DESIGNS OF APPLICATION-SPECIFIC MULTIVIEW / 3D VIDEO CODING

by

Yu Gao

M.Sc., Beijing University of Technology, 2008

B.Sc., Beijing University of Technology, 2005

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Doctor of Philosophy

in the
School of Engineering Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2014

APPROVAL

Name: Yu Gao

Degree: Doctor of Philosophy

Title: DESIGNS OF APPLICATION-SPECIFIC MULTIVIEW / 3D VIDEO CODING

Examining Committee: **Chair**: Dr. Ivan V. Bajić

Dr. Jie Liang, Associate Professor, Senior Supervisor

Dr. Mirza Faisal Beg, Professor, Supervisor

Dr. Gene Cheung, Associate Professor,
National Institute of Informatics, Japan, Supervisor

Dr. Sami Hakam Muhaidat, Assistant Professor,
Khalifa University, UAE, Supervisor

Dr. Jiangchuan Liu, Professor, SFU Examiner

Dr. Ming-Ting Sun, Professor,
University of Washington, External Examiner

Date Approved: September 9th, 2013

Partial Copyright Licence

Abstract

Many applications of multiview or three dimensional (3D) videos have been developed. This poses great challenges to video coding. We study multiview video coding (MVC), perceptual multiview video coding, 3D geometry compression, interactive multiview video streaming (IMVS), and free viewpoint video (FVV). The applications studied in this thesis can be classified into two categories.

In the first category we focus on rate-distortion (RD) performance, where the distortion can be measured by mean squared errors (MSE), human visual system based MSE, or metro distance. First, we consider the application of FVV and propose a novel inpainting assisted approach to efficiently compress multiview videos. The decoder can independently recover missing data via inpainting, resulting in lower rate. Second, we study the application of just noticeable distortion (JND)-based MVC and propose to exploit inter-view or temporal redundancy of JND maps to synthesize or predict target JND maps, which are then used to adjust prediction residuals. Third, we study 3D geometry compression and propose a new 3D geometry representation. We project 3D geometry to a collection of surrounding tiles, and subsequently encode these tile images using a modified MVC. The crux of the scheme is the optimal placement of image tiles.

In the second category, we study applications where real-time computation and the associated complexity also need to be considered, in addition to the RD performance. These applications include IMVS and FVV. We first consider view switching in IMVS, an application where a network client requests from server a single view at a time but can periodically switch to other views as the video is played back uninterrupted. We propose the optimal frame structure such that frequently chosen view switches are pre-computed while infrequent ones are computed in real-time upon request. On the other hand, we examine the decoder side computational complexity of view synthesis in FVV. We propose to optimally tradeoff transmission rate for decoder-side complexity. For regular view synthesis, we find the optimal subset of intermediate views to code. For a novel inpainting assisted paradigm, we argue that some expensive operations can be avoided by directly sending intra-coded blocks.

Acknowledgments

During my PhD program, 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 has influenced me most and guided me during the study towards my doctoral degree at Simon Fraser University. He is a great professor and great person. I am very grateful for the offer he gave me to do research under his guidance, grateful for the comfortable environment he provided to let me free to explore the unknown, grateful for the chances he offered me to present my work and communicate with people in academic conferences, and grateful for the opportunities he created to work together with researchers around the world. 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 (NII), Japan, who not only supervised me when I was an intern at NII but also gave fundamental ideas and pivotal suggestions to me in several papers of mine, and thus made a great impact to this thesis.

I would like to thank Professor Pascal Frossard, Dr. Thomas Maugey, and Professor Weisi Lin for extensively exchanging ideas when we were working on several papers. Without their brilliant thoughts, I could not complete those works that constitute the main body of this thesis.

I am very grateful to my supervisors, Professor Sami Hakam Muhaidat and Professor Mirza Faisal Beg, for their constructive advices and comments to this thesis. In addition, I have benefited enormously from taking Professor Sami Hakam Muhaidat's course: Stochastic Systems.

I also gratefully thank the internal and external examiners, Professor Jiangchuan Liu and Professor Ming-Ting Sun at University of Washington, for their suggestions to my thesis. I also thank Professor Ivan V. Bajić for being the Chair of my defense.

I would thank all the amazing members at the Multimedia Communication Laboratory, for making my life during the last five years a fun, challenging and memorable one. Especially, I would like to thank Xiaoyu Xiu (who recommended me to Dr. Jie Liang's group and contributed a lot in my papers), Dong Zhang, Chongyuan Bi, Xiao Luo, Xing Wang, Siyu Wu, Golara Javadi, Omar Alhussein, Xiaozheng Huang, Jing Wang, Mehdi Seyfi, Calvin Che, Duncan Chan, Andrew Au, Mohammad

Jafar Taghiyar Renani, and Parmida Beigi to name a few, for their insightful comments and helpful discussions.

Finally, I would like to dedicate this thesis to my parents and my girlfriend Lanting Wang, 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 Motivation

1.1.1 Multiview Image and Video

A digital image is a two-dimensional (2D) array of digital values (called pixels). A video is a set of consecutively moving visual images. People used to watch video programs on television via cable, tape, or optical disc. In the past decade, the consumption of Internet streaming video has been explosively growing. As reported by Sandvine [1] in 2013, Netflix accounted for nearly one-third of US Internet traffic at home. The efficient storage and streaming of such huge amount of data poses great challenges to the industry and research community.

Recent years, the three-dimensional (3D) visual technology has been drastically changing the video entertainment industry. 3D films that greatly enhance depth perception have been widespread since the huge success of the film “Avatar” in 2009, though the history of 3D films can be traced back to the beginning of the 20th century. Further, 3D visual technology has stepped out theaters and occupied living rooms with the advent of 3D television (3DTV), which provides either stereoscopic display or autostereoscopic experience without the need of glasses. Besides the commercial success of 3D films and 3DTVs, there have been a great deal of efforts on offering novel visual experience beyond the conventional 2D display. Panoramas extend the capability of regular capturing devices, and present much wider fields of view. Panoramic photos can be captured by special cameras like fisheye, or be synthesized using multiview techniques. Another novel way to display 3D content is the free viewpoint TV (FTV) proposed by Nagoya University [32], which enables viewers to interactively switch the viewpoints of a dynamic scene on a conventional 2D monitor, offering the 3D photographical experience instead of that created by 3D graphics. Similar to FTV, Carnegie

[1] http://www.tubefilter.com/2013/05/14/netflix-us-internet-traffic-percentage/

Figure 1.1: An example of multiview texture-plus-depth images from the video sequence `Ballet`.

Mellon University introduced to Super Bowl XXXV an "Eye Vision" system [2], where 30 cameras were installed around the stadium that gave TV viewers a flying-through feeling. Since these 3D visual techniques employ multiview images / videos instead of 3D geometrical models, the enormous efforts to create the models are avoided, and the visual experience is more realistic.

The multiview images / videos to support those 3D visual presentations are captured by a number of cameras. In this thesis, we call the image / video from a fixed camera as a view. To enable depth perception or render new views, depth maps should be measured or estimated. The Depth value refers to the distance between the image plane of camera and the corresponding point in the scene. The viewpoints that are not captured can possibly be synthesized using the color images (also called texture images in the thesis) from neighboring cameras using the geometrical information provided by depth maps. This view synthesis procedure is called depth image based rendering (DIBR) [29]. To enable the above applications, we need to store and transmit the multiview texture-plus-depth images [69], as shown in Fig. 1.1 [3]. Compared to single view image / video, the amount of raw data is increased by several times. In this thesis, facing the aggravating challenge, we propose several solutions of multiview image / video compression in order to satisfy the diverse requirements of different application scenarios.

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

[3] The video sequence can be downloaded from http://research.microsoft.com/en-us/downloads/5e4675af-03f4-4b16-b3bc-a85c5bafb21d/

1.1.2 Multiview Video Coding

Digital images and videos are huge in size but rich in repetitive patterns (redundancy). One second of 24-bit 1080p video (resolution: 1920×1080) at 30 frames per second (fps) needs approximately 1.5 billion bits (186.6 megabytes) to represent the uncompressed raw data. However, pixels in a small spatial region tend to be similar, a pixel patch may recur in the same image, and the changes between consecutive frames in a video are relatively small. Image and video compression methods try to remove the redundancies and represent the target image / video in a compact bitstream.

The design of most video coding methods is primarily aimed at having the highest compression efficiency, which is the ability to encode video at the lowest possible bit rate while maintaining a certain level of video quality. The ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations have developed a series of video coding standards, such as H.261, H.262/MPEG-2, H.263, MPEG-4, H.264/AVC, and the most recent H.265/HEVC (high efficiency video coding). Although the first version of HEVC has just been finalized, Netflix has started using HEVC for its famous TV series "House of Cards".

