3,4}$, each encoded using a different predictor, $P_{2,4}$ and $P_{3,3}$, respectively. Depending on which predictor is available at decoder during stream time, different coded frames $F_{i,j}$'s can be transmitted to enable correct decoding and (slightly different) reconstruction of original picture $F^o_{i,j}$. This is done to lower transmission rate by exploiting correlation between the requested picture and frames in the decoder buffer, and to avoid coding drift [63].

Structure Slice

As discussed in Sec. 6.3.1, depending on the view-switch coordinate h selected by client at view-switch instant $(i-\delta)\Delta$, a set of frames of different captured viewpoints will be transmitted for possible decoding at view-switch instant $i\Delta$. Given δ, we define *structure slice* $\Xi_i(\delta)$, with *center coordinate* $c(\Xi_i(\delta))$, as a set of frames to enable selection of view-switch coordinates in span $\{k \mid \max(1/d, c(\Xi_i(\delta))-\delta L) \leq k \leq \min(K/d, c(\Xi_i(\delta))+\delta L), k \in \mathcal{Z}^+\}$ at view-switch instant $i\Delta$. Center coordinate $c(\Xi_i(\delta))$ is the client's selected view coordinate h at view-switch instant $(i-\delta)\Delta$.

Consider the example in Fig. 6.6, where initial chunk contains frames $I_{0,2}$, $P_{0,3}$, $P_{1,2}$ and $P_{1,3}$ to cover view-switches to positions 2, 2.5 and 3 at time 1. If the client selects view-switch coordinate $h=4$ (view 2) at time 1, then the corresponding structure slice transmitted is $\Xi_2^{(1)}(1) = \{P_{2,1}, P_{2,2}, P_{2,3}\}$ with $c(\Xi_2^{(1)}(1)) = 4$, to cover possible view-switches to positions 1.5, 2 and 2.5 at time 2. Instead, if client remains in coordinate $h=5$ (view 2.5) at time 1, then the structure slice $\Xi_2^{(2)}(1) = \{P_{2,2}, P_{2,3}\}$ will be sent to decoder with $c(\Xi_2^{(2)}(1)) = 5$, for the possible switches to positions 2, 2.5 and 3 at time 2. Finally, if client switches to position 3 at time 1, to enable possible view-switches to position 2.5, 3 and 3.5, the structure slice sent to decoder will be $\Xi_2^{(3)} = \{P_{2,2}, P_{2,3}, P_{2,4}\}$ with $c(\Xi_2^{(3)}) = 3$. Notice that different slices can contain the same frames, and can also contain different number of frames.

Transmission Schedule

Which slice $\Xi_i(\delta)$ of structure $\mathcal{T}$ is transmitted for view-switch instant $i\Delta$ depends on slice $\Xi_{i-1}(\delta)$ transmitted previously (for differential coding), and client's selected view-switch coordinate h at view-switch instant $(i-\delta)\Delta$. The center coordinate $c(\Xi_i)$ of Ξ_i will necessarily be h, and the view span of slice Ξ_i will be $\mathcal{V}_i = \{k \mid \max(1/d, h-\delta L) \leq k \leq \min(K/d, h+\delta L), k \in \mathcal{Z}^+\}$, so that all reachable view coordinates at instant $i\Delta$, given client's selection of view coordinate h at instant $(i-\delta)\Delta$, are available for client's selection.

We can formalize the association among $\Xi_{i-1}(\delta)$, h and $\Xi_i(\delta)$ via a *transmission schedule* $G(\delta)$. More precisely, $G(\delta)$ dictates which structure slice $\Xi_i(\delta)$ will be transmitted for client's selection at view-switch instant $i\Delta$, given previous transmitted slice $\Xi_{i-1}(\delta)$ and client's selected view-switch coordinate h at view-switch instant $(i-\delta)\Delta$:

$$G(\delta) : (\Xi_{i-1}(\delta), h) \Rightarrow \Xi_i(\delta), \quad \max(1/d, c(\Xi_{i-1}(\delta)) - L) \leq h \leq \min(K/d, c(\Xi_{i-1}(\delta)) + L) \tag{6.3}$$

where center coordinate of $\Xi_i(\delta)$ is $c(\Xi_i(\delta)) = h$. In what follows, we denote a scheduled transmission from slice $\Xi_{i-1}(\delta)$ to slice $\Xi_i(\delta)$, with client's selected view-switch coordinate h at instant $(i-\delta)\Delta$, as $(\Xi_{i-1}(\delta), h) \overset{G(\delta)}{\Rightarrow} \Xi_i(\delta)$.

Note that for a given structure $\mathcal{T}$ and slice $\Xi_{i-1}(\delta)$ available for decoding at view-switch instant $(i-1)\Delta$, if client selects view coordinate h at view-switch instant $(i-\delta)\Delta$, there may exist different decodable slices $\Xi_i(\delta)$'s, and hence different transmission schedules $G(\delta)$'s, that enable all reachable view-switch coordinates $\mathcal{V}_i$ at instant $i\Delta$. For example, in Fig. 6.6, we assume the slice $\Xi_2 = \{P_{2,2}, P_{2,3}\}$ is available at decoder for view-switches at time 2, with center view coordinate $c(\Xi_2) = 5$. If the client selects view coordinate 6 at time 2, then the corresponding span of reachable view coordinates at time 3, $\mathcal{V}_3 = \{5, 6, 7\}$. Thus, there are two possible schedules, resulting in $\Xi_3^{(1)} = \{P_{3,2}, P_{3,3}, P_{3,4}^{(2)}\}$ and $\Xi_3^{(2)} = \{P_{3,2}, P_{3,3}, P_{2,4}, P_{3,4}^{(1)}\}$ transmitted respectively. Obviously, the schedule of $\Xi_3^{(1)}$ which has a smaller size of transmitted frames than $\Xi_3^{(2)}$ should be selected as optimal schedule. Correspondingly, our optimization will hence consider not just optimal structure $\mathcal{T}$, but also optimal schedule $G(\delta)$ for the chosen structure $\mathcal{T}$.

Feasible Structure Space

Based on the above discussion, we can define a *feasible frame structure* $\mathcal{T}$ given δ as one where every reachable view-switch coordinate, as constrained by the media interaction model (Sec. 6.2.3), can be requested by a client every Δ-frame interval and be executed with zero-delay using $\mathcal{T}$. Mathematically, we say that $\mathcal{T}$ is feasible given δ if there exists at least one *feasible schedule* $G(\delta)$, such that each sequence of client's permissible selection of view-switch coordinates, $h_1, h_2, \ldots$, will lead to a corresponding scheduled transmission of decodable slices $\Xi_{i+\delta}(\delta), \Xi_{i+1+\delta}(\delta), \ldots$, such that center coordinate and view span of each slice $\Xi_{i+\delta}(\delta)$ are $c(\Xi_{i+\delta}(\delta)) = h_i$ and $\mathcal{V}_{i+\delta} = \{k \mid \max(1/d, h_i - \delta L) \leq k \leq \min(K/d, h_i + \delta L), k \in \mathcal{Z}^+\}$, respectively. Center coordinates and view spans of slices defined above ensure all reachable view-switch coordinates can be selected by client at instants $(i+\delta)\Delta$, $(i+1+\delta)\Delta$, etc.

More generally, RTT between server and client can take on different values resulting in different δ's. In what follows, we define *feasible space* Θ as the set of all feasible frame structures $\mathcal{T}$'s, where a feasible structure $\mathcal{T}$ is one where there exists at least one feasible schedule $G(\delta)$ for each possible δ.

Structure Slice Probability and Frame Transmission Probability

To properly define transmission cost, we first define *structure slice probability* $p(\Xi_i(\delta))$ as the probability that structure slice $\Xi_i(\delta)$ for decoding at instant $i\Delta$ is transmitted, given schedule $G(\delta)$. Considering the structure slices $\Xi_i(\delta)$'s in the initial chunk, where $0 \leq i \leq \delta$, are always sent to client, this probability could be computed recursively using view transition probability $\Omega_{k',k}(l)$:

$$p(\Xi_i(\delta)) = \begin{cases} 1, & 0 \leq i \leq \delta \\ \displaystyle\sum_{\Xi_{i-1}(\delta) \in \mathcal{G}} p(\Xi_{i-1}(\delta)) \sum_{c'} \Omega_{c',c(\Xi_{i-1}(\delta))}(c(\Xi_i(\delta))), & i > \delta \end{cases} \tag{6.4}$$

where $\mathcal{G} = \{\Xi_{i-1}(\delta) \mid (\Xi_{i-1}(\delta), c(\Xi_i(\delta))) \overset{G(\delta)}{\Rightarrow} \Xi_i(\delta)\}$. In words, (6.4) states that $p(\Xi_i(\delta))$ is the sum of probability of each slice $\Xi_{i-1}(\delta)$ switching to slice $\Xi_i(\delta)$, scaled by the slice probability of $\Xi_{i-1}(\delta)$ itself, $p(\Xi_{i-1}(\delta))$, given schedule $G(\delta)$ dictates slice transmission in frame structure $\mathcal{T}$.

Further, we define *frame transmission probability* $q(F_{i,j}, \delta)$ as the probability that a frame $F_{i,j}$ is transmitted from server to client, which can be calculated using the defined structure slice probability (6.4):

$$q(F_{i,j}, \delta) = \sum_{\Xi_i(\delta) | F_{i,j} \in \Xi_i(\delta)} p(\Xi_i(\delta)) \tag{6.5}$$

In words, the transmission probability of a frame $F_{i,j}$ is the sum of probabilities of slices $\Xi_i(\delta)$'s that include $F_{i,j}$.

Storage Cost

For a given frame structure $\mathcal{T}$ and the associated QPs for texture and depth images, Q_t and Q_d, we can define the corresponding *storage cost* by simply adding up the sizes of all the frames $F_{i,j}$'s in $\mathcal{T}$, *i.e.*,

$$B(\mathcal{T}, \mathbf{Q}) = \sum_{F_{i,j} \in \mathcal{T}} |F_{i,j}(\mathbf{Q})| = \sum_{F_{i,j} \in \mathcal{T}} \Big(|F_{i,j}^t(Q_t)| + |F_{i,j}^d(Q_d)| \Big) \tag{6.6}$$

where $\mathbf{Q}$ is the pair of QPs for texture and depth maps $\mathbf{Q} = [Q_t, Q_d]^T$, $|F_{i,j}|$ is the size of frame $F_{i,j}$ which depend on the specific QPs $\mathbf{Q}$, $F_{i,j}^t$ and $F_{i,j}^d$ denote the texture and depth maps of frame $F_{i,j}$ respectively.

Transmission Cost

Given a frame structure $\mathcal{T}$ and the associated QPs $\mathbf{Q}$, we can define the corresponding *transmission cost*. First, given the relationship between δ and RTT in (6.2), it is straightforward to see that the same transmission schedule $G(\delta)$ for a given frame structure $\mathcal{T}$ can be applicable to all RTT's, $(\delta-1)\Delta \leq RTT < \delta\Delta$, *i.e.*, the same slice $\Xi_i(\delta)$ of structure $\mathcal{T}$ will be transmitted for each view-switch instant $i\Delta$. Therefore, to facilitate the definition of transmission cost, we map the PDF of RTT, $\psi(x)$, into a discrete probability mass function (PMF) of an integer number δ of view-switch interval Δ, $\Psi(\delta)$, by integrating $\psi(x)$ over the range $[(\delta-1)\Delta, \delta\Delta)$:

$$\Psi(\delta) = \int_{(\delta-1)\Delta}^{\delta\Delta} \psi(x), \qquad 1 \leq \delta \leq \delta_{\max} \tag{6.7}$$

Where $\delta_{\max} = \lfloor(\Delta + RTT_{\max})/\Delta\rfloor$. Then, given schedules $G(\delta)$'s for possible δ's, transmission cost $C(\mathcal{T}, G(), \mathbf{Q})$ of a frame structure $\mathcal{T}$ associated with QPs $\mathbf{Q}$ is defined as the expected transmission cost, *i.e.*,

$$C(\mathcal{T}, G(), \mathbf{Q}) = \sum_{\delta=1}^{\delta_{\max}} \Psi(\delta) \;\; t_\delta(\mathcal{T}, G(\delta), \mathbf{Q}) \tag{6.8}$$

where $G()$ denotes the set of schedules $G(\delta)$'s for all δ's.

For a given schedule $G(\delta)$, individual transmission cost $t_\delta(\mathcal{T}, G(\delta), \mathbf{Q})$ of structure $\mathcal{T}$ and QPs $\mathbf{Q}$ depends on view transition probability $\Omega_{k',k}(l)$, which can be calculated by adding up the sizes of all frames $F_{i,j}$'s in $\mathcal{T}$, scaled by the corresponding frame transmission probability (6.5):

$$t_\delta(\mathcal{T}, G(\delta), \mathbf{Q}) = \sum_{F_{i,j} \in \mathcal{T}} q(F_{i,j}, \delta) \; |F_{i,j}(\mathbf{Q})| \tag{6.9}$$

Distortion Cost

Since clients can request captured or synthesized views for observation, we define *distortion cost* as the average distortion of all captured and synthesized views available in the system. For the distortion of a picture in a captured view, we use the Mean-Squared-Error (MSE) between the original and coded versions of the texture maps of the picture. On the other hand, since no captured image is available for a virtual view, we synthesize an image using the uncompressed textures and the depth images of neighboring captured views as reference to calculate its MSE. We also denote Λ as the discrete set of available QPs for texture and depth coding.

Notice that the distortion of both captured views and virtual views are mainly influenced by the chosen QPs $\mathbf{Q}$, but independent of a particular frame structure $\mathcal{T}$. For example, in Fig. 6.6, captured view 4 at time 3 can be reconstructed with roughly the same distortion using either a P-frame $P_{3,4}^{(1)}$ or $P_{3,4}^{(2)}$, and virtual view 3.5 at time 3 can also be similarly synthesized using different coded frame pair, $P_{3,3}$ and $P_{3,4}^{(1)}$, $P_{3,3}$ and $P_{3,4}^{(2)}$. Let $D_j^c(Q_t)$ be the average distortion of frames at all view-switch instants of captured view j given the texture QP Q_t, and $D_k^s(\mathbf{Q})$ be the average distortion of synthesized frames at all view-switch instants of virtual view k given the texture/depth QPs $\mathbf{Q}$ used for both neighboring captured views. Distortion cost $D(\mathbf{Q})$ is then given by

$$D(\mathbf{Q}) = \frac{1}{(K-1)K' + K} \left(\sum_{j=1}^{K} D_j^c(Q_t) + \sum_{j=1}^{K-1} \sum_{k=1}^{K'} D^s_{j+\frac{k}{K'+1}}(\mathbf{Q}) \right) \tag{6.10}$$

From (6.6) and (6.10), it can be seen that coarse QPs $\mathbf{Q}$ result in smaller frame size of texture and depth coding, $|F_{i,j}^t(Q_t)|$ and $|F_{i,j}^d(Q_d)|$, and larger distortion $D(\mathbf{Q})$. This means that given a storage constraint, a frame structure can afford more redundant representations of one picture using redundant P-frames to lower transmission rate, at the expense of sacrificed visual quality. On the other hand, finer QPs $\mathbf{Q}$ can lower the distortion, but the increased frame size will lead to less redundancy (more M-frames) used in a frame structure, resulting in larger transmission rate.

6.3.3 Optimization Definition

We can now formally define our IMVS problem as a combinatorial optimization problem as follows.

Definition 6.3.1. *The IMVS optimization problem, denoted as* `IMVS`*, is to find a structure $\mathcal{T}$ using a combination of I-, P- and M-frames, and associated schedules $G(\delta)$'s for possible δ's, as well as texture/depth QPs $\mathbf{Q}$, that minimize the transmission cost $C(\mathcal{T}, G(), \mathbf{Q})$ while both a storage constraint $\bar{B}$ and a distortion constraint $\bar{D}$ are observed. Mathematically, this optimization problem is given by:*

$$\begin{aligned} \min_{\mathcal{T}\in\Theta, G(), \mathbf{Q}\in\Lambda} \quad & C(\mathcal{T}, G(), \mathbf{Q}) \\ s.t. \quad & B(\mathcal{T}, \mathbf{Q}) \leq \bar{B}, \quad D(\mathbf{Q}) \leq \bar{D} \end{aligned} \tag{6.11}$$

6.4 Algorithm Development

In this section, we develop algorithms to select a good frame structure, associated schedules, and texture/depth QPs for the optimization problem `IMVS` in (6.11). We first propose one iterative procedure by optimizing structure $\mathcal{T}$ and associated schedule set $G()$ only, then QPs $\mathbf{Q}$ only, while keeping the other set of variables fixed. We then present a greedy algorithm to optimize a frame structure $\mathcal{T}$ and schedule set $G()$ given QPs $\mathbf{Q}$. Finally, we present a low-complexity algorithm to update QPs $\mathbf{Q}$ for a given frame structure $\mathcal{T}$ and schedule set $G()$.

6.4.1 Two Sub-Problems

The optimization problem (6.11) is a joint optimization of frame structure $\mathcal{T}$ and texture/depth QPs $\mathbf{Q}$. To simplify the optimization, we divide `IMVS` into two simpler sub-problems, optimizing one set of variables while keeping the other set fixed. We formalize the definitions of the two sub-problems as follows.

Definition 6.4.1. *Given chosen texture/depth map QPs $\mathbf{Q}^{(k)}$ at iteration k that satisfy distortion constraint $\bar{D}$, `IMVS` degenerates to sub-problem one: find structure $\mathcal{T}$ and associated schedule set $G()$ to minimize transmission cost $C(\mathcal{T}, G(), \mathbf{Q}^{(k)})$, subject to storage constraint $\bar{B}$, i.e.,*

$$\begin{array}{ll} \min\limits_{\mathcal{T}\in\Theta, G()} & C(\mathcal{T}, G(), \mathbf{Q}^{(k)}) \\ s.t. & B(\mathcal{T}, \mathbf{Q}^{(k)}) \leq \bar{B} \end{array} \tag{6.12}$$

Notice that since the quality of view synthesis depends only on QPs $\mathbf{Q}^{(k)}$ and not on the particular chosen structure $\mathcal{T}$, the distortion constraint can be ignored in this sub-problem.

Definition 6.4.2. *Given a fixed frame structure $\mathcal{T}^{(k)}$ and associated schedule set $G^{(k)}()$ at iteration k, `IMVS` degenerates to sub-problem two: find QPs $\mathbf{Q}$ for texture and depth coding, such that the expected transmission cost $C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q})$ is minimized while observing both the storage constraint $\bar{B}$ and the distortion constraint $\bar{D}$:*

$$\begin{array}{ll} \min\limits_{\mathbf{Q}\in\Lambda} & C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}) \\ s.t. & B(\mathcal{T}^{(k)}, \mathbf{Q}) \leq \bar{B}, \quad D(\mathbf{Q}) \leq \bar{D} \end{array} \tag{6.13}$$

Based on the two sub-problems, we can summarize the iterative procedure for `IMVS` in Table 6.2. The crux is to solve the two sub-problems. In the following, we will propose a

Table 6.2: Summarized iterative optimization of frame structure, transmission schedule and quantization parameters

1) Initialize a pair of texture/depth QPs $\mathbf{Q}^{(0)}$ satisfying the distortion constraint $\bar{D}$. Set $k = 0$.
2) Fix $\mathbf{Q}^{(k)}$. Optimize structure $\mathcal{T}$ and associated schedule set $G()$, which minimize the expected transmission rate in sub-problem one (6.12). For $k > 0$, if the pre-defined convergence criterion is satisfied, stop.
3) Fix $\mathcal{T}^{(k)}$ and $G^{(k)}()$. Find $\mathbf{Q}$ that minimizes the expected transmission rate in sub-problem two (6.13).
4) Go to step 2. $k \leftarrow k+1$.

greedy frame structure and schedule optimization algorithm and a QP update algorithm, both with low complexity, to separately address the two sub-problems.

6.4.2 Frame Structure & Schedule Optimization

We first present a frame structure and schedule optimization algorithm given fixed QPs $\mathbf{Q}^{(k)}$. Though (6.12) differs in some respects from the frame structure optimization problem in [5], a similar proof can be easily constructed to show that sub-problem one is also NP-hard. Given the computational complexity of (6.12), we first convert the storage-constrained problem (6.12) into the following unconstrained problem:

$$\begin{aligned}\min_{\mathcal{T}\in\Theta, G()} \quad & J(\mathcal{T}, G(), \mathbf{Q}^{(k)}) = C(\mathcal{T}, G(), \mathbf{Q}^{(k)}) + \lambda B(\mathcal{T}, \mathbf{Q}^{(k)}) \\ & = \sum_{F_{i,j}\in\mathcal{T}} \left(\sum_{\delta} \Psi(\delta) q(F_{i,j}, \delta) + \lambda \right) |F_{i,j}(\mathbf{Q}^{(k)})|\end{aligned} \tag{6.14}$$

where the Lagrangian multiplier λ is a fixed parameter that represents the tradeoff between transmission rate and storage, and $J(\mathcal{T}, G(), \mathbf{Q}^{(k)})$ is the Lagrangian cost. To find the optimal λ that minimizes transmission cost while observing storage constraint in (6.12), a bisection-search method is used over a predefined range $[\lambda_{min}, \lambda_{max}]$.

We can interpret (6.14) as follows: a captured picture can be represented by multiple P-frames, each P-frame $P_{i,j}^{(h)}$ having a comparatively small transmission cost $\Psi(\delta)q(P_{i,j}^{(h)}, \delta)|P_{i,j}^{(h)}|$, but all together comprising a large storage cost $\lambda \sum_h |P_{i,j}^{(h)}|$. When λ is small, the penalty on large storage is negligible and redundant P-frames are attractive. However, when λ is large, the penalty on large storage becomes expensive and one single representation of the picture as M-frame with larger transmission cost but smaller storage is more preferable.

To solve (6.14) efficiently, we use a greedy approach to find near-optimal frame structure and associated schedules. In a nutshell, we iteratively build one "structure layer" t_i and "schedule layer" $g_i()$ one view-switch instant at a time from front to back. Structure layer t_i is comprised of frames $F_{i,j}$'s of all captured views j's at instant $i\Delta$, and schedule layer $g_i()$ consists of local schedules $g_i(\delta)$'s for all possible δ's, each mapping a structure slice $\Xi_{i-1}(\delta)$ in structure layer t_{i-1} to a structure slice $\Xi_i(\delta)$ in t_i, given client's view-switching feedback at instant $(i-\delta)\Delta$. At each view-switch instant $i\Delta$, the key question is: given structure $\mathcal{T}_{i-1}$ and schedule set $G_{i-1}()$ constructed up to instant $(i-1)\Delta$, how to optimally construct structure layer t_i and schedule layer $g_i()$ to minimize (6.14)?

To construct locally optimal structure layer t_i at view-switch instant $i\Delta$, we first initialize structure layer t_i^0 with K M-frames, one for each of K captured views. More precisely, for each M-frame $M_{i,j}$ of captured view j, we assign all frames $F_{i-1,k}$'s in structure layer t_{i-1} of instant $(i-1)\Delta$ that can switch to view j at instant $i\Delta$, as predictors of $M_{i,j}$. Since an M-frame is not a redundant representation (one frame per captured picture), the initial structure layer has minimum storage of all possible layers.

Corresponding to initial structure layer t_i^0, we construct initial schedule $g_i^0(\delta)$ given δ as follows. We first designate structure slices $\Xi_i(\delta)$'s using created M-frames, where each slice $\Xi_i(\delta)$ of center coordinate $c(\Xi_i(\delta)) = h$ has enough M-frames to enable view-switches to coordinates $\mathcal{V}_i = \{k \mid \max(1/d, h-\delta L) \leq k \leq \min(K/d, h+\delta L), k \in \mathcal{Z}^+\}$. Then, given client's view coordinate selection h at instant $(i-\delta)\Delta$, an initial schedule $g_i^0(\delta)$ will map any previously designated structure slice $\Xi_{i-1}(\delta)$ in t_{i-1} with center coordinate $c(\Xi_{i-1}(\delta)) = h'$, $\max(1/d, h-L) \leq h' \leq \min(K/d, h+L)$, to the same $\Xi_i(\delta)$, *i.e.*, $(\Xi_{i-1}(\delta), h) \overset{g_i^0(\delta)}{\Rightarrow} \Xi_i(\delta)$.

However, large M-frame sizes in initial structure layer t_i^0 lead to large transmission cost. To reduce transmission cost, we incrementally add the most "beneficial" redundant P-frames one at a time—beneficial meaning one that reduces the Lagrangian cost—thereby increasing storage. We terminate when no more beneficial redundant P-frames can be added.