The video coding standards from H.261 to HEVC are all block-based. In H.264 [101] and HEVC [89], the size of block is adapted to local characteristics of image. Small blocks are used to code complex regions while large blocks are used to code smooth regions. In a block, the inter-pixel correlation is decorrelated by coding tools like intra prediction and transform, and the temporal redundancy of neighboring frames is exploited by motion-compensated prediction. Quantization is a way to trade off image quality for compression ratio. The quantized transform coefficients are converted to bitstream by entropy coding, where remaining statistical redundancies are removed.

As mentioned, we require multiview video coding (MVC) [30] to enable 3D visual experiences. The amount of raw data is proportionally increased with the increased number of views. However, if the cameras are closely spaced as applicable to most applications, the correlation among neighboring views is strong. The main challenge for MVC is thus how to efficiently reduce the inter-view redundancy. Similar to the removal of temporal redundancy, the motion-compensated prediction (usually called disparity-compensated prediction in this case) has reasonable performance, but this time the images from neighboring views are used as references. Fig. 1.2 is an example of MVC predictive frame structure, where view 0 is compressed by a regular single view coding scheme while the frames of view 1 are predicted by the frames of view 0 at the same time instant via disparity estimation, in addition to the traditional motion estimation. Taking inter-view correlation into consideration, MVC outperforms the simulcast scheme that independently codes each view.

1.1.3 Related Applications of MVC

In this thesis, we consider several applications of MVC. Free viewpoint video (FVV) allows users to smoothly change the viewpoint. FVV can be enabled by stored multiview texture-plus-depth video,

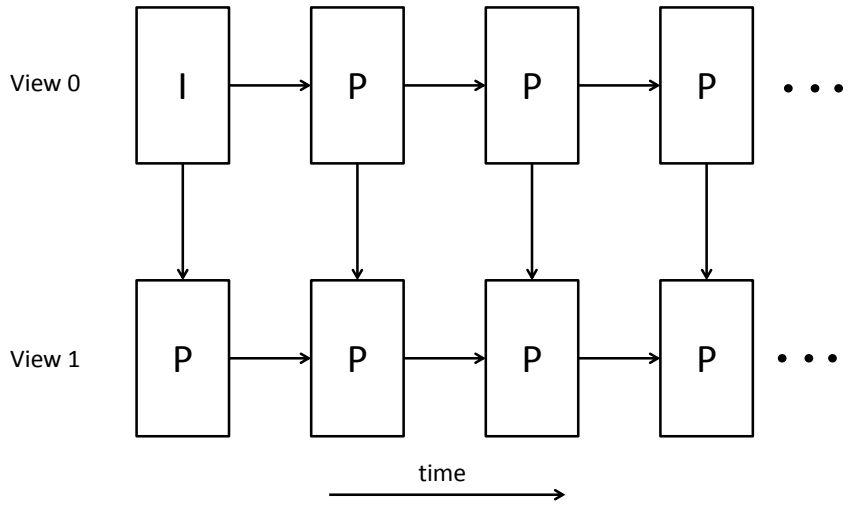


Figure 1.2: Example of MVC inter-frame and inter-view prediction structure, where I denotes intra-coded frame and P denotes inter-predicted frame.

and it is critical to compress the data efficiently. In interactive multiview video streaming (IMVS) [15], a network client requests from server a single view at a time but can periodically switch to other views as the video is played back uninterrupted, where the MVC strategy should be changed adapting to user behaviors. 3D geometry compression [25] plays an essential role in computer graphics, and it could be converted to an MVC problem and represented in a more compact format. The just noticeable distortion (JND)-based [53] perceptual video coding can be extended to multiview scenario, and it is possible to borrow some tools in MVC to the generation of multiview JND. To sum up, it is necessary to design novel MVC algorithms based on the characteristics of each application.

1.2 Background Knowledge

To facilitate the understanding of our contributions, we review some important background knowledge that is closely related to the work in this thesis, including view synthesis, image inpainting and JND.

1.2.1 DIBR-Based View Synthesis

Image-based rendering (IBR) enables 3D scene to be represented by a number of sample images without using explicit 3D geometry. IBR-based view synthesis is used to create novel views of a scene from images taken from different viewpoints. There are many image-based representations for a scene such as light field [58], lumigraph [45], concentric mosaics [85], view morphing [80], but the works in this thesis focus on texture-plus-depth format in which a view is not only described by

a color image but also a depth map where the per-pixel depth value corresponds to the distance between camera and the 3D scene. With a scene represented by multiple closely-spaced views, depth image-based rendering (DIBR) [29] can be used for view synthesis.

One of DIBR techniques, forward warping, project the pixels in the reference view to 3D space using the camera calibration parameters and depth map of the view, and then the 3D points are back-projected to the target view. With Lambertian assumption [4], the pixel values in the reference view are not changed during DIBR. Mathematically, given pixel (x_r, y_r) in reference view r and its depth value $d(x_r, y_r)$, the corresponding 3D world coordinate of (x_r, y_r) is

$$(x_w, y_w, z_w)^{\mathrm{T}} = \mathbf{R}_r^{-1} \cdot (d(x_r, y_r) \cdot \mathbf{A}_r^{-1} \cdot (x_r, y_r, 1)^{\mathrm{T}} - \mathbf{t}_r), \tag{1.1}$$

where $\mathbf{R}_r$, $\mathbf{A}_r$, and $\mathbf{t}_r$ are rotation matrix, intrinsic matrix, and translate vector of view r, respectively. The coordinate of the rendered pixel (x_v, y_v) in target view v can be obtained by

$$(x_v, y_v, 1)^{\mathrm{T}} = d(x_v, y_v)^{-1} \mathbf{A}_v \cdot (\mathbf{R}_v \cdot (x_w, y_w, z_w)^{\mathrm{T}} + \mathbf{t}_v), \tag{1.2}$$

where $\mathbf{R}_v$, $\mathbf{A}_v$, and $\mathbf{t}_v$ are rotation matrix, intrinsic matrix, and translate vector of view v, respectively. $d(x_v, y_v)$ is the depth value of pixel (x_v, y_v) in target view v, and is equal to $d(x_r, y_r)$ assuming 1D parallel camera arrangement.

After DIBR, there are some holes left due to disocclusion, a fact that the background objects blocked by foreground objects in the reference view appear in the target view. We can use image inpainting techniques (discussed in the following) to fill the holes, or merge several rendered images together if there are more than one reference views.

1.2.2 Image Inpainting

Image inpainting modifies the pixel values in selected image regions such that the regions do not stand out with respect to its surroundings. Image inpainting can be used to remove unwanted objects in an image, to restore damaged pictures, or to conceal errors due to unreliable transmission.

Generally speaking, there are two ways of image inpainting. The first way uses either a variational principle or a partial differential equation (PDE). In this category, Ogden *et al* [76] proposed to use a Gaussian pyramid and a series of interpolation, upsampling, and downsampling to inpaint. [66] proposed to interpolate missing pixels by extending the isophotes of image to mimic human visual system that tries to complete edges in an occlusion context. The term “image inpainting” was coined by Bertalmío *et al* in [5], where PDE is used to solve the inpainting problem. The pixel values along the edges are propagated from the boundary of missing regions towards the interior. Bertalmío *et al* [4] further proposed to use the Navier-Stokes equation of fluid dynamics to inpaint image.

[4] In Lambertian reflectance, the apparent brightness of a surface to an observer is the same regardless of the observer's angle of view.

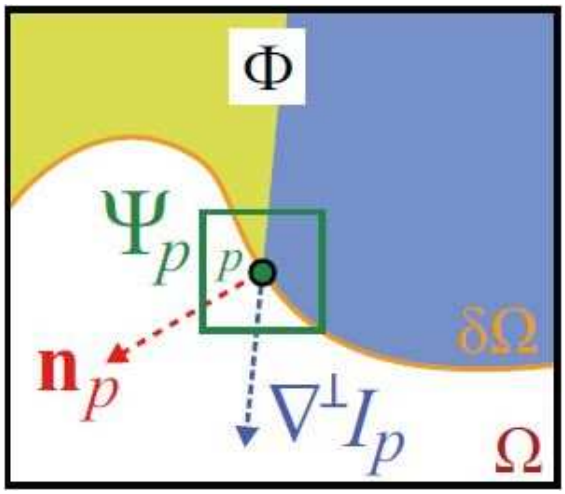


Figure 1.3: Notation diagram of Criminisi's inpainting algorithm [21].