In details, we describe the algorithm as follows. First, as initial structure layer t_i^0, we construct one M-frame for each captured view j at view-switch instant $i\Delta$. We then designate structure slices $\Xi_i(\delta)$'s and determine the corresponding schedules $g_i^0(\delta)$'s as described earlier, and compute the resulting local Lagrangian cost in (6.14). Given the initial structure and schedule layers, we improve t_i and $g_i()$ by iteratively making local structure augmentations: selecting one candidate from a set of structure augmentations at each iteration, which offers the largest decrease in local Lagrangian cost. The possible augmentations are:

- Add new P-frame $P_{i,j}$ to t_i, predicted from existing frame $F_{i,k}$ of neighboring view k of *same* instant $i\Delta$.
- Add a new P-frame $P_{i,j}$, predicted from an existing frame $F_{i-1,k}$ in t_{i-1} of the *previous* instant $(i-1)\Delta$.
- Select a different predictor $F_{i,k}$ of the *same* instant $i\Delta$ for an already constructed P-frame $P_{i,j}$ in t_i.

Notice that the last augmentation does not increase the number of representations of a given captured view, while each of the first two increases the number of frame representations by one P-frame.

Suppose that given constructed structure layer t_i^l in the l-th iteration, we build up the corresponding schedule $g_i^l(\delta)$ given δ by minimizing transmission bandwidth. More specifically, given a client's selected view coordinate h at view-switch instant $(i-\delta)\Delta$ and a structure slice $\Xi_{i-1}(\delta)$ in t_{i-1} with center coordinate $c(\Xi_{i-1}(\delta)) = h'$, $\max(1/d, h-L) \leq h' \leq \min(K/d, h+L)$, we designate structure slice $\Xi_i(\delta)$ by finding the set of frames $F_{i,j}$'s in t_i^l, which possesses the smallest size of transmitted frames while enabling all reachable coordinates $\mathcal{V}_i = \{k \mid \max(1/d, h-\delta L) \leq k \leq \min(K/d, h+\delta L), k \in \mathcal{Z}^+\}$ at view-switch instant $i\Delta$. This can be mathematically expressed as

$$g_i^l(\delta) : \min_{\Xi_i(\delta) \in \mathcal{G}'} \sum_{F_{i,j} \in \Xi_i(\delta)} |F_{i,j}| \tag{6.15}$$

where $\mathcal{G}' = \{\Xi_i(\delta) \mid (\Xi_{i-1}(\delta), h) \Rightarrow \Xi_i(\delta)\}$.

Note that different from the greedy frame optimization method in [67] where there is only one logical schedule for a given structure due to the assumption of zero network delay, in the proposed greedy algorithm, we need to optimize multiple schedules $g_i(\delta)$'s for different δ's in the schedule layer given a structure layer t_i at instant $i\Delta$, each corresponding to clients' view-switch feedbacks at different instants $((i-\delta)\Delta)$'s.

The above process repeats to find the most locally beneficial augmentation at each iteration, update the corresponding schedules by (6.15) and compute the local Lagrangian cost in (6.14), until no more Lagrangian cost reduction can be found. Note that after updating the local schedules at each iteration, it is possible that some frames in t_i are not used by any view-switch. In this case, those unused frames will be removed from the structure.

It is informative to compare our frame structure optimization problem with that in [63] and [66]. On one hand, both aims at optimizing frame structure with a given storage constraint. On the other hand, our optimization problem differs from that considered in [63] and [66] in three aspects. First, we consider frame structure optimization for variable RTT delays instead of structure optimization for idealized network with no delay as considered in [63] and [66]. Correspondingly, additional frames of more captured views than the requested one needs to be transmitted by server to enable clients play back video uninterrupted. Second, different from the view-switch model in [63] and [66] which limits the number of available requested views to clients by K camera-captured views, K' virtual views in-between each pair of captured views can also be requested by clients, so that a view-switch can take place between views more smoothly. In those senses, the frame structure optimization problem in [63] and [66] is a special case of the proposed one with $\delta = 0$ and $K' = 0$. Third, in our greedy algorithm to solve (6.14), we do not apply any time-consuming recursive optimization as described in [63] and [66], which sets up a layer t_i with considering future view-switches as well via recursion. Instead, starting with the initial structure, we incrementally made modifications to layer t_i only based on the structure $\mathcal{T}_{i-1}$ and the schedule G_{i-1} built up to previous view-switch instant $(i-1)\Delta$. It was shown in [67] that greedy algorithm can achieve similar optimal frame structure as that generated by recursive algorithm, but with much less computations.

6.4.3 Optimal Quantization Parameters Update

We next present a low-complexity algorithm to optimally update QPs $\mathbf{Q}$ for given structure $\mathcal{T}^{(k)}$ and schedule set $G^{(k)}()$, as defined by the second sub-problem (6.13). To find the optimal solution of the constrained optimization problem (6.13), the naive approach of exhaustively searching all candidates $\mathbf{Q}$'s that satisfy both storage constraint $\bar{B}$ and distortion constraint $\bar{D}$ is too expensive in practice. Instead, we develop a strategy to update QPs by first studying rate-quantization (RQ) and distortion-quantization (DQ) characteristics of multiview videos.

RQ Model Analysis

During the last decades, the relationship between rate and QP of video coding has been extensively studied for applications such as rate control. For example, in TM5 [97], a simple linear RQ model, *i.e.*, $R(QP) = X/QP$, is proposed, where X is a constant and R is size of the coded frame. The more accurate RQ models with high computational complexity, *i.e.*, $R_i = U(V\delta_i^2/QP_i^2 + W)$ in TMN8 [98] and $R_i = V_1 \times MAD_i/QP_i + V_2 \times MAD_i^2/QP_i^2$ in VM8 [99], are employed respectively. In TMN8, δ_i^2 is the variance of residues in a macroblock, U, V and W are constants. In VM8, MAD_i is the mean absolute value of a residue macroblock, V_1 and V_2 are two constants. In [100], an improved linear model with an offset indicating the overhead bits, *i.e.*, $R(QP) = X/QP + L$, is proposed for H.263. Based on the experiments on a large number of multiview video sequences, we adopt this modified linear RD model for H.263.

Fig. 6.7 shows the relationship between coded bits R and $1/QP$ of one I-, P- and DSC-frame, on the texture and depth coding respectively of the sequences `Dog` and `Pantomime` [76]. As shown in Fig. 6.7, R is linearly correlated with $1/QP$ no matter if the frame in question is coded using I-, P, or DSC-frame. As a consequence, the storage cost $B(\mathcal{T}, \mathbf{Q})$ in (6.6) of a given frame structure $\mathcal{T}$ can be written as a function of QPs $\mathbf{Q}$ as:

$$B(\mathcal{T}, \mathbf{Q}) = \sum_{F_{i,j} \in \mathcal{T}} \left(\frac{X_{i,j}^t}{Q_t} + L_{i,j}^t + \frac{X_{i,j}^d}{Q_d} + L_{i,j}^d \right) = \frac{X_1}{Q_t} + \frac{X_2}{Q_d} + L \tag{6.16}$$

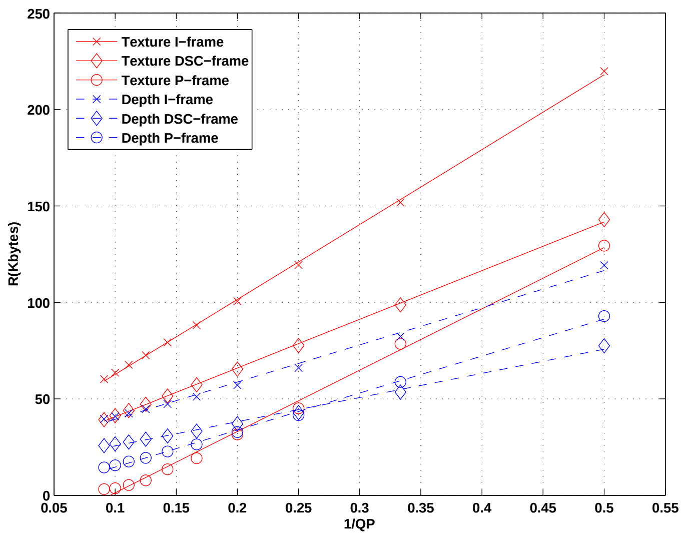
where $X_{i,j}^t$ and $L_{i,j}^t$, $X_{i,j}^d$ and $L_{i,j}^d$ are the individual parameters of texture and depth components for frame $F_{i,j}$, and $X_1 = \sum_{F_{i,j} \in \mathcal{T}} X_{i,j}^t$, $X_2 = \sum_{F_{i,j} \in \mathcal{T}} X_{i,j}^d$ and $L = \sum_{F_{i,j} \in \mathcal{T}} (L_{i,j}^t + L_{i,j}^d)$ are the corresponding parameters of the overall structure $\mathcal{T}$. In our experiments, instead of calculating the specific parameters for each frame $F_{i,j}$, X_1, X_2 and L of a given structure $\mathcal{T}$ can be directly estimated from a number of available RQ points (no fewer than 3) by the least-square solution of the following linear problem:

$$\mathbf{A}\,[X_1, X_2, L]^T = \mathbf{B} \tag{6.17}$$

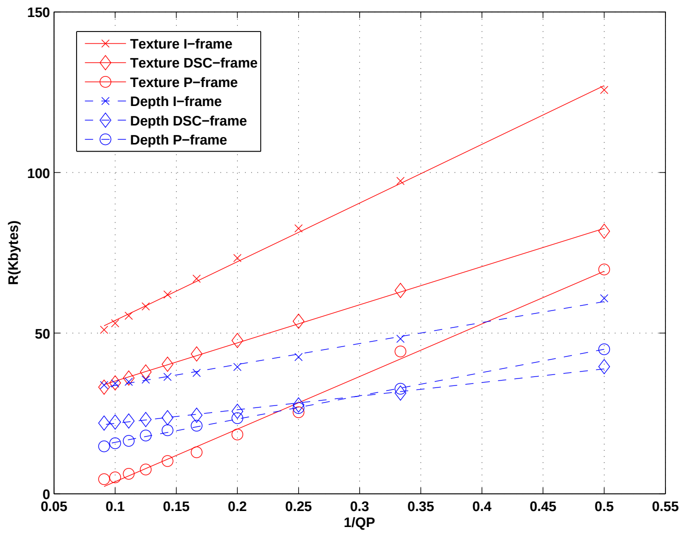
where matrix $\mathbf{A}$ and column vector $\mathbf{B}$ are composed of row vectors $[1/Q_t, 1/Q_d, 1]$'s and the storages of each available RQ point respectively.

DQ Model Analysis

To the best of our knowledge, the relationship between view synthesis distortion and texture/depth QPs has not been formally studied in the literature. However, in our experiments, we observed that the distortion cost defined in (6.10) is roughly correlated with



(a)



(b)

Figure 6.7: Relationship between R and $1/QP$ of I-, P- and DSC-frames, for the texture and depth coding of sequence (a) `Dog` and (b) `Pantomime`.

texture and depth QPs through a linear model, *i.e.*,

$$D(\mathbf{Q}) = Y_1 Q_t + Y_2 Q_d + Z \tag{6.18}$$

where Y_1, Y_2 and Z are constants. Fig. 6.8 shows the relationship between average distortion D and QPs $\mathbf{Q}$ of the sequences `Dog` and `Pantomime`, where $K = 4$ and $K' = 4$.

Quantization Parameters Update

Based on the RQ model (6.16) and DQ model (6.18), given the storage and distortion constraints $\bar{B}$ and $\bar{D}$ in (6.13), the set of valid QPs (Q_t, Q_d)'s can be shown to be the shaded region in Fig. 6.9. In Fig. 6.9, l_1 and l_2 are two boundaries of the valid region, which are determined by the corresponding constraints $\bar{B}$ and $\bar{D}$ respectively, and $\mathbf{Q}^1 = [Q_t^1, Q_d^1]^T$ and $\mathbf{Q}^2 = [Q_t^2, Q_d^2]^T$ are two intersection points between l_1 and l_2, with $Q_t^1 < Q_t^2$ and $Q_d^2 < Q_d^1$.

We introduce the following lemma, which can lead to a closed-form solution to optimally update QPs in (6.13).