In this thesis, however, we focus on the second way, the patch-based inpainting. Criminisi's inpainting algorithm [21] is a popular patch-based inpainting method, which is a significant extension of [27]. Criminisi's inpainting algorithm propagates texture patterns from known pixel regions to spatial regions with missing pixels, assuming that self-similarity typically exists in natural images. For convenience and consistency, we reuse some notations from Criminisi's algorithm. We denote the image by I, and Φ and $\Omega = I \setminus \Phi$ respectively denote the source and target regions in the inpainting process. As illustrated in Fig. 1.3, the pixels in Ω that are neighbors of Φ form the boundary, denoted by $\delta\Omega$. The region Φ includes the known pixels, while Ω represents the holes with unknown pixels. A square-shaped target patch Ψ_p centered at pixel p on the boundary $\delta\Omega$ is chosen by a priority-based patch selection procedure. The patch Ψ_p has two non-overlapping parts: the known region $\Psi_p \cap \Phi$ (also called *template* in the sequel) and the target unknown region $\Psi_p \cap \Omega$.

Mathematically, the priority value $P(p)$ for a patch centered at p takes surrounding pixels and linear structures into consideration:

$$P(p) = C(p)\, D(p), \tag{1.3}$$

where $C(p)$ and $D(p)$ are the confidence and data terms respectively. The confidence term $C(p)$ evaluates the amount of reliable information in known region $\Psi_p \cap \Phi$ by summing up the confidence values in the known region (normalized by $|\Psi_p|$):

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \Phi} C(q)}{|\Psi_p|}, \tag{1.4}$$

The more known and filled pixels in $\Psi_p \cap \Phi$, the higher value $C(p)$ is. Hence the confidence term $C(p)$ promotes the selection of patches with more known pixels—ones that are easier to inpaint.

The data term $D(p)$ encourages detected linear structure to be extrapolated as early as possible. The reason is that pixel neighborhood with prominent linear structures (*e.g.* strong edges) are more likely to continue propagating into the unknown region. If the normal n_p to the boundary $\delta\,\Omega$ is close to the isophote $\nabla I_p^{\perp}$ (gradient) of known region at pixel p, their inner product will be a large value; we compute $D(p)$ as:

$$D(p) = \frac{\nabla I_p^{\perp} \cdot n_p}{\alpha}, \tag{1.5}$$

where α is used for normalization and $\alpha = 255$ for typical 8-bit image. Similar to $C(p)$, the data term can be interpreted as another measure of "ease-to-inpaint": higher $D(p)$ implies more reliable propagation of linear structure. Combining the two into priority term $P(p) = C(p)\,D(p)$, the priority essentially leads to an "easy-to-hard" order.

For a given target patch Ψ_p with highest priority, template matching is performed in [21] to find the patch Ψ_q in Φ that is the most similar to known pixels in $\Psi_p \cap \Phi$:

$$\Psi_q^* = \arg\min_{\Psi_q \in \Phi} d(\Psi_p, \Psi_q) \tag{1.6}$$

where the distortion term $d(\,)$ is computed using only known pixels in Ψ_p and their corresponding pixels in Ψ_q. After the optimal Ψ_q^* is found, the pixels in Ψ_q^* that correspond to missing pixel locations in $\Psi_p \cap \Omega$ will be copied over for completion of Ψ_p.

1.2.3 Just Noticeable Distortion

Just noticeable distortion (JND) model [53] for image means that any error around a pixel below its JND threshold cannot be perceived by human being. In this thesis, the pixel-domain JND is used to eliminate the perceptual redundancy in MVC. A JND map consists of JND thresholds, one for each image pixel. The JND threshold is a value corresponding to the compound effect of background luminance adaptation and texture masking. The spatial JND threshold around pixel (x, y) can be computed using the nonlinear model proposed in [110]:

$$\mathrm{JND}_s(x, y) = T_l(x, y) + T_t(x, y) - C_{l,t} \min\{T_l(x, y), T_t(x, y)\},$$

where $T_l(x, y)$ and $T_t(x, y)$ are the visibility thresholds for background luminance adaptation and texture masking, respectively; $C_{l,t}$ refers to the overlapping effect in masking.

Psychovisual experiments have showed that human eyes are sensitive to luminance contrast rather than absolute luminance intensity, *i.e.*, very bright or very dark regions can hide more noise than middle gray level regions. This phenomenon is called background luminance adaptation. Therefore $T_l(x, y)$ is modeled as the function of the average background luminance $\overline{I(x, y)}$, and

its parameters are fitted from the data collected in psychovisual experiments [19]. It can be written as

$$T_l(x,y) = \begin{cases} 17(1-\sqrt{\frac{\overline{I(x,y)}}{127}})+3, & \text{if } \overline{I(x,y)} \leq 127 \\ \frac{3}{128}(\overline{I(x,y)}-127)+3, & \text{otherwise} \end{cases}$$

When $\overline{I(x,y)} \leq 127$, $T_l(x,y)$ is a monotonously decreasing function. Therefore pixels around darker background luminance areas can hide more errors. When $\overline{I(x,y)} > 127$, $T_l(x,y)$ is a linearly increasing function; hence regions with brighter background luminance have more perceptual redundancies.

The texture masking accounts for the reduction in the visibility of the pixel around which spatial non-uniformity happens. Intuitively, textured regions can hide larger errors than smooth ones. However, the areas with object edges are perceptually vulnerable to noise. To avoid over-estimation, the edge information is taken into consideration in [110] when calculating the visibility threshold of texture masking:

$$T_t(x,y) = \eta \cdot G(x,y) \cdot W_e(x,y), \tag{1.7}$$

where η is a control parameter, $G(x,y)$ is the maximal weighted average of gradients around the pixel at (x,y), and $W_e(x,y)$ is an edge-related weight, which can be obtained by edge detection and Gaussian filter.

In contrast to JND, minimally noticeable distortion (MND) [53] may be slightly visible. The MND can be employed to image and video compression. The fast JND generation methods proposed in Chapter 3 could be applicable to the pixel-domain MND as well.

1.3 Related Works

We next review the related works in the literature, including MVC, inpainting assisted coding, geometry compression, and IMVS. We show how they motivate our works and the differences from the thesis.

1.3.1 Related Works on MVC

Multiview image and video coding refers to the compression of multiple color images of the same 3D scene captured using an array of closely spaced cameras. Many papers on this topic focus on the efficient coding of the entire set of images by exploiting the inter-view data redundancy inherent in the camera setup. A straightforward way to achieve it is to use disparity-compensated prediction. Similar to motion-compensated prediction in single-view video coding, for each block in the target view, disparity compensation finds the best matching block in a reference view, then encodes and transmits the disparity vector (DV) and the prediction residual for reconstruction at

the decoder. In [64], motion compensation and disparity compensation are combined to encode stereo sequences. The concept of Group of Group of Pictures (GoGOP) for inter-view prediction is introduced in [49], where a picture can refer to decoded pictures in other views even at different time instants. In [70, 73], various modified hierarchical bidirectionally predicted structures are developed for inter-view prediction. In [52], the problem of optimal GoGOP prediction structure for multiview video coding is studied. However, simple 2D translational inter-view motion assumed by disparity compensation cannot accurately represent the geometry transformation in a view-switch from one camera viewpoint image to another; hence, disparity compensation is not always efficient.

Texture-plus-depth format is an alternative data representation that is particularly useful for free viewpoint video [69]. Texture and depth images from multiple camera viewpoints are encoded together, and one can synthesize novel intermediate viewpoints via DIBR [97]. In this kind of representation, an alternative to translational disparity prediction is view synthesis prediction (VSP) [65, 72, 111], where a synthesized version of a target view is used for predictive coding. Though high in computation complexity, using only texture images it is also possible to first estimate disparity (depth) using stereo-matching methods [26,57] and then partially synthesize a target viewpoint image [105, 107].

Most view synthesis methods assume rectified cameras, *i.e.*, the two cameras are parallel and only differ from each other by a small horizontal shift. To deal with more general camera setups, a rectification-based view interpolation method is proposed in [28]. In [105], an improved rectification-based view interpolation and extrapolation method is developed for MVC. In Chapter 2, we also try to deviate from the assumption of translational inter-view motion by using DIBR to render the target picture, but the warped images are used for further inpainting-based completion instead of additional reference.

As depth images are used only for view synthesis and are not themselves directly viewed, different rate-distortion (RD) optimization procedures have been designed for depth map coding to optimize the synthesized view quality [12, 13, 17, 56]. In particular, since depth images possess unique signal characteristics such as piecewise smoothness (PWS), new coding tools such as graph Fourier transform (GFT) designed specifically for depth signals have been proposed [50, 82].

Recently, HEVC has been extended to support encoding of 3D video, namely multiview video and associated depth data [74, 91], similar to the MVC extension of H.264/AVC [10]. There are mainly two types of tools employed: disparity compensation and view synthesis prediction. As discussed earlier, this is the hybrid signal prediction / residual coding paradigm used in conventional video coding standards, where the encoder dictates exactly how a target signal should be constructed at the decoder.