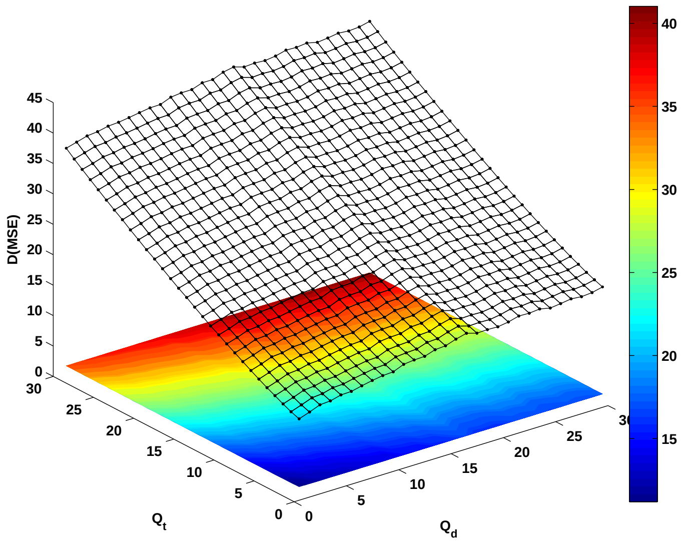
Lemma 6.4.1. *Given a fixed structure $\mathcal{T}^{(k)}$ and schedule set $G^{(k)}()$ at the k-th iteration of the algorithm, the optimal QPs $\mathbf{Q}$ of sub-problem (6.13) is located at the boundary line l_2, corresponding to the distortion constraint $\bar{D}$.*

Proof. The proof is given in Appendix 6.A. □

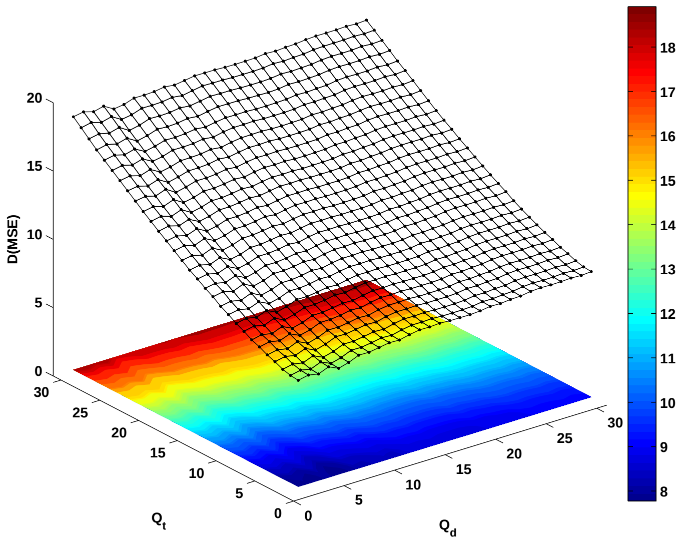
The conclusion of Lemma 6.4.1 suggests that we can now limit the search range of optimal QPs for given structure $\mathcal{T}^{(k)}$ and schedule set $G^{(k)}()$ to a line on which the distortions of all QPs are identically equal to the distortion constraint $\bar{D}$. Further, it turns out that with the help of Lemma 6.4.1, we can even derive a closed-form solution to update QPs of sub-problem two at each iteration, without any search process. More specifically, we first simplify sub-problem two (6.13) to a single-constraint problem, stated formally as a theorem below.

Theorem 6.4.1. *Given structure $\mathcal{T}^{(k)}$ and schedule set $G^{(k)}()$, the optimization of transmission $C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q})$ in terms of $\mathbf{Q}$, with storage constraint $\bar{B}$ and distortion constraint $\bar{D}$, is mathematically equivalent to the following univariate optimization problem:*

$$\begin{aligned} \min_{Q_t} \quad & A_1/Q_t + A_2/(\bar{D} - Z - Y_1 Q_t) + S \\ s.t. \quad & Q_t^1 \le Q_t \le Q_t^2 \end{aligned} \tag{6.19}$$



(a)



(b)

Figure 6.8: Relationship between distortion D and quantization parameters (Q_t, Q_d) of sequence (a) `Dog` and (b) `Pantomime`.

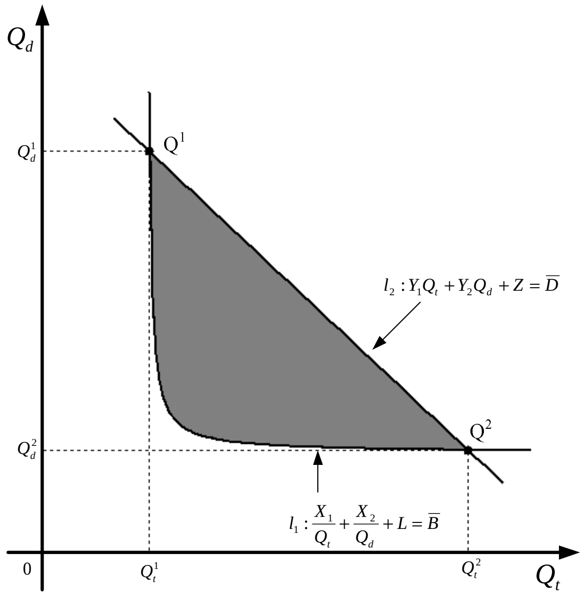

Q_d
Q^1
Q_d^1
$l_2 : Y_1 Q_t + Y_2 Q_d + Z = \bar{D}$
Q^2
Q_d^2
$l_1 : \frac{X_1}{Q_t} + \frac{X_2}{Q_d} + L = \bar{B}$
0
Q_t^1
Q_t^2
Q_t

Figure 6.9: Region of valid QP candidates for the second sub-problem.

where

$$
\begin{aligned}
A_1 = &\ \sum_{\delta} \Psi(\delta) \left(\sum_{F_{i,j} \in \mathcal{T}^{(k)}} q(F_{i,j}, \delta) X_{i,j}^t \right) \\
A_2 = &\ \left(\sum_{\delta} \Psi(\delta) \left(\sum_{F_{i,j} \in \mathcal{T}^{(k)}} q(F_{i,j}, \delta) X_{i,j}^d \right) \right) Y_2 \\
S = &\ \sum_{\delta} \Psi(\delta) \left(\sum_{F_{i,j} \in \mathcal{T}^{(k)}} q(F_{i,j}, \delta) (L_{i,j}^t + L_{i,j}^d) \right)
\end{aligned}
$$

Proof. By using Lemma 6.4.1 and taking RQ model (6.16) and DQ model (6.18) into (6.8), optimization sub-problem (6.13) can be found to become (6.19). □

Note that after applying Lemma 6.4.1, the denominator of the second component in (6.19) denotes that the corresponding view synthesis distortion of the optimized QPs just satisfies the distortion constraint $\bar{D}$, while the constraint $Q_t^1 \leq Q_t \leq Q_t^2$ guarantees the storage constraint $\bar{B}$ is satisfied.

Given the convexity of the target function in (6.19) over the range $[Q_t^1, Q_t^2]$, we next evaluate the minimization by taking derivative of (6.19) with respect to Q_t [6]. The optimal

[6]Though we focus on the optimization of (6.19) in terms of Q_t, the same process can be carried on Q_d as well, correspondingly, (6.19) is formulated as function of Q_d

(Q_t^*, Q_d^*) [7] is then found to be

$$
\begin{aligned}
Q_t^* = &\begin{cases} \frac{\sqrt{A_1}(\bar{D}-Z)}{\sqrt{A_1}Y_1+\sqrt{A_2}Y_1}, & Q_t^1 \leq \frac{\sqrt{A_1}(\bar{D}-Z)}{\sqrt{A_1}Y_1+\sqrt{A_2}Y_1} \leq Q_t^2 \\ Q_t^1, & \frac{\sqrt{A_1}(\bar{D}-Z)}{\sqrt{A_1}Y_1+\sqrt{A_2}Y_1} < Q_t^1 \\ Q_t^2, & \frac{\sqrt{A_1}(\bar{D}-Z)}{\sqrt{A_1}Y_1+\sqrt{A_2}Y_1} > Q_t^2 \end{cases} \\
Q_d^* = &\ (\bar{D} - Z - Y_1 Q_t^*)/Y_2
\end{aligned}
\tag{6.20}
$$

Note that similar to (6.16), to reduce the computational complexity, instead of using the specific expression in (6.19), all the necessary parameters A_1, A_2, Y_1, Y_2 and Z to calculate (Q_t^*, Q_d^*) in (6.20) can be directly derived by solving the linear problem (6.17) from multiple quantization parameters (Q_t, Q_d)'s for the fixed $\mathcal{T}^{(k)}$ and $G^{(k)}()$. The optimal QPs $\mathbf{Q}$ in Step 3 of the proposed iterative optimization algorithm can be then updated accordingly based on (6.20).

At the end of this section, for the completeness of description, we describe in Table 6.3 the complete procedures to iteratively optimize frame structure, associated schedule and quantization parameters, based on the proposed greedy frame structure and schedule generation algorithm in Sec. 6.4.2 and QP update algorithm in Sec. 6.4.3.

We claim that the proposed iterative joint optimization of frame structure, transmission schedule and quantization parameters in Table 6.3 is guaranteed to converge, which is stated formally as a theorem below.

Theorem 6.4.2. *Based on the greedy frame structure and schedule generation and QP update algorithms, the convergence of the proposed iterative optimization is guaranteed.*

Proof. The proof is given in Appendix 6.B. □

6.5 Simulation Results

6.5.1 Simulation Setup

To gather multiview video data for our experiments, we encoded the first 90 frames of sequences `Dog` and `Pantomime` [76] of 4 captured views ($K = 4$), at resolution 1280×960 and 30 frames per second. Each sequence has different characteristic and camera setup,

[7]The QPs of a practical video codec are chosen from a discrete set of values, while the RQ and DQ models we developed in this chapter are in continuous domain. Therefore, there is an inherent rounding error for the resulting optimal QPs, Q_t^* and Q_d^*.

Table 6.3: Procedures to iteratively find optimal frame structure, transmission schedule and quantization parameters

1) Initialize texture/depth quantization parameters $\mathbf{Q}^{(0)}$ satisfying the distortion constraint $\bar{D}$. Set $k = 0$, and specify values of λ_{min}, λ_{max}, and a tolerance ε as the convergence criterion.

2) Fix $\mathbf{Q}^{(k)}$ for any $k \geq 0$. Search the suitable trade-off parameter λ^* over $[\lambda_{min}, \lambda_{max}]$ for the given storage constraint $\bar{B}$. Generate optimal structure $\mathcal{T}^{(k)}$ and schedule set $G^{(k)}()$ based on the greedy structure and schedule generation algorithm, which achieves the following unconstrained minimum given λ^*:

$$\begin{aligned} \min_{\mathcal{T}^{(k)} \in \Theta, G^{(k)}()} \quad & J(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)}) \\ & = C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)}) + \lambda^* B(\mathcal{T}^{(k)}, \mathbf{Q}^{(k)}) \end{aligned}$$

For $k > 0$, if $C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)}) > C(\mathcal{T}^{(k-1)}, G^{(k-1)}(), \mathbf{Q}^{(k)})$, continue to use frame structure and schedule at the previous iteration, *i.e.*, set $\mathcal{T}^{(k)} = \mathcal{T}^{(k-1)}$, $G^{(k)}() = G^{(k-1)}()$. If $\left\|(C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)}) - C(\mathcal{T}^{(k-1)}, G^{(k-1)}(), \mathbf{Q}^{(k)}))/C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)})\right\| \leq \varepsilon$, stop the iteration and output $(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)})$ as optimal result. Otherwise, go to step 3 to continue the iteration.