The key differentiator for our proposal in Chapter 2 is that we leverage on the self-discovery power of the decoder, so that the decoder can recover remaining missing pixels in the reconstructed viewpoint image via inpainting procedures. Instead of disparity compensation, our proposal reduces

inter-view redundancy by mapping pixels from a reference view to target view to create a starting image, and then transmits auxiliary information to assist the decoder in the completion of the rest of the image in a RD manner, thus avoiding the aforementioned shortcomings of disparity compensation.

1.3.2 Related Works on Inpainting Assisted Video Coding

Employing inpainting at the decoder to aid compression was first proposed in [61] for 2D images. In a nutshell, the work in [61] essentially advertises an advanced intra-prediction scheme based on (local) inpainting. This inpainting is more sophisticated than uni-directional pixel copy employed in video coding standards like H.264 intra [101], where texture in the target block is predicted using observed pixels from adjacent causal blocks that have already been reconstructed. To further improve inpainting quality, edge information (called assistant information) for the target block is optionally sent, so that sharp edges in the reconstructed block can be preserved. The success of [61] has inspired a set of follow-up works that also employ inpainting for 2D image and video coding [11, 60, 78, 103, 106]. For example, the authors in [103] generalize the notion of assistant information to a set of parameters for different model distributions; *i.e.*, instead of simple edges, other assistant information can be transmitted to aid inpainting. Blocks with statistics that do not fit with model distributions are classified as non-featured blocks and coded using traditional DCT-based methods. The authors in [78] propose an inpainting procedure based on Laplace partial differential equation (PDE) and total variation for HEVC intra coding, and later for depth map coding as well [11]. In [22], towards the goal of efficient free viewpoint video compression, the concept of auxiliary information (AI) is introduced to assist a receiver in completing holes in a DIBR-synthesized secondary view image via sender-assisted inpainting, given texture and depth maps of the primary view.

Though also employing inpainting at the decoder, our work in Chapter 2 differs from these works fundamentally in the following regard: inpainting for intra-prediction as described above exploits *local* inter-pixel data redundancy in images, while our proposed encoder-driven inpainting strategy exploits also data redundancy in *non-local* pixel patches due to self-similarity in images. Specifically, our inpainting scheme is derived from inpainting schemes based on template-matching such as [21] that identify and copy non-local pixel patches from distant spatial regions to the target patch. This concept is similar to our previous work [22], which is significantly extended here in many aspects, as will be discussed in Chapter 2.

1.3.3 Related Works on Geometry Compression

In computer graphics, geometry models are commonly represented by triangles recording information such as vertex positions and connectivity. Recently, billions of triangles are used in 3D

applications like video games, which requires efficient geometry compression to reduce storage and transmission costs.

Previous work on compression of static 3D geometry is extensive. [25] first motivated the geometry compression problem and proposed generalized triangle mesh to solve the problem. [96] proposed to compress triangular meshes directly through prediction of vertices and connectivity. On the other hand, multiresolution progressive compression was studied. The vertex positions are hierarchically predicted, and high compression ratio is achieved [95]. [99] proposed a progressive coding scheme based on over-complete expansions by first projecting a 3D object onto a 3D sphere. The selection of multiple spheres for geometric projection is further optimized in [98].

However, the compression of time-varying meshes (TVM) brings new challenges. In [75, 108, 109], TVM in each frame is first divided into patches, then a patch-based matching procedure is used to exploit inter-frame correlation. The residual information is coded using scalar or vector quantization. In Chapter 4, we pursue an image-based approach, where 3D geometry is first projected to tiles before the tile images are coded using a multiview image codec. An image-based approach means motion compensation well understood in video coding can be directly applied to reduce temporal redundancy.

1.3.4 Related Works on IMVS

Interactive Multiview Video Streaming

The MVC schemes focus on the compression of all viewpoint images as bulk data and did not take user interactivity into consideration. Specifically, in an interactive multiview video streaming (IMVS) scenario [16] where a receiver periodically requests only one out of a large number of available views at a time from the server for playback, the server must encode the multiview video in a sufficiently flexible manner, so that the lone requested view can be transmitted for correct decoding while exploiting inter-view data redundancy (between views already transmitted to user and the newly requested view) to lower transmission rate. Toward this goal, [63] proposed to pre-encode three separate streams to support three most common types of view-switching requested by users. [15] designed a redundant frame structure to optimize the trade-off between storage cost and expected streaming rate, where distributed source coding (DSC) frames [18] were used to facilitate view switching. The study of storage and transmission cost tradeoff was further generalized by considering network delay in [104]. A more recent work [2] optimized frame prediction structure for IMVS to minimize visual distortion subject to both the storage and transmission constraints.

Decoder-side rate-complexity tradeoff in IMVS

In IMVS [15], only the single video view currently selected by a client is transmitted from server, lowering transmission rate. The technical challenge in [15] is to design a frame structure that efficiently compresses multiview video and provides periodic view-switching mechanisms in the encoded bitstream at the same time. [104] extends the work in [15] by considering interactive streaming of free viewpoint video, where texture and depth maps of two coded views that sandwich the virtual view currently selected by a client are transmitted from server, so that the virtual view can be synthesized at client via DIBR. In Chapter 6, following the same IMVS setup in [15, 104], we focus exclusively on the rate-complexity (RC) tradeoff in client-side view synthesis, which is not considered in [104].

To the best of our knowledge, only [67] studies the complexity of DIBR-based view synthesis at decoder in a formal manner. The common assumption in [67] and this work is that overall system computation load (or power consumption) can be reduced by trading off view synthesis complexity with an increase in transmission bitrate. The increase in transmission rate will not cause the same proportional increase in power consumption; this is true for 3G network data transmission, for example, since energy consumption is dominated by the *tail energy* at the end of a transfer, not the actual amount of data transmitted [3]. For LTE, [51] experimentally shows that the power consumption grows slow with increased downlink throughput, and that the energy per bit decreases drastically and down to a very low level if the size of downlink bulk data constantly becomes larger. The FVV applications considered in Chapter 6 involves massive data transmission through downlink to client, and thus those observations could apply to our case. The key idea in [67] is to compute and transmit from server the disoccluded pixels in an DIBR-synthesized image from a virtual viewpoint, so that the client can simply integrate these server-computed pixels without performing any inpainting, but [67] requires pre-encoding and storage of inpainted pixels for every possible virtual view, resulting in a large storage cost.

1.4 Outline and Main Contributions

In this thesis we discuss how to compress multiview videos from different applications. We study not only classical applications such as MVC, but also some new applications such as perceptual MVC, 3D geometry representation and compression, IMVS, and FTV.

The application-specific MVC studied in this thesis can be generally classified into two categories. In the first category, we focus on the rate-distortion (RD) performance, where the distortion can be measured by the mean squared errors (MSE) in Chapter 2, human visual system (HVS)-based MSE in Chapter 3, or the metro distance in Chapter 4. In the second category, we study applications where the real-time computation and associated complexity also need to be considered. We first consider the pre-computation and real-time computation of IMVS at the encoder

(server) side in Chapter 5, and then study the computational complexity of view synthesis for FVV at the decoder (client) side in Chapter 6.

In Chapter 2, we consider the application of FVV, where users can change the viewpoint and the system can synthesize the corresponding view. We propose a novel inpainting assisted approach to efficiently compress multiview videos that support FVV. The inter-view redundancy in multiview videos is removed by projecting a coded reference view to a target view via DIBR. The remaining holes in the target view are filled in a block-by-block manner. The decoder can independently recover some missing data via inpainting without explicit directions from encoder, resulting in lower rate at the same MSE.

In Chapter 3, we study the application of JND-based MVC. JND can be used to conceal imperceptible distortions in video coding, leading to more compact compression. However, the direct calculation of JND maps incurs high complexity, and the problem is aggravated when JND maps for multiple views are needed. We propose to exploit inter-view or temporal redundancy of JND maps to synthesize or predict target JND maps, which are then used to adjust prediction residuals. Experimental results show that the RD performance is improved if the distortion is measure by a HVS-based metric.

In Chapter 4, we study 3D geometry compression, and propose a new 3D geometry representation method. Instead of compressing the 3D geometry directly, we project the geometry of a 3D object to a collection of surrounding tiles, and subsequently encode these tile images using a modified MVC tuned for piecewise smooth signals. The crux of the scheme is the "optimal" placement of image tiles, which yields the tradeoff between rate and distortion. The performance is measured by the convex hull of rate and the metro distance (measuring the reconstruction quality of a 3D mesh), and we show that our tile placement outperforms the naïve tiling.

In Chapter 5, we discuss view switching in IMVS, an application where a network client requests from server a single view at a time but can periodically switch to other views as the video is played back uninterrupted. Given the statistics of user behaviors, we propose the optimal frame structure such that frequently chosen view switches are pre-computed while infrequent ones are computed in real-time upon request.

In Chapter 6, we examine the decoder side computational complexity of view synthesis in the FVV. We propose to optimally tradeoff transmission rate for decoder-side complexity. For DIBR-based view synthesis, we find the optimal subset of intermediate views to code. For a novel inpainting assisted paradigm, we argue that some expensive operations at decoder can be avoided by directly sending intra-coded blocks.

In Chapter 7, we summarize and conclude the research work of this thesis and discuss the outlook of possible extensions.

The results in this thesis have been reported in [34–37, 40–42]. Some other papers have also been published during my PhD studies, but are not included in the thesis, including [33, 38, 39].

1.5 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
MVC	multiview video coding
IMVS	interactive multiview video streaming
FVV	free viewpoint video
FTV	free viewpoint television
3DTV	three-dimensional television
JVT	joint video team
MPEG	moving experts group
VCEG	video coding experts group
JMVC	joint multiview video coding
HEVC	high efficiency video coding
IBR	image-based rendering
DIBR	depth-based image rendering
VSP	view synthesis prediction
HVS	human visual system
JND	just noticeable distortion
AI	auxiliary information
QP	quantization parameter
RD	rate-distortion
GOP	group of picture
GoGOP	group of group of pictures
PWS	piecewise smoothness
GFT	graph Fourier transform
KLT	Karhunen-Lòeve transform
DCT	discrete cosine transform
CABAC	context-adaptive binary arithmetic coding
SP	switching P
DSC	distributed source coding
LDPC	low-density parity check codes
DP	dynamic programming
PDE	partial differential equation
PDF	probability density function
PMF	probability mass function

MSE	mean-squared-error
PSNR	peak-signal-to-noise ratio
SSIM	structural similarity
BD	Bjontegaard Delta

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 $\mathbf{A}$
$\mathbf{A}^+$	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

Encoder-Driven Inpainting Based MVC for FVV

In this chapter, we consider free viewpoint video (FVV) systems, where a user has the freedom to select a virtual view from which an observation image of the 3D scene is rendered, the scene is commonly represented by texture and depth images from multiple nearby viewpoints. In such representation, there exists data redundancies across multiple dimensions. It is important to exploit the redundancies for effective multiview video compression. Existing schemes attempt to eliminate them via the traditional video coding paradigm of hybrid signal prediction / residual coding; typically, the encoder codes explicit information to guide the decoder to the location of the most similar block along with the signal differential. In this chapter, we argue that, given the inherent redundancies in the representation, the decoder can often independently recover missing data via inpainting without explicit directions from encoder, resulting in lower coding overhead. Specifically, after the pixels in a reference view are projected to a target view via depth image based rendering (DIBR) at the decoder, the remaining holes in the target view are filled via an inpainting process in a block-by-block manner. We implemented our encoder-driven inpainting strategy as an extension of High Efficiency Video Coding (HEVC). Experimental results show that our coding strategy can outperform comparable implementation of HEVC by up to 0.8dB in reconstructed image quality.

2.1 Introduction

In recent multiview video systems, texture maps (RGB images) and depth maps[1] (per-pixel distance between captured objects and capturing camera) of the 3D scene as observed from multiple closely spaced cameras are captured at the encoder into a texture-plus-depth representation [69]. Armed with the texture and depth images of multiple views, images of any arbitrary intermediate viewpoint can also be constructed at the decoder via depth-image-based rendering (DIBR) [97] in order to enable new applications such as free viewpoint video (FVV) [94].

It is apparent that this texture-plus-depth representation contains data redundancy across multiple dimensions. First, a voxel of an object in the 3D scene that is visible from multiple camera-captured views will be represented as different pixels in multiple viewpoint images (inter-view redundancy). Assuming that the 3D scene is Lambertian[2], a 3D voxel reflects the same color value to different viewpoints, and recording its value many times in multiple viewpoint images leads to redundant information. Second, it is well understood that values of neighboring pixels of the same object in a viewpoint image tend to be correlated statistically (inter-pixel redundancy). Finally, it is observed that natural images tend to be self-similar. In other words, similar image patches tend to recur in different spatial regions throughout the same image (inter-patch redundancy). Previous computer vision research efforts have successfully exploited this nonlocal self-similarity characteristic of images for denoising [8] and inpainting [21] (completion of missing pixels in a spatial region).

For an encoder to compactly describe information in a texture-plus-depth representation of a 3D scene, it is critical to exploit these inherent redundancies in the data representation. The vast majority of conventional coding schemes [31, 70, 84] attempt to eliminate this redundancy following the traditional video coding paradigm of hybrid signal prediction / residual coding, as is done in video coding standards like H.264 [101] and HEVC [89]. Specifically, to reconstruct a given target block, a sender transmits explicit instructions like motion vector (MV) to guide the receiver to the location of the most similar block, which serves as a predictor signal. Then, the difference between the predictor and target block, namely the prediction residual, is transform coded and transmitted to the receiver to improve the quality of the reconstructed signal. This paradigm has a long legacy in video coding research, dating back to the first ISO video coding standard MPEG1 and ITU-T standard H.261 in the late 1980's, where one of the crucial design criteria was a computationally inexpensive video decoder. In that light, the hybrid signal prediction / residual coding paradigm where the encoder dictates exactly how each code block should be reconstructed is a suitable design choice that results in today's many practical codecs across many platforms.

Given that the cost of computation has drastically decreased, the strict requirement that the

[1] Depth images can be acquired directly using depth sensors [44], or computed from neighboring color images using stereo-matching algorithms [92].

[2] Reflective surfaces such as wine glasses and mirrors are not Lambertian. However, for closely spaced capturing cameras, the Lambertian surface assumption is nonetheless a good approximation.

video decoder must be computationally simple is no longer necessary in many practical cases. Thus, in this chapter we argue that one can leverage on the computation power of a decoder to recover a desired signal and lower the overall transmission cost. In particular, the decoder can often independently recover missing pixels via inpainting without explicit directions from encoder, resulting in lower coding overhead. Therefore, we propose the following encoder-driven inpainting strategy for the texture-plus-depth representation. First, we directly project pixels from one reference view to a target view via DIBR, thus eliminating inter-view redundancy. This projection results in a number of disocclusion holes, namely spatial areas that are occluded in the reference view by foreground objects but become exposed after the view change. However, assuming that the viewpoints are close, these holes are relatively small. To complete these holes, we first order the blocks with missing pixels in terms of decreasing difficulty for inpainting. For the most difficult blocks, we then transmit explicit instructions called auxiliary information (AI) to guide the decoder in the block reconstruction process. AI typically consists of the location information of the best predictor block for inpainting, and color values for missing pixels. The incomplete blocks typically contain known pixels projected from neighboring view via DIBR as well as missing pixels, but only the missing pixels are explicitly coded via a graph Fourier transform (GFT) [82] or a sparsification procedure using the discrete cosine transform (DCT) [12, 13], in order to achieve low coding cost. Finally, the decoder can independently complete missing pixels in the blocks that are easy to inpaint via a template-matching algorithm such as [21]. We believe that our new coding strategy opens the way to new coders where decoder's capabilities are judiciously used to lower transmission costs.

The outline of the chapter is as follows. We first overview our encoder-driven inpainting based coding system in Section 2.2. We describe our design of AI used to guide the inpainting of disocclusion holes at decoder in Section 2.3. The methods for sparsification of DCT and GFT of code blocks are described in Section 2.4. The order in which the missing holes are completed is crucial in our proposal; we show that finding the optimal filling order is an NP-hard problem and present a heuristic "hard-to-easy" order in Section 2.5. Finally, we discuss our experimentation and conclude in Section 2.6 and 2.7, respectively.

2.2 Coding System Overview

2.2.1 System Overview

We propose a coding strategy based on encoder-driven inpainting for texture-plus-depth representation of multiview video data. Specifically, the objective of our coding strategy is to code texture and depth image pairs of multiple views in a rate-distortion (RD) optimal manner. For the sake of simplicity, we describe our coding strategy for two neighboring viewpoint images (two pairs of texture and depth images). The application of our strategy to more complex frame prediction structure

is straightforward.