3) Fix $\mathcal{T}^{(k)}$ and $G^{(k)}()$. Randomly select l ($l \geq 3$) quantization parameter pairs $\mathbf{Q}_i = [Q_{t,i}, Q_{d,i}]^T$, $i = 1, 2, \ldots, l$, calculate the transmission cost C_i and distortion cost D_i for the given $\mathcal{T}^{(k)}$ and $G^{(k)}()$. Then, estimate parameters A_1, A_2, Y_1, Y_2 and Z by solving two linear problems as (6.17). Update $\mathbf{Q}^{(k)}$ to $\mathbf{Q}^{(k+1)}$ based on (6.20) so that $\mathbf{Q}^{(k+1)}$ can achieve the following constrained minimum:

$$\begin{aligned} \min_{\mathbf{Q}^{(k+1)} \in \Lambda} \quad & C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k+1)}) \\ s.t. \quad & B(\mathcal{T}^{(k)}, \mathbf{Q}^{(k+1)}) \leq \bar{B}, \quad D(\mathbf{Q}^{(k+1)}) \leq \bar{D} \end{aligned}$$

4) Go to step 2, $k \leftarrow k + 1$.

e.g., different distance between neighboring capturing cameras, and capturing objects with various range of depth values. The depth estimation algorithm discussed in Chapter 5 is used to generate the depth maps. We set the number of synthesized views between neighboring captured views to be $K' = 4$; hence there are $(K-1)K' + K = 16$ views available for clients' selection. To synthesize the virtual views, the view synthesis reference software 2.0 [32] is used for DIBR. Note that different multiview video characteristics can affect the amount of geometric errors caused by depth map estimation, which is reflected by the resulting view synthesis distortion of DIBR. To generate data for DSC implementation of M-frames, we use the algorithm in [68], developed using H.263 tools (with half-pixel motion estimation). In addition, the random access period Δ' and view-switch period Δ are set to be 30 and 3, respectively. The Lagrangian multiplier λ is swept from 0.01 to 40.96 to induce different tradeoffs between storage and transmission rate.

For view-switching interactive model, we set view difference bound $L = K'+1 = 5$, which means the distance between two consecutive view-switches cannot exceed that between two neighboring captured views. In addition, we assume the view-switching probability function in the form $\Phi(n) = \phi_1 - \phi_2 \|n\|$, $-5 \leq n \leq 5$, where $\phi_2 = (11\phi_1 - 1)/30$ such that $\sum_n \Phi(n) = 1$. Note that ϕ_1 is the probability that client switches to the view coordinate where she remains in the same view-switch direction as previous view-switch, and ϕ_2 is the decreased probability when she transitions to neighboring view coordinates in other view-switch directions. By changing ϕ_1, we can model the behaviors of video streaming clients with different view-switching habits.

For the PDF of RTT delay, we assume an uniform distribution with upper bound $RTT_{max} = 5\Delta$ (500ms) for simplicity, *i.e.*, $\psi(x) = 1/(5\Delta)$, $x \in (0, 5\Delta)$. Correspondingly, the PMF $\Psi(\delta)$ of δ could be calculated from (6.7) as $\Psi(\delta) = 0.2$, $1 \leq \delta \leq 5$.

6.5.2 Simulation Results

Convergency Speed of Iterative Optimization

We first examine how fast the proposed iterative joint optimization algorithm of frame structure, transmission schedule and QPs could converge. In Fig. 6.10, supposing $\phi_1 = 0.2$, we plot the change of transmission rate at each step of the proposed iterative algorithm, where the storage constraint is 320 KBytes for all three sequences, the distortion constraint is set to be 23 and 11 for `Dog` and `Pantomime` respectively. In Fig. 6.10, the points with step

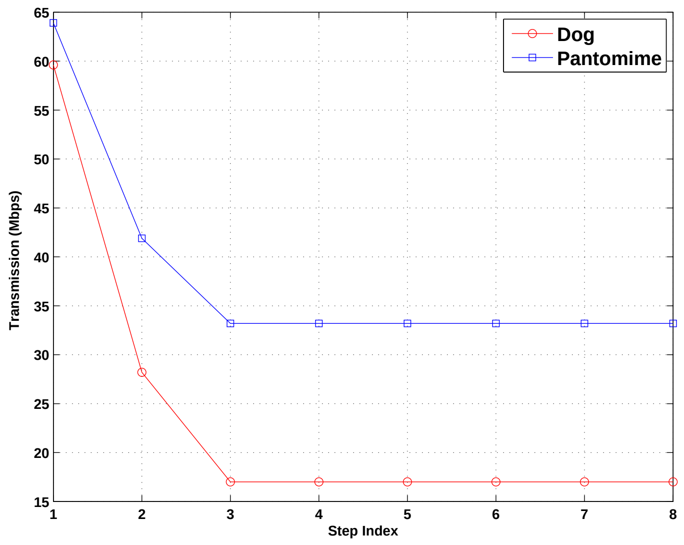
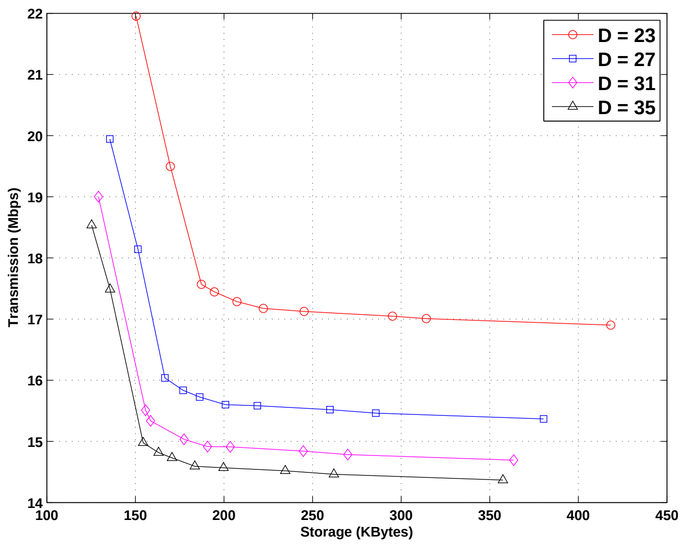


Figure 6.10: Convergency rate of the iterative procedure, where three sequences are encoded at 320KBytes with distortion $D = 23$ for `Dog`, $D = 11$ for `Pantomime`.

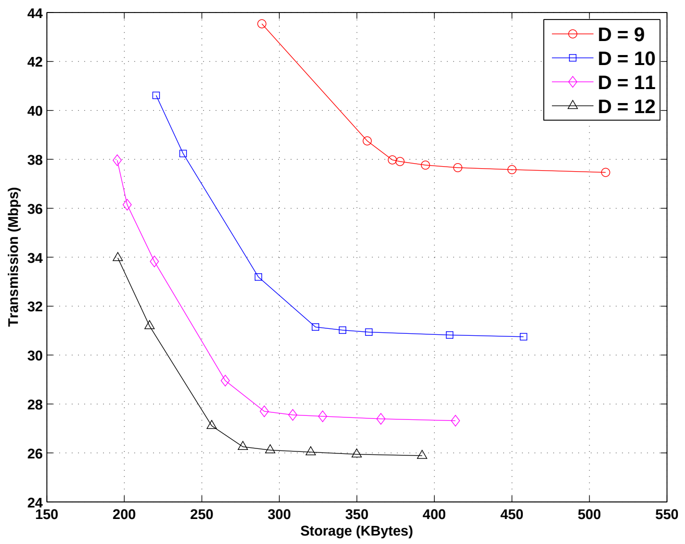
indices $2i - 1$ and $2i$ represent the transmission rates after Step 2 and Step 3 at the i-th loop of the iterative procedure, respectively. First, we can see that as shown in the proof of Theorem 6.4.2 in Appendix 6.B, the transmission rate is a non-increasing function at each step of the iterative algorithm. Second, we can observe that when applied to different sequences, the proposed iterative algorithm can coverage to the optimal solution very fast within 2 iterations, demonstrating the efficiency of the proposed algorithm to real multiview video data.

Algorithm Performance Comparison with Different Distortion Constraints

We next study the change of transmission rate resulting from our optimized structure, schedule and QPs when we vary storage and distortion constraints. When ϕ_1 is set to be 0.2, we generated tradeoff points between storage and transmission rate for different view synthesis distortion, shown in Fig. 6.11. First, for a given distortion, we see an inverse proportional relationship between transmission rate and storage, because larger storage budget means more frame structure redundancy, resulting in more bandwidth-efficient P-frames used in a frame structure. Second, we observe that in general larger distortion means a smaller transmission rate and storage. This is also expected, since better view synthesis quality means smaller quantization distortion, leading to comparatively large frame sizes of encoded texture and depth maps, which are expensive for both storage and transmission



(a)



(b)

Figure 6.11: Tradeoff between storage and transmission rate with different distortion constraints. (a) `Dog`; (b) `Pantomime`.

bandwidth. In addition, for the same storage, the transmission rate drops more slowly as the increase of distortion constraint.

Algorithm Performance Comparison Using Different Source Coding Models

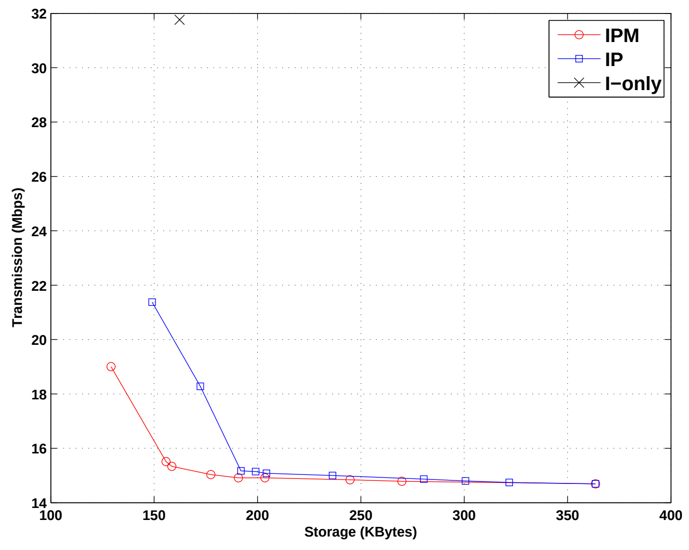
Then, we analyze the effects of different video source coding models on the performance of frame structures generated by our proposed algorithm. In Fig. 6.12, we plot the tradeoff points between storage and transmission rate for our algorithm using I-, P- and M-frames (`IPM`), using I- and P-frames (`IP`) and using only I-frames (`I-only`), when ϕ_1 is set to be 0.2 and distortion constraint is set to be 31 and 10 for `Dog` and `Pantomime` respectively. First, we observe that `I-only` has a single tradeoff point, because placing I-frames at all switching points results in no flexibility to trade off between storage and transmission rate, therefore could not take advantage of extra storage if available to lower transmission rate.

Second, for the same storage, `IPM` offers lower transmission rate by up to 51.7% for `Dog`, 22.8% for `Pantomime`, due to the optimal usage of redundant P- and DSC-frames. The performance improvement of `IPM` over `I-only` for `Pantomime` is much smaller than `Dog`. This is due to the relatively small size of I-frames in `Pantomime`, as a result of almost textureless background region in the sequence, which introduces adequate spatial redundancy for efficient intra prediction.

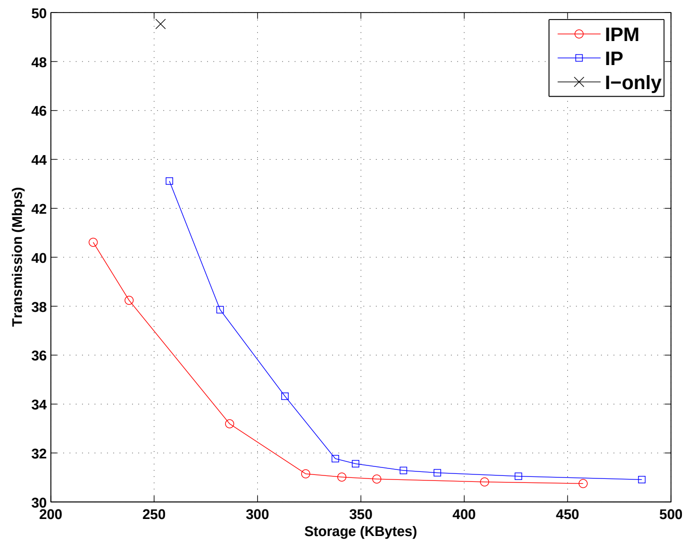
Third, we observe that structures using M-frames can offer a noticeable improvement over those using I-frames, with transmission rate saving up to 27.5% for `Dog` and 11.3% for `Pantomime`. The improvement is larger at stringent storage constraint, because DSC-frames are more often used by the optimized frame structure to lower overall storage.

Algorithm Performance Comparison with Different RTT Delays

We then evaluate the impact of RTT delays on the performance of frame structures optimized from the proposed algorithm. Given the corresponding storage and distortion constraints, Fig. 6.13 depicts the change in expected transmission rate with the increase of RTT delay when the same frame structure generated from the proposed algorithm is individually operated on server-client channels with different RTT delays. More specifically, we first optimize the frame structure, schedule and QPs to lower the expected transmission rate with respect to the PDF of RTT, $\psi(x)$, subject to given storage and view synthesis distortion

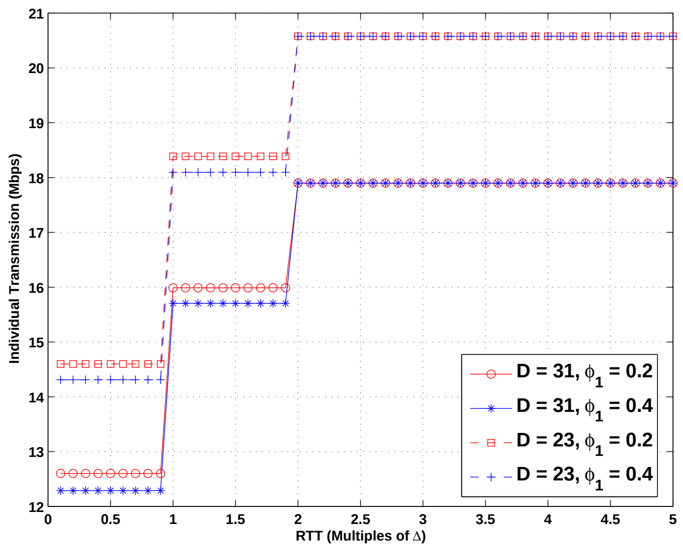


(a)

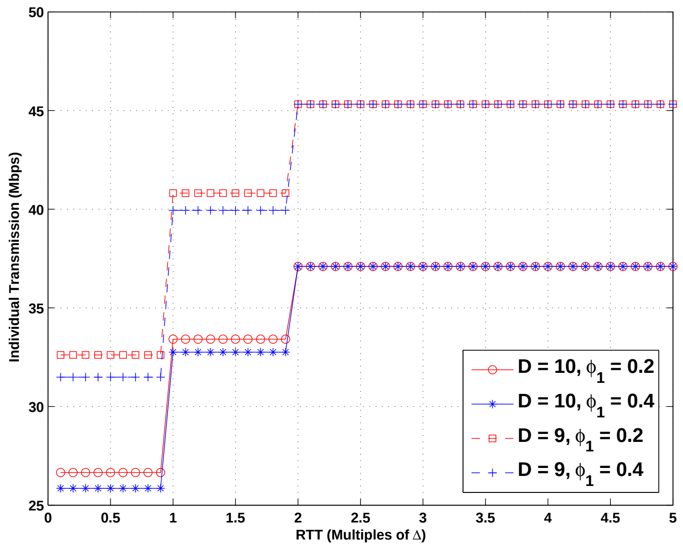


(b)

Figure 6.12: Tradeoff between storage and transmission rate using different coding configurations, for a given distortion constraint. (a) `Dog` with $D = 31$; (b) `Pantomime` with $D = 10$.



(a)



(b)

Figure 6.13: Transmission rate of a frame structure versus RTT delay. (a) `Dog` at 360KBytes; (b) `Pantomime` at 420KBytes.

constraints as discussed in Sec. 6.4. We then compare the corresponding individual transmission rate (6.9) when the resulting frame structure performs on different specific RTT delays. To induce different view-switching probability $\Phi(n)$, ϕ_1 is set to be 0.2 and 0.4 for two trials.

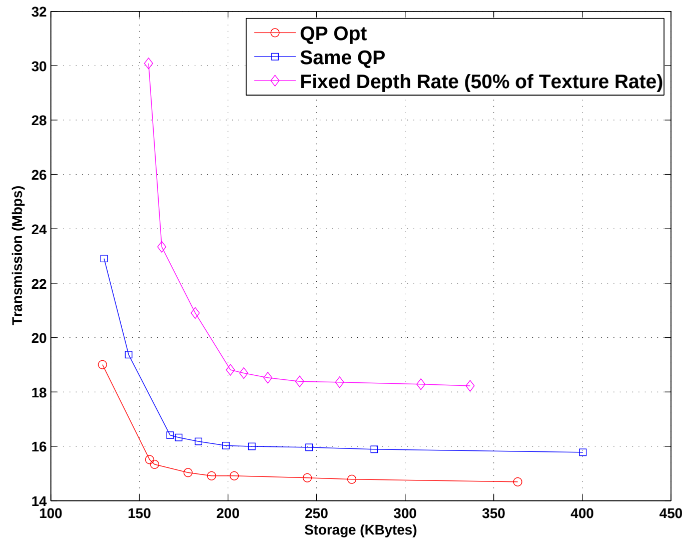
First, we see that transmission rate is a non-decreasing step function with the increase of RTT delay, and in general, larger RTT delay results in more transmission bandwidth consumption. This is intuitive: given a frame structure $\mathcal{T}$, all RTT delays RTT's, $(\delta-1)\Delta \leq RTT \leq \delta\Delta$ will use the same transmission schedule $G(\delta)$, leading to the same coded frames delivered from video server for each transmission, while in overall larger RTT delay means more view-switch positions to cover at each structure slice, resulting in larger transmission rate.

Second, we can observe that transmission rate cannot be further increased when $RTT \geq 2\Delta$. This can also be easily explained: when $RTT \geq 2\Delta$, one client is able to reach all $(K-1)K' + K = 16$ available view-switch positions within one RTT no matter which view-switch position she choose one RTT before. Correspondingly, each structure slice needs to cover all $K = 4$ captured views, resulting in a constant transmission rate.

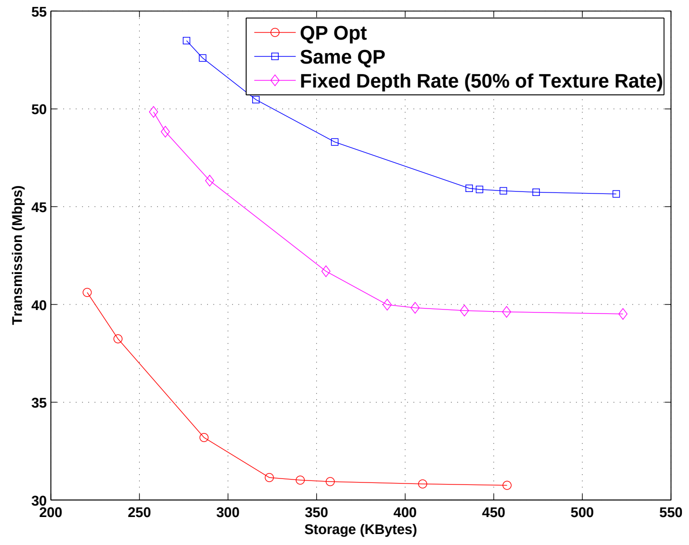
Third, as ϕ_1 is increased, the transmission rate of the optimized frame structure decrease. This is expected: larger ϕ_1 means higher probability that client remains at the same view-switch direction, which also increases the probability that client stays at the same view after one view-switch; therefore, more P-frames predicted from the previous frames of the same view are used in the structure, resulting in lower transmission rate. Moreover, the frame structure has the same transmission rate when $RTT \geq 2\Delta$, independent on ϕ_1, because of transmission of all captured views at each slice structure.

Improvement of Texture/Depth QP Optimization

Finally, we verify the effectiveness of the proposed quantization optimization algorithm for texture and depth map coding. Using the same distortion constraints for three sequences in Fig. 6.12, Fig. 6.14 compares the tradeoff points between storage and transmission rate generated by our quantization optimization algorithm (`QP Opt`) with two anchor results. The first (`Same QP`) uses the same QP to encode both texture and depth maps. The second (`Fixed Depth Rate`) is a constant rate allocation method with a pre-defined depth rate equal to 50% of texture rate. We observe that `QP Opt` consistently outperforms the other two methods for all test sequences, while `Fixed Depth Rate` is better than `Same QP` for



(a)



(b)

Figure 6.14: Tradeoff between storage and transmission rate using different selection methods for texture and depth QPs, for a given distortion constraint: (a) `Dog` with $D = 31$; (b) `Pantomime` with $D = 10$.

`Pantomime` but worse for `Dog`. For example, at a storage of 300 KBytes `QP Opt` yields a transmission rate reduction over `Same QP` about 8%, 38%, and over `Fixed Depth Rate` about 19%, 32%, for `Dog` and `Pantomime` respectively. It illustrates the importance of joint texture and depth quantization optimization.

6.6 Summary

In this chapter, we propose three major technological improvements to existing IMVS works. First, in addition to camera-captured views, we make available additional virtual views between each pair of captured views for clients' selection, by transmitting both texture and depth maps of neighboring captured views and synthesizing intermediate views at decoder using DIBR. Second, we construct a Markovian view-switching model that more accurately captures viewers' behaviors. Third, we optimize frame structures and schedule the transmission of frames in a network-delay-cognizant manner, so that clients can enjoy zero-delay view-switching even over transmission network with non-negligible RTT.

We formalize the joint optimization of the frame encoding structure, transmission schedule, and QPs of the texture and depth maps, and propose an iterative algorithm to achieve fast and near-optimal solutions. Experimental results show that our proposed rate allocation method can lower transmission rate by up to 38% over naive schemes. In addition, for the same storage, using our generated frame structures can lower transmission rate by up to 55% compared to I-frame-only structures, and up to 27% compared to structures without M-frames.

6.A Proof of Lemma 6.4.1

As shown in Fig. 6.9, given storage and distortion constraints $\bar{B}$ and $\bar{D}$ in (6.13), both two boundary lines l_1 and l_2 of the valid QP region are monotonically decreasing functions. Therefore, for any point $\mathbf{Q}^a = [Q_t^a, Q_d^a]^T$ in the valid region, we can always identify one unique point $\mathbf{Q}^b = [Q_t^b, Q_d^b]^T$ on l_2 such that $Q_t^b = Q_t^a$ and $Q_d^b \geq Q_d^a$. On the other hand, given a frame structure $\mathcal{T}^{(k)}$ associated with schedule set $G^{(k)}()$ and a texture quantization parameter Q_t, transmission cost function $C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q})$ is strictly decreasing function in terms of depth quantization parameter Q_d. So, we can have $C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^a) \geq C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^b)$. This proves that the optimal quantization solution $\mathbf{Q}$ in (6.13) is located on l_2.

6.B Proof of Theorem 6.4.2

In Table 6.3, given frame structure $\mathcal{T}^{(k-1)}$, schedule $G^{(k-1)}()$ and QP $\mathbf{Q}^{(k)}$ at step 2 of the k-th iteration, we compare the new result of $\mathcal{T}^{(k)}$ and $G^{(k)}()$ to that of $\mathcal{T}^{(k-1)}$ and $G^{(k-1)}()$ such that $C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)}) \leq C(\mathcal{T}^{(k-1)}, G^{(k-1)}(), \mathbf{Q}^{(k)})$. This means the transmission cost function is no-increasing at step 2. On the other hand, given frame structure $\mathcal{T}^{(k)}$, schedule set $G^{(k)}()$ and QP $\mathbf{Q}^{(k)}$ of the k-th iteration, we searches at step 3 the entire space of all possible QPs, Λ, for the best solution $\mathbf{Q}^{(k+1)}$ to lower transmission cost, *i.e.*, $\mathbf{Q}^{(k)} \in \Lambda$. This means $C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k+1)}) \leq C(\mathcal{T}^{(k)}, G^{(k)}(), \mathbf{Q}^{(k)})$. Hence, we prove that the transmission cost function is a no-increasing function at each step of the proposed iterative optimization algorithm.

Since both Θ and Λ are finite space, the no-increasing nature of the transmission cost function guarantees that the proposed iterative algorithm is surely to converge.

Chapter 7

Conclusions

7.1 Conclusions

In this thesis, we investigate the coding techniques in the practical designs of multiview video coding and interactive multiview video streaming. We first articulate that multiview video coding and interactive multiview video streaming are two fundamentally different applications of multiview video, and overview existing coding algorithms to support two applications respectively. After realizing the shortcomings of the previous works, various efficient coding techniques are presented for multiview video coding and interactive multiview video streaming, to achieve significant improvements over the existing methods.

We first develop projective rectification-based view interpolation and extrapolation methods and apply them to multiview video coding. Comparing to most of existing view synthesis methods without depth, which assume aligned cameras, our methods have little constraint on camera setup and require no camera parameters for view synthesis, due to the usage of view rectification. Experimental results show that the proposed view synthesis schemes achieve better performance than existing methods, and lead to improved RD performance than the current JMVC standard. Another important observation is that although the quality of the extrapolated views is generally lower than that of the interpolated views, the average RD performance in multiview video coding of view extrapolation-based method across all views can outperform that of the view interpolation-based method as the increase of the number of views, because view extrapolation can be applied to more views than view interpolation.