We first independently code texture and depth images of one view with the help of a video codec; this view is called the *independent view*. Then we propose to code the second view (the *dependent view*) as follows. We project the decoded texture pixels of the independent view to the dependent view image via DIBR, where the geometric information provided by the depth map of the independent view and the camera calibration parameters are used to identify pixel correspondences between the two views. After that projection step, there are two kinds of holes in the DIBR-synthesized image that require filling in order to have a good reconstruction of the dependent view; these are: i) *rounding holes*, and ii) *disocclusion holes*. First, the disparity values of the pixels in the independent view likely have fractional values; when projected to the dependent view they need be rounded to the nearest integers to land on the 2D grid of the rendered image. This rounding operation can result in rounding holes. By assuming a rectified camera setup where the relative camera positions correspond to pure horizontal shifts, the pixel disparities have only horizontal components. Thus we can identify the rounding holes simply as follows. For a given hole in the projected image, we check if the nearest available pixels to the left and right of the hole have almost identical depth values (namely, less than a pre-defined threshold δ). If so, we identify the hole as a rounding hole and perform simple horizontal linear interpolation to fill in the missing pixel(s).

After identification and filling of the rounding holes in the projected image, only disocclusion holes remain. Disocclusion holes represent spatial regions in the dependent view that are not visible in the independent view due to occlusion by foreground objects. An example of such disocclusion holes is shown in Fig. 2.1 [3]. Unlike rounding holes, disocclusion holes may contain novel information that cannot be easily extrapolated from available neighboring pixels. Hence, the encoder has to provide information to the decoder so that it can properly reconstruct the dependent view. In particular, it has to code additional information to help the decoder fill in the disocclusion holes in the projected image in a RD optimal manner. In this chapter, we assume that the decoder has the computational resources to execute inpainting procedures. Thus the encoder only provides carefully chosen *auxiliary information* (AI) to guide the decoder through the reconstruction of difficult spatial regions, so that the decoder can self-discover missing pixels in the remaining holes in the dependent view via inpainting. The construction of this AI data is described in the next section.

The depth pixels of the independent view are also projected to the dependent view, and rounding holes are identified and filled in the same manner as in the texture image[4]. However, the disocclusion holes are simply extrapolated using adjacent background depth pixels. This is because depth images are known to be piecewise smooth [50, 82]. We further find empirically that adjacent background depth pixels are good predictors for signal extrapolation into the disocclusion holes. The

[3] Video sequence `Undo_Dancer` can be found from ftp://mpeg3dv.research.nokia.com

[4] The corresponding depth values for the two views are the same if we assume the two viewpoints are rectified [46].

(a) view 1 (b) original view 5 (c) DIBR estimation of view 5

Figure 2.1: Illustration of disocclusion holes left after DIBR, in a view from the video sequence Undo_Dancer. The forward warping is performed from view 1 to view 5. The black regions close to the right side of the foreground (the man) show the missing pixels that have to be filled after projection.

overall procedure of the proposed coding strategy is summarized in Fig. 2.2.

2.2.2 Encoder-driven Disocclusion Hole Filling

We provide now more details about our original coding strategy that relies on inpainting to fill in disocclusion holes. In the computer vision literature, there exist many inpainting [4, 5, 9, 21, 77] to complete holes in a given image. The key difference between our work and inpainting schemes like [21] is that, in our multiview coding scenario, the target image to be inpainted is known at the encoder. Hence, the encoder can provide additional information to guide the image completion process at the decoder. Inspired by the patch-based template-matching inpainting algorithm in [21], we will employ a similar inpainting framework and complete the rendered image on a per-patch basis. In a nutshell, our patch-based inpainting framework performs the following operations. We first select a pixel on the boundary of disocclusion holes in the DIBR projection of the dependent view; the selected pixel is the center of a target patch that will be inpainted. Missing pixels in the target patch are then filled using known pixels in the reconstructed dependent view via template-matching, possibly with help of AI provided by the encoder. Then, another target patch is selected for filling, and the process continues until all missing pixels are completed. The order in which the patches are selected for filling is called the *patch order*. Given this patch-based inpainting framework, there are two key questions to solve for effective coding performance: i) for a given target patch, how to best complete it, possibly with the aid of AI? ii) what is the optimal patch order

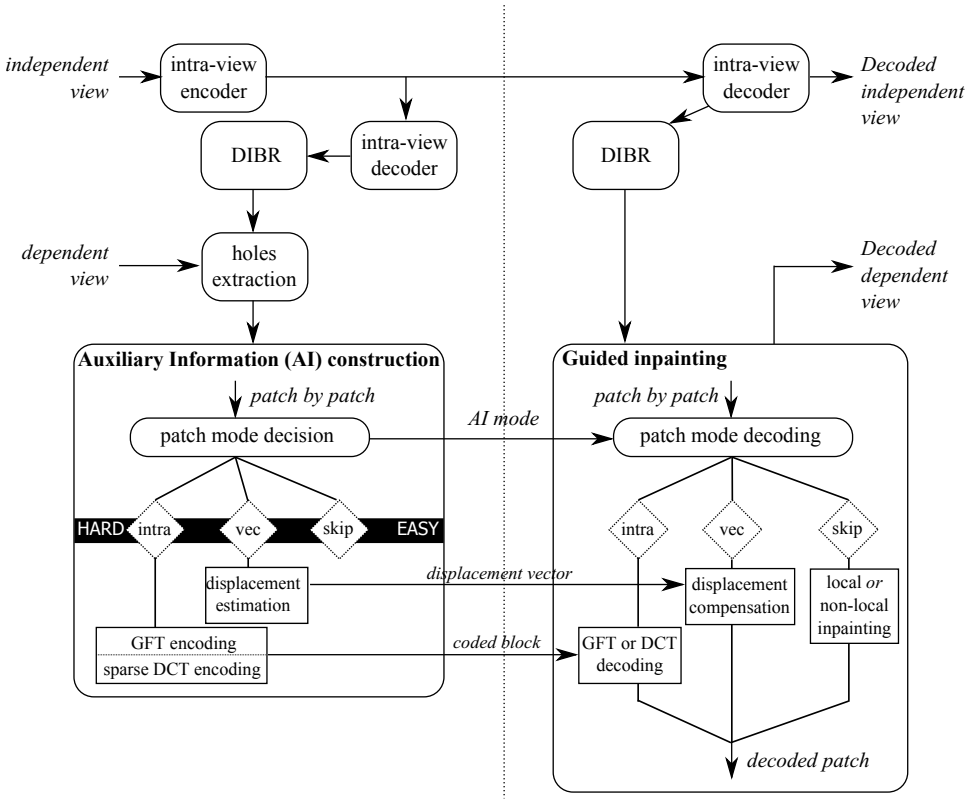


Figure 2.2: The block diagram of the proposed strategy. Dependent views are obtained by DIBR estimation from independently coded views, along with encoder-driven inpainting of the disocclusion holes.

to complete the rendered image?

We first observe that, given a local target patch to be completed, the level of difficulty in inpainting it—called *local hardness* in the sequel—depends on the degree of self-similarity exhibited between the target patch and already completed spatial regions in the predicted image. If the missing pixels in the target patch can be found in the completed spatial regions, then the encoder only needs to convey a simple message (called the *skip* mode) to the decoder to signal that the missing information can be self-discovered using an unassisted inpainting procedure such as template-matching [21]. If the missing pixels are available in the completed spatial regions but it is difficult for the decoder to discover them unassisted, then the encoder can provide lightweight search instructions (called the *vec* mode) to guide the discovery process. Finally, if the missing pixels are entirely innovative, then the encoder has no choice but to explicitly encode the missing pixels (called the *intra* mode). Note that *only the subset of missing pixels in a target block requires intra coding*. This observation can be exploited for novel patch-based transform coding that can outperform conventional scheme like DCT, which typically codes the entire code block, regardless of whether pixels in the block are missing or not. These different kinds of information compose the set of AI that the encoder convey to the decoder on a patch-by-patch basis for encoder-guided inpainting. Finally, we remark that the different kinds of AI have different coding costs, and the choice of AI is determined by a RD criterion. The details of the AI design is discussed in Section 2.3 and the patch-based transform coding is described in Section 2.4.

The second question is related to the order of patches in the inpainting process. Clearly, a left-to-right top-to-bottom raster scanning order employed in conventional block-based image / video coding algorithms is not appropriate. A key innovation in Criminisi's inpainting algorithm [21] is the order in which target patches should be selected for inpainting: the main idea is to select easy-to-inpaint patches first, so that propagation of likely errors in hard-to-inpaint patches to other regions will be minimized. This patch ordering problem is called the *global hardness* of patches in the inpainting process. In stark contrast to the "easy-to-hard" patch order in Criminisi's algorithm, we propose a "hard-to-easy" patch order for our encoder-assisted inpainting algorithm. The basic idea is that, once the hardest patches are filled in (with ample assistance from the encoder), the remaining patches are all easy-to-inpaint. They can be completed by the decoder unassisted, and hence the encoder can directly save bits from reduction in AI signaling cost. Note that the problem of error propagation from hard-to-inpaint patches to other spatial regions can be easily contained in our setting, since the encoder-guided inpainting process can implicitly control the inpainting quality at the decoder. The details of our proposed "hard-to-easy" patch order is presented in Section 2.5.