We next propose a geometrical model to analyze the performance of rectification-based view interpolation and extrapolation, which further allows more flexible camera setup in terms of positions and directions. Moreover, this model enables us to derive the theoretical gain of projective rectification on the quality of the interpolated or extrapolated view when unaligned cameras are used. On the other hand, we also develop an improved RD model to analyze the performances of different practical MVC schemes. Our model is a generalization of conventional motion compensation-based single view RD model to multiview video coding, with the consideration of modeling different view synthesis prediction methods. Simulation results of this model verify the experimental observations for various MVC schemes.

We also develop a depth estimation algorithm for depth-based multiview video representation. Compared with the conventional depth estimation algorithms, several improvements are proposed in our approach. First, multiview geometry is used to significantly reduce the complexity of the block-based matching as well as the probability of mismatch. Second, a structural similarity and maximum likelihood-based scheme is proposed to combine the depth information from multiple reference views. Third, a depth map smoothing algorithm is developed to improve the efficiency of depth coding. Experimental results show the superior performance of the proposed algorithm in view synthesis and depth coding. In addition, the depth maps generated from the proposed depth estimation method are used as the 3D representation method for our research on interactive multiview video streaming.

Finally, we propose three major technological improvements to existing works in interactive multiview video streaming. First, in addition to camera-captured views, we allow clients to select additional virtual views between each pair of captured views, by encoding both texture and depth maps of neighboring captured views and synthesizing intermediate views at decoder. Second, we construct a Markovian view-switching model that more accurately captures viewers' behaviors in view selections. Third, we consider the non-negligible transmission delay in the design of the frame structure and transmission schedule. As a consequence, frames in the structure corresponding to a possible delay can be additionally transmitted, so that a user can experience zero-delay view-switching. We formalize the joint optimization of the frame encoding structure, transmission schedule, and quantization parameters of the texture and depth maps, and propose an iterative algorithm to achieve fast and near-optimal solutions. Experimental results show that our proposed rate allocation method can provide significant transmission rate saving over naive schemes. In addition, for the same storage, using our generated frame structures can greatly lower transmission rate compared to I-frame-only structures and the structures without M-frames.

7.2 Future Work

The works in this thesis also reveal some interesting topics for future research, as stated as below.

7.2.1 Efficient Depth Coding

For monoscopic and multiview video content, highly optimized coding methods have been developed, including our view interpolation and extrapolation-based MVC schemes discussed in Chapter 3. However, for depth-based 3D video formats, specific coding methods for depth data that yield high compression efficiency are still in the early stage of investigation. Currently, depth data is treated as monochrome 2D video, which typically differs statistically from the colorful video content. Although color video contains detailed texture information, depth data is usually composed of large homogeneous regions of slow-changing values. In addition, object boundaries may cause abrupt changes in depth values. Therefore, depth data often contains very low and very high frequencies.

In our IMVS work of this thesis, we adopt the same conventional coding schemes to encode both texture and depth information. However, in conventional coding schemes, high frequencies are omitted at high compression rate. While this effect only leads to slightly degradation of texture information, the effects on depth coding are much severe. Therefore, besides the advanced coding techniques for texture information, efficient compression of depth data is also worth our future study.

7.2.2 Improved 3D Representation Format

Albeit depth map is widely used as a 3D scene representation format, the main drawback of 2D plus depth format is that it is only capable of rendering a limited depth range and is not specially designed to handle occlusions. Also, stereo signals are not easily accessible by this format, i.e., receivers needs to generate the second view to drive a stereo display.

To overcome the shortcomings of 2D plus depth format while maintaining some of its key merits, MPEG is now in the process of considering an improved representation format. The targets of this format include:

- Enable stereo devices to cope with varying display types and sizes. This includes the ability to vary the baseline distance for stereo video so that the depth perception experienced by the viewers is within a comfortable range.

- Facilitate support for high-quality auto-stereoscopic displays. Since directly providing all the necessary views for displays is not practical due to production and transmission, the new format aims to enable the generation of many high-quality views from a limited amount of input data.

7.2.3 Distributed Multiview Video Coding

The common idea of multiview video coding is to exploit the correlations between adjacent views in addition to temporal and spatial correlations within a single view. In other words, all the prediction processes are performed during encoding. Although prediction does improve the coding efficiency, it also incurs some problems in practical multiview video capturing and transmission system. Firstly, conducting inter-view prediction is based on the assumption that video data from different views can be freely exchanged for encoding. However, the communication between cameras with high data volume is difficult in practice. Second, all cameras are required to work simultaneously and multiview video sequences are required to be compressed with low latency. As we can see, all the above factors impose a big computational burden on encoder. Therefore, to lower encoding complexity, is there any way to separately encode each frame of multiview video while maintaining the coding performance as good as that of jointly encoding?

In theory, distributed source coding can provide one solution, which shows that correlated sources are encoded without using information from each others, coding performance can still be as good as that of dependent coding if the compressed signals can be jointly decoded [101, 102]. Up to nowadays, several practical Slepian-Wolf and Wyner-Ziv coding techniques have been proposed for conventional 2D video coding [103, 104, 105]. Since there are more redundancies in multiview video data than in single video data, the distributed multiview video coding can achieve better performance than single video.

7.2.4 Content-dependent View-switching Model

Many existing view-switching models including the one proposed in this thesis, are based on the assumption that viewers exhibit the same view-switching tendencies when watching different multiview video sequences. Such assumption is essentially content-independent since view-switching probability function does not vary according to the visual content of multiview video. However, it has been shown in [65] that viewers show strong content dependencies when switching views. For example, viewers intend to do more view-switches when there is an exciting action in a sport video, such as curvet. In addition, viewers' behaviors are quite different between martial arts and gymnastics, because gymnastics are often performed smoothly without sudden and fancy actions in martial arts. Therefore, a better view-switching model that uses content-dependent switching probability is also worth our future attention to more accurately capture viewers' behaviors in view selections.

Bibliography

[1] "The matrix," http://www.wikipedia.org/wiki/Bullet-time.

[2] "Eyevision," http://www.ri.cmu.edu/events/sb35/tksuperbowl.html.

[3] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1461–1473, Nov. 2007.

[4] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp, "Client-driven selective streaming of multiview video for interactive 3dtv," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1558–1565, Nov. 2007.

[5] G. Cheung, A. Ortega, and T. Sakamoto, "Coding structure optimization for interactive multiview streaming in virtual world observation," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Cairns, Queensland, Australia, Oct. 2008.

[6] A. Aaron, P. Ramanathan, and B. Girod, "Wyner-ziv coding of light fields for random access," in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Siena, Italy, Sept. 2004.

[7] D. Taubman and R. Rosenbaum, "Rate-distortion optimized interactive browsing of jpeg2000 images," in *Proc. IEEE Conf. Image Process.*, Barcelona, Spain, Sept. 2003.

[8] N.-M. Cheung, H. Wang, and A. Ortega, "Video compression with flexible playback order based on distributed source coding," in *Proc. SPIE Conf. Visual Commun. and Image Process.*, San Jose, CA, Jan. 2006.

[9] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. Journ. Comput. Vision*, vol. 47, no. 1, pp. 7–42, May. 2002.

[10] S. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor—system description, issues and solutions," in *Proc. Conf. Comput. Vision Pattern Recog. Workshop*, Washington, DC, Jun. 2004.

[11] X. Xiu, D. Pang, and J. Liang, “Projective rectification-based view interpolation and extrapolation for multiview video coding and free viewpoint video,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 6, pp. 693–707, Jun. 2011.

[12] D. Pang, X. Xiu, and J. Liang, “Multiview video coding using projective rectification-based view extrapolation and synthesis bias correction,” in *Proc. IEEE Int. Conf. Multimedia Expo*, New York, USA, Jun. 2009.

[13] X. Xiu and J. Liang, “Projective rectification-based view interpolation for multiview video coding and free viewpoint generation,” in *Proc. Pict. Coding Symp.*, Chicago, IL, USA, May 2009.

[14] ——, “An improved rate-distortion model for multiview video coding,” in *Proc. IEEE Conf. Image Process.*, Hong Kong, China, Sept. 2010.

[15] ——, “Rate-distortion analysis of rectification-based view interpolation for multiview video coding,” in *Proc. IEEE Int. Conf. Multimedia Expo*, New York, USA, Jun. 2009.

[16] ——, “An improved depth map estimation algorithm for view synthesis and multiview video coding,” in *Proc. SPIE Conf. Visual Commun. and Image Process.*, Huangshan, China, Jul. 2010.

[17] X. Xiu, G. Cheung, and J. Liang, “Delay-cognizant interactive streaming of multiview videos with free viewpoints synthesis,” *IEEE Trans. Multimedia*, submitted, Jun. 2011.

[18] ——, “Frame structure optimization for interactive multiview video streaming with bounded network delay,” in *Proc. IEEE Conf. Image Process.*, accepted, Brusseles, Belgium, Sept. 2011.

[19] X. Xiu, G. Cheung, A. Ortega, and J. Liang, “Optimal frame structure for interactive multiview video streaming with view synthesis capability,” in *Proc. IEEE Int. Conf. on Multimedia Expo Workshop on Hot Topics in 3D*, Bacelona, Spain, Jul. 2011.

[20] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge Univ. Press, 2003.

[21] R. Hartley, “Theory and practice of projective rectification,” *Int. Journ. Comput. Vision*, vol. 35, no. 2, pp. 115–127, 1999.

[22] S. Birchfield and C. Tomasi, “Depth discontinuities by pixel-to-pixel stereo,” *Int. Journ. Comput. Vision*, vol. 35, no. 3, pp. 269–293, 1999.

[23] M. Droese, T. Fujii, and M. Tanimoto, “Ray-space interpolation based on filtering in disparity domain,” in *Proc. 3D Image Conf.*, Jun. 2004.

[24] ——, “Ray-space interpolation constraining smooth disparities based on loopy belief propagation,” in *Proc. Int. Conf. Syst. Signal. and Image Process.*, Sept. 2004.

[25] S. Roy, “Stereo without epipolar lines: A maximum flow formulation,” *Int. Journ. Comput. Vision*, vol. 1, no. 2, pp. 1–15, 1999.

[26] V. Kolmogorov and R. Zabih, “Computing visual correspondence with occlusions using graph cuts,” *Int. Journ. Comput. Vision*, vol. 2, pp. 508–515, 2001.

[27] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, “Multi-view imaging and 3dtv,” *IEEE Signal Process. Mag.*, vol. 24, no. 6, pp. 10–21, 2007.

[28] C. Fehn, E. Cooke, O. Schreer, and P. Kauff, “3d analysis and image-based rendering for immersive tv application,” *Signal Process. Image Commun.*, vol. 17, no. 9, pp. 705–715, 2002.

[29] H.-Y. Shum, S. B. Kang, and S.-C. Chan, “Survey of image-based representations and compression techniques,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 11, pp. 1020–1037, Nov. 2003.

[30] H.-Y. Shum, S.-C. Chan, and S. B. Kang, *Image-based rendering.* Springer, 2007.

[31] Y. Liu, Q. Huang, S. Ma, D. Zhao, and W. Gao, “Joint video/depth rate allocation for 3d video coding based on view synthesis distortion model,” *Signal Process. Image Commun.*, vol. 24, no. 8, pp. 666–681, 2009.

[32] M. Tanimoto, T. Fuji, K. Suzuki, N. Fukushima, and Y. Mori, “Reference softwares for depth estimation and view synthesis,” in *ISO/IECJTC1/SC29/WG11*, Archamps, France, Apr. 2008.

[33] T. Berger, *Rate distortion theory: a mathematical basis for data compression.* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[34] B. Girod, “Efficiency analysis of multihypothesis motion-compensated prediction for video coding,” *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 173–183, Feb. 2000.

[35] M. Flierl, A. Mavlankar, and B. Girod, “Motion and disparity compensated coding for multi-view video,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1474–1484, Nov. 2007.

[36] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the h.264/avc video coding standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[37] “Requirements on multiview video coding v7,” ISO/IECJTC1/SC29/WG11, N8218, Jul. 2006.

[38] “Joint draft 1.0 on multiview video coding,” ISO/IEC MPEG ITU-T VCEG, JVT-U209, Nov. 2006.

[39] A. Smolic, K. Muller, T. Rein, and T. Wiegand, “Free viewpoint video extraction, representation, coding and rendering,” in *Proc. IEEE Conf. Image Process.*, Singapore, Oct. 2004.