2.3 Inpainting Based Coding And Auxiliary Information

Towards a solution to address local hardness in encoder-driven image inpainting, we first overview a well-known template-matching inpainting algorithm [21] as we use a similar framework in our system. We then discuss the design and implementation of auxiliary information (AI).

2.3.1 Encoder-driven Patch Completion

Given a target patch Ψ_p, we now discuss our encoder-driven patch completion procedure using AI to fill in the missing pixels in Ψ_p. In a nutshell, we seek to design a set of AI modes $\{\varphi\}$ for the completion of missing pixels in the target region and to eventually choose for each target patch Ψ_p the AI mode that minimizes the RD cost, *i.e.*,

$$\varphi^* = \arg\min_{\varphi} \; d\,(p,\varphi) + \lambda \cdot r\,(p,\varphi), \tag{2.1}$$

In (2.1), $r(p,\varphi)$ is the coding rate of the mode φ for patch Ψ_p centered at p, $d(p,\varphi)$ is the distortion between missing pixels in $\Psi_p \cap \Omega$ and the reconstructed pixels using mode φ, and λ is a pre-defined weighting parameter that trades off the importance of distortion and rate. We use sum of squared differences as distortion in this chapter. The index of the coding mode in (2.1) is compressed via a context-based arithmetic coder.

In this chapter, we design three AI modes with different coding costs $r(p,\varphi)$ and different degrees of influence on patch reconstruction. The AI "skip" mode results in zero rate beyond the signaling cost of the mode itself. AI "vec" mode encodes a motion or disparity vector to inform the decoder the location of the best matching patch in the known region. The AI "intra" mode encodes the intensity of missing pixels in the target patch, and thus usually results in the highest rate. The encoder chooses among these three modes for a given target patch Ψ_p in order to solve the optimization problem in (2.1). We describe in details the three coding modes in the rest of this section.

2.3.2 AI Modes

AI "skip"

The AI "skip" mode instructs the decoder to self-discover missing pixels in Ψ_p using only information in source region Φ. This can be done either *locally* or *nonlocally*. *Local skip* means that, given the strong inter-pixel redundancy exhibited in the local patch, the missing pixels in $\Psi_p \cap \Omega$ can be deduced from neighboring known pixels via simple signal extrapolation schemes such as [5]. In contrast, *nonlocal skip* instructs the decoder to perform template matching to complete missing pixels in the target patch Ψ_p using its template $\Psi_p \cap \Phi$. This is similar to the template matching in [21], except that the search region includes not only the known region Φ in the same image,

but also designated decoded pictures in the decoder's buffer. Designated pictures could include L_{T} previous decoded pictures in time from the same view and L_{V} decoded pictures of the same time instant but from neighboring views. For simplicity, only the decoded picture of previous time instant from the same view is used as reference in our chapter. Local skip and nonlocal skip finally translate to different AI mode indices for arithmetic encoding.

The AI "skip" mode is expected to be a dominant mode at low rate when coding rate is of higher priority than distortion. At high rate, however, because of the lack of direct control in the pixel completion process, the patch quality reconstruction is limited. We present next two other modes with more direct control on the inpainting results, hence on the reconstruction quality.

AI "vec"

When the template matching of "nonlocal skip" fails to identify a good match for $\Psi_p \cap \Omega$, we offer the AI "vec" mode as an alternative to improve the reconstruction quality by directly locating the pixels in the known region that are the most similar to the missing pixels in the target patch. We stress here the difference between "nonlocal skip" and "vec". "Nonlocal skip" relies on the known pixels in the target region of Ψ_p for template-matching with known patches, which may not always lead to the best completion results. The "vec" mode, on the other hand, simply informs the decoder about the location of the pixels in the known region that are the most similar to the missing pixels in the target patch; it does not rely on template-matching at all.

To leverage on both self-similarity of still image and temporal redundancy of video sequences, we propose two kinds of "vec" mode, namely intra-frame AI "vec" and inter-frame AI "vec". For intra-frame "vec", a *similarity vector* (SV) pointing to the known region in the same image is signaled to the decoder. On the other hand, the inter-frame "vec" is akin to motion estimation in differential coding for single-view video: a *motion vector* (MV) is used to represent the displacement of the current block to the most similar one in a reference picture (*i.e.*, the previous frame in time in the same view).

AI "intra"

When no pixels similar to $\Psi_p \cap \Omega$ are found in the search space of nonlocal "skip" and "vec" modes, *e.g.*, in the case of disocclusion of novel objects, the AI "intra" mode is used to directly code the missing pixels in the target patch Ψ_p. In this mode, the block is predicted, transformed, quantized, and entropy-coded. Since the shapes of known pixels $\Psi_p \cap \Phi$ and causal neighbors of Ψ_p are arbitrary, the directional intra prediction used in conventional block-based video coding such as HEVC [90] is not suitable here. Instead, we propose to use the signal extrapolation scheme in AI "local skip" as the prediction, and to code only the resulting prediction results. Another noticeable deviation from conventional block-based video coding is that, in our scenario, only missing pixels in

a block requires coding and not the full block of pixels. We discuss in Section 2.4 two methods to encode only the missing pixels in a square target block for AI "intra".

Arithmetic Coding of AI Mode Indices

The coding modes are compressed with arithmetic coding. Let us consider a set of AI mode decisions $M = \{m_i\}_{i=1..N}$ for N consecutive patches. In order to transmit this vector M, we use an arithmetic coder, which needs the mode probabilities as input, both at the encoder side and decoder side. As mentioned above, we have designed an encoder-driven inpainting algorithm that adopts the "hard-to-easy" order. It means that "hard" modes such as AI "intra" or "vec" are chosen more frequently at the beginning, while "easy" mode AI "skip" is likely chosen at the end of the coding process. In order to take into account this evolution in the coding of M, we adapt the input probabilities of the arithmetic coder. For a given mode flag m_i to code, we evaluate the probabilities $p_{i,l}$ of each coding mode $l = 1, ..., L$ over the W last mode decisions

$$p_{i,l} = \frac{\sum_{j=1}^{W} b_{i-j,l}}{W}, \tag{2.2}$$

where

$$b_{i-j,l} = \begin{cases} 1 & \text{if } m_{i-j} = l \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

The L probabilities, available at the decoder side also, are the contexts of our arithmetic coding of mode indices. Note that we code the mode index directly and do not have a binarization process.

2.4 Transform Coding of Missing Patch Pixels

In our proposed patch-based coding scheme, the center p of a $K \times K$ target square patch Ψ_p is always on the boundary of known and unknown regions as shown in Fig. 1.3. Hence, the patch Ψ_p contains known pixels in $\Psi_p \cap \Phi$ as well as missing pixels in $\Psi_p \cap \Omega$. If one naïvely use regular block-based DCT to encode the patch (or the prediction residual of the patch), then the resulting $K \times K$ transform coefficients will contain information belonging to both known and unknown pixels, resulting in undesirable representation redundancy. In this section, we propose two block-based coding procedures to encode only the missing pixels in a patch, namely i) the graph Fourier transform (GFT), and ii) the sparsification of DCT.

In the literature, shape-adaptive DCT [87] is suitable for coding pixels in arbitrarily-shaped image segments. The transformation is computed by cascaded application of 1-D varying-length DCT transforms, after geometrical shifts in vertical and horizontal directions. The GFT employed in this chapter, however, has better decorrelation than the shape-adaptive DCT, due to the preservation of correlation among neighboring pixels by maintaining their relative positions. Different from the

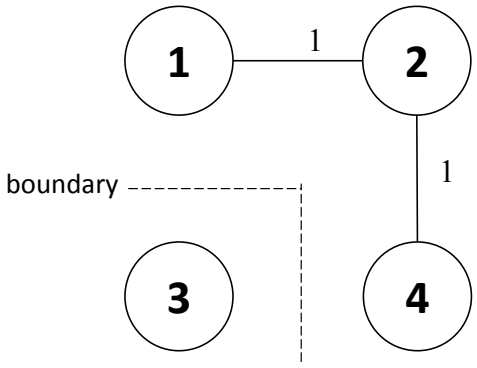


Figure 2.3: An example graph from a 2-by-2 patch.

shape-adaptive DCT, the proposed sparsification of DCT is compatible to regular inverse DCT, because the sparsification procedure performed at the encoder only adjusts the values of some DCT coefficients. Moreover, the shape-adaptive DCT is not good at coding noncontiguous object pixels, *e.g.*, holes, which are commonly seen in the rendered images.

2.4.1 Graph Fourier Transform

GFT has recently been proposed for transform coding of a block in a piecewise smooth image like a depth map that straddles sharp boundaries, so that filtering across discontinuities is avoided; this results in a sparser signal representation in transform domain than DCT [50, 83]. The key idea is to represent pixels in the block as nodes in a graph $\mathcal{G}$, and connect each pixel with each of its four neighbors with an edge of weight 1 only if both pixels reside on the same side of a boundary. In essence, a block of pixels is divided into connected sub-graphs, as illustrated in Fig. 2.3.

The graph $\mathcal{G}$ is described by a few matrices. First, an *adjacency matrix* $\mathbf{A}$ describes the connectivity of the graph $\mathcal{G}$, where $A_{i,j} = 1$ if node i and j are connected via an edge and 0 otherwise. For the graph in Fig. 2.3, the adjacency matrix is

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2.4}$$

A *degree matrix* $\mathbf{D}$ is a diagonal matrix, where $D_{i,i} = \sum_j A_{i,j}$. For the graph in Fig. 2.3, the degree

matrix is

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

Finally, the *graph Laplacian* matrix $\mathbf{L}$ is defined as the difference between the degree matrix and the adjacency matrix [86]:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2.6}$$

For the graph in Fig. 2.3, the graph Laplacian matrix is

$$L = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}. \tag{2.7}$$

GFT is then defined as follows. It is a linear transform matrix Φ composed of the eigenvectors of $\mathbf{L}$, *i.e.*, $\mathbf{L}\phi_i = \lambda_i \phi_i$, where ϕ_i is the i-th row of Φ written as a column vector, and λ_i is the i-th eigenvalue of Φ, which could be seen as the i-th *graph frequency* of the graph $\mathcal{G}$. For the simplified example,

$$\Phi = \begin{bmatrix} 0.4082 & -0.8165 & 0 & 0.4082 \\ -0.7071 & 0 & 0 & 0.7071 \\ -0.5774 & -0.5774 & 0 & -0.5774 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.8}$$

And the corresponding eigenvalues are $3, 1, 0, 0$.

A given pixel block $\mathbf{x}$ is then interpreted as a graph-signal on graph $\mathcal{G}$; after computing GFT coefficients $\alpha = \Phi\mathbf{x}$, the coefficients α are quantized and entropy-coded. Unlike block transforms such as DCT where the same transform is applied for every pixel block, GFT is an *adaptive* transform; *i.e.*, different signal-dependent transforms Φ are used for different input graph-signals $\mathbf{x}$, since the graph construction is dependent on the signals. Previous works [50, 82] have shown that this overhead of encoding side information to describe GFT is not expensive for depth maps, and there is overall substantial coding gain for GFT over fixed transforms like DCT for coding depth maps.

Based on the success of GFT to code depth maps, we propose here to use GFT to encode only the missing pixels $\Psi_p \cap \Omega$ in a given patch Ψ_p. We first construct a graph $\mathcal{G}$ only for these missing pixels: each missing pixel in $\Psi_p \cap \Omega$ is denoted by a node, and there is an edge of weight 1 connecting two nodes if they represent two missing pixels that are neighbors. See Fig. 2.4 for an illustration. In this way, the graph is composed only of nodes that represent the n missing pixels.

Given this graph of missing pixels, one can compute the adjacency and degree matrices $\mathbf{A}$ and $\mathbf{D}$ accordingly. The graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ can also be computed, and its eigen-decomposition

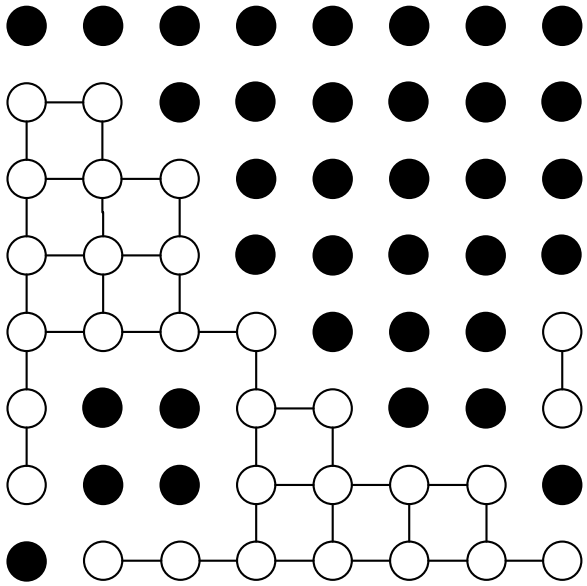


Figure 2.4: An example of graph construction for an 8×8 patch. White circles denote unknown pixels. Black circles denote known pixels. Edges connect neighboring unknown pixels. The weights assigned to every edges are unity.

can be performed to obtain the GFT operator Φ. To encode missing pixels $\mathbf{x}$ (stacked together as a vector), we simply compute the GFT coefficients $\alpha = \Phi\mathbf{x}$, and quantize and entropy encode them into bits. Note that, unlike the classical DCT that has $K \times K$ transform coefficients for $K \times K$ pixels in the patch Ψ_p, the number of GFT coefficients is only equal to the number of missing pixels in Ψ_p. Since the locations of the missing pixels are known at decoder already, and a graph can be readily constructed, and the operator Φ can be computed to permit reconstruction of the missing pixels. We will show in Section 2.6 that usage of GFT to encode missing pixels in a patch can outperform DCT in coding performance.

2.4.2 Sparsification Procedure using DCT

We introduce here another option to encode missing pixels in a patch by sparsification of the DCT coefficients. While the GFT leads to good coding performance, the complexity required to compute the GFT via eigen-decomposition both at the encoder and decoder can be high, especially, if the number of missing pixels is large. Compared to the GFT, the DCT sparsification procedure is less complex.

Since the values of rendered pixels in a patch are known at encoder and decoder prior to any transform encoding, the known pixels in $\Psi_p \cap \Phi$ can be viewed as degrees of freedom at encoder side: they can be manipulated in order to reduce the cost of coding the patch Ψ_p as their decoded values are simply discarded. Specifically, we propose a sparsification procedure in the DCT domain

that exploits these degrees of freedom to minimize the number of non-zero DCT coefficients. The percentage of non-zero quantized transform coefficients has an approximately linear relationship with bitrate [48], hence minimizing the number of non-zero coefficients corresponds to reduce the coding rate.

Let $\mathbf{x}$ be pixels in a K-by-K patch, stacked into a column vector. Let Θ be the DCT transform; the DCT coefficients $\mathbf{y}$ can be computed simply: $\mathbf{y} = \Theta\mathbf{x}$. Let $\mathbf{V}$ be a K^2-by-K^2 diagonal matrix where entry $V_{i,i} = 1$ if the i-th pixel in $\mathbf{x}$ is an unknown pixel and 0 otherwise. Our objective is to minimize the rate-distortion cost of AI "intra" for the patch Ψ_p by manipulating the coefficients $\mathbf{y}$ in the transform domain

$$\min_{\mathbf{y}} \ \|\mathbf{V}(\Theta^{-1}\mathbf{y} - \mathbf{x})\|_2^2 \ + \ \lambda \ \|\mathbf{y}\|_0, \tag{2.9}$$

Note that $\lambda = \lambda_{\text{mode}} * B$, the product of the Lagrange multiplier used in AI mode decision (λ_{mode}) and the average bits to code a non-zero quantized coefficient (B), and hence $B\|\mathbf{y}\|_0$ is the coding rate of the patch. In Eq. (2.9), the l_0-norm is a rate proxy, while the l_2-norm is a measure of distortion, counting only the distortion contributed by the unknown pixels.

The minimization of the l_0-norm in (2.9) is in general a non-convex and NP-hard problem. For efficient computation, one can use an iterative re-weighted least squares algorithm and replace the l_0-norm with a *sparsity-promoting* weighted l_2-norm [12]:

$$\min_{\mathbf{y}} \ [\mathbf{V}(\Theta^{-1}\mathbf{y} - \mathbf{x})]^{\mathrm{T}}[\mathbf{V}(\Theta^{-1}\mathbf{y} - \mathbf{x})] + \mathbf{y}^{\mathrm{T}}\mathbf{W}_{\lambda}\mathbf{y}, \tag{2.10}$$

where the weight matrix is

$$\mathbf{W}_{\lambda} = \begin{pmatrix} \lambda w_1 & 0 & \cdots & 0 \\ 0 & \lambda w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda w_{K^2} \end{pmatrix}, \tag{2.11}$$

where $\{w_1, w_2, \cdots, w_{N^2}\}$ are iteratively updated