[40] K. Hideaki, K. Masaki, K. Kazuto, and Y. Yoshiyuki, “Multiview video coding using reference picture selection for free-viewpoint video communication,” in *Proc. Pict. Coding Symp.*, San Francisco, CA, Dec. 2004.

[41] K. Muller, P. Merkle, H. Schwarz, T. Hinz, A. Smolic, T. Oelbaum, and T. Wiegand, “Multi-view video coding based on h.264/avc using hierarchical b-frames,” in *Proc. Pict. Coding Symp.*, Beijing, China, Apr. 2006.

[42] X. San, H. Cai, J. Lou, and J. Li, “Multiview image coding based on geometric prediction,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1536–1548, Nov. 2007.

[43] D. Tian, P. Pandit, P. Yin, and C. Gomila, “Study of mvc coding tools,” Joint Video Team, JVT-Y044, Shenzhen, China, Oct. 2007.

[44] W. S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, “Depth map coding with distortion estimation of rendered view,” *SPIE Visual Inf. Process. Commun.*, vol. 17, no. 11, pp. 1536–1548, Nov. 2007.

[45] K. J. Oh, S. Yea, A. Vetro, and Y. S. Ho, “Depth reconstruction filter and down/up sampling for depth coding in 3d video,” *IEEE Signal Process. Lett.*, vol. 16, no. 9, pp. 747–750, Sept. 2009.

[46] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P. H. N. de With, and T. Wiegand, “The effects of multiview depth video compression on multiview rendering,” *Signal Process. Image Commun.*, vol. 24, no. 12, pp. 73–88, Jan. 2009.

[47] Y. Morvan, A. Smolic, D. Farin, K. Muller, and P. H. N. de With, “Multiview depth-image compression using an extended h.264 encoder,” in *Proc. Advanced Concept Intell. Vision Syst.*, Delft, Netherlands, Aug. 2007.

[48] K. Yamamoto, M. Kitahara, H. Kimata, T. Yendo, T. Fujii, M. Tanimoto, S. Shimizu, K. Kamikura, and Y. Yashima, “Multiview video coding using view interpolation and color correction,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1436–1449, Nov. 2007.

[49] D. Farin, Y. Morvan, and P. H. N. de With, “View interpolation along a chain of weakly calibrated cameras,” in *Proc. IEEE Workshop Content Generation Coding 3D Telev.*, Eindhoven, Netherlands, Jun. 2006.

[50] B. Girod, “The efficiency of motion-compensating prediction for hybrid coding of video sequences,” *IEEE J. Sel. Areas Commun.*, vol. 5, no. 7, pp. 1140–1154, Aug. 1987.

[51] ——, “Motion-compensating prediction with fractional-pel accuracy,” *IEEE Trans. Commun.*, vol. 41, no. 4, pp. 604–612, Apr. 1993.

[52] M. Flierl and B. Girod, “Video coding with motion-compensated lifted wavelet transforms,” *Signal Process. Image Commun.*, vol. 19, no. 7, pp. 561–575, Aug. 2004.

[53] P. Ramanathan and B. Girod, “Rate-distortion analysis for lightfield coding and streaming,” *Signal Process. Image Commun.*, vol. 21, no. 6, pp. 462–475, Jul. 2006.

[54] M. Magnor, P. Ramanathan, and B. Girod, “Multi-view coding for image-based rendering using 3d scene geometry,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 11, pp. 1092–1105, Nov. 2003.

[55] K. Takahashi and T. Naemura, “Theoretical model and optimal prefilter for view interpolation,” in *Proc. IEEE Conf. Image Process.*, San Diego, CA, Oct. 2008.

[56] M. Ishii, K. Takahashi, and T. Naemura, “Rate-distortion performance of multiview image coding,” in *Proc. IEEE Conf. Image Process.*, San Diego, CA, Oct. 2008.

[57] M. Levoy and P. Hanrahan, “Light field rendering,” in *Proc. SIGGRAPH*, New Orleans, LA, Aug. 1996.

[58] P. Ramanathan and B. Girod, “Random access for compressed light fields using multiple representations,” in *Proc. IEEE Int. Workshop Multimedia Signal Process.*, Siena, Italy, Sept. 2004.

[59] C.-L. Chang and B. Girod, “Receiver-based rate-distortion optimized interactive streaming for scalable bitstreams of light fields,” in *Proc. IEEE Int. Conf. Multimedia Expo*, Taiwan, Jun. 2004.

[60] I. Bauermann and E. Steinbach, “Rdtc optimized compression of image-based scene representation (part i): modeling and theoretical analysis,” *IEEE Trans. Image Process.*, vol. 17, no. 5, pp. 709–723, May 2008.

[61] ——, “Rdtc optimized compression of image-based scene representation (part ii): practical coding,” *IEEE Trans. Image Process.*, vol. 17, no. 5, pp. 724–736, May 2008.

[62] “Video coding for low bitrate communication,” ITU-T Recommendation H.263, Febr. 1998.

[63] G. Cheung, A. Ortega, and N.-M. Cheung, “Generation of redundant coding structure for interactive multiview streaming,” in *Proc. Int. Packet Video Workshop*, Seattle, WA, May 2009.

[64] A. M. Tekalp, E. Kurutepe, and M. R. Civanlar, “3dtv over ip: End-to-end streaming of multiview video,” *IEEE Signal Process. Mag.*, pp. 77–87, 2007.

[65] J.-G. Lou, H. Cai, and J. Li, “A real-time interactive multi-view video system,” in *Proc. ACM Multimedia*, Singapore, Nov. 2005.

[66] G. Cheung, N.-M. Cheung, and A. Ortega, “Optimized frame structure using distributed source coding for interactive multiview streaming,” in *Proc. IEEE Conf. Image Process.*, Cairo, Egypt, Nov. 2009.

[67] G. Cheung, A. Ortega, and N.-M. Cheung, “Interactive streaming of stored multiview video using redundant frame structures,” *IEEE Trans. Image Process.*, vol. 20, no. 3, pp. 744–761, March 2011.

[68] N.-M. Cheung, A. Ortega, and G. Cheung, “Distributed source coding techniques for interactive multiview video streaming,” in *Proc. Pict. Coding Symp.*, Chicago, IL, May 2009.

[69] M. Karczewicz and R. Kurceren, “The sp- and si-frames design for h.264/avc,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 637–644, Jul. 2003.

[70] N.-M. Cheung and A. Ortega, “Distributed source coding application to low-delay free viewpoint switching in multiview video compression,” in *Proc. Pict. Coding Symp.*, Lisbon, Portugal, Nov. 2007.

[71] “Computation of the fundamental matrix,” http://www.cs.unc.edu/simblloyd/comp290-089/fmatrix.

[72] D. G. Lowe, “Object recogninition from local scal-invariant features,” in *Proc. IEEE Int. Conf. Comput. Vision*, Kerkyra, Greece, Sept. 1999.

[73] “Joint multiview coding (JMVC) 3.0,” Nov. 2008, garcon.ient.rwth-aachen.de.

[74] “Test sequences,” https://research.microsoft.com/en-us/groups/vision/ImageBasedRealities/3DVideoDownload/.

[75] “Test sequences,” https://www.3dtv-research.org/3dav-MVC-Sequences-YUVs/.

[76] “Test sequences,” http://www.tanimoto.nuee.nagoya-u.ac.jp/.

[77] E. Martinian, A. Behrens, J. Xin, and A. Vetro, “View synthesis for multiview video compression,” in *Proc. Pict. Coding Symp.*, Beijing, China, Apr. 2006.

[78] S. Yea and A. Vetro, “View synthesis prediction for multiview video coding,” *Signal Process. Image Commun.*, vol. 24, no. 1, pp. 89–100, Jan. 2009.

[79] H. Kimata, S. Shimizu, Y. Kunita, M. Isogai, K. Kamikura, and Y. Yashima, “Real-time mvc viewer for free viewpoint navigation,” in *Proc. IEEE Int. Conf. Multimedia Expo*, Hannover, Germany, Jun. 2008.

[80] J. Starck, A. Maki, S. Nobuhara, A. Hilton, and T. Matsuyama, "The multiple-camera 3d production studio," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 6, pp. 856–869, Jun. 2009.

[81] D. Geiger, B. Ladendorf, and A. Yuille, "Occlusions and binocular stereo," *Int. Journ. Comput. Vision*, vol. 14, pp. 211–226, 1995.

[82] P. N. Belhumeur, "A bayesian-approach to binocular stereopsis," *Int. Journ. Comput. Vision*, vol. 19, pp. 237–260, 1996.

[83] J. Xin, A. Vetro, E. Martinian, and A. Behrens, "View synthesis for multiview video compression," in *Proc. Pict. Coding Symp.*, Beijing, China, Apr. 2006.

[84] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[85] M. J. Black and A. Jepson, "Estimating optical flow in segmented images using variable-order parametric models with local deformations," *IEEE Trans. Pattern Analy. Mach. Intel.*, vol. 18, no. 10, pp. 972–986, 1996.

[86] R. Kumar, P. Anandan, and K. Hanna, "Direct recovery of shape from multiple views: a parallax based approach," in *Proc. Int. Conf. Pattern Recog.*, Jerusalem, Israel, 1994.

[87] H. Tao, H. S. Sawhney, and R. Kumur, "A global matching framework for stereo computation," in *Proc. IEEE Int. Conf. Comput. Vision*, Vancouver, Canada, Jul. 2001.

[88] M. Magnora, P. Eisert, and B. Girod, "Multi-view image coding with depth maps and 3d geometry for prediction," in *Proc. SPIE Conf. Visual Commun. and Image Process.*, San Jose, CA, Jan. 2001.

[89] K. Karantzalos and N. Paragios, "Higher order polynomials, free form deformations and optical flow estimation," in *Proc. IEEE Conf. Image Process.*, Genoa, Italy, Sept. 2005.

[90] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. Journ. Comput. Vision*, vol. 59, no. 2, pp. 167–181, Sept. 2004.

[91] S. Ince, E. Martinian, S. Yea, and A. Vetro, "Depth estimation for view synthesis in multiview video coding," in *3DTV Conf.*, Kos Island, Greece, Dec. 2006.

[92] "Test sequences," ftp://ftp.merl.com/pub/avetro/mvc-testseq/orig-yuv/.

[93] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *Proc. IEEE Conf. Image Process.*, San Antonio, TX, Oct. 2007.

[94] T. Fujii and M. Tanimoto, "Free viewpoint tv system based on ray-space representation," in *Proc. of SPIE*, vol. 4864, 2002.

[95] S. Wee, W. Tan, J. Apostolopoulos, and M. Etoh, "Optimized video streaming for networks with varying delay," in *Proc. IEEE Int. Conf. Multimedia Expo*, Lausanne, Switzerland, Aug. 2002.

[96] D. Bertsekas and R. Gallager, Eds., *Data Networks.* Prentice Hall, 1992.

[97] J. Katto and M. Ohta, "Mathematical analysis of mpeg compression capability and its application to rate control," in *Proc. IEEE Conf. Image Process.*, Washington, DC, Oct. 1995.

[98] J. Ribas-Corbera and S. Lei, "Rate control in dct video coding for low-delay communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 172–185, Febr. 1999.

[99] "Mpeg4 video verification model version 8.0," ISO/IECJTC1/SC29/WG11, Stockholm, Sweden, Jul. 1997.

[100] C. Wong, O. C. Au, B. Meng, and H. Lam, "Novel h.26x optimal rate control for low-delay communications," in *Proc. Int. Conf. Inf. Commun. Signal Process.*, Singapore, Dec. 2003.

[101] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. 19, pp. 471–480, Jul. 1973.

[102] A. D. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at decoder," *IEEE Trans. Inf. Theory*, vol. 22, pp. 1–10, Jan. 1976.

[103] A. Aaron, S. Rane, and B. Girod, "Transform-domain wyner-ziv codec for video," in *Proc. SPIE Conf. Visual Commun. and Image Process.*, San Jose, CA, Jan. 2004.

[104] D. Rowitch and L. Milstein, "On the performance of hybrid fec/arq system using rate compatible punctured turbo codes," *IEEE Trans. Commun.*, vol. 48, no. 6, pp. 948–959, Jun. 2000.

[105] A. Aaron, R. Zhang, and B. Girod, "Wyner-ziv coding of motion video," in *Proc. Asilomar Conf. Signals Syst. and Comput.*, Pacific Grove, CA, Nov. 2002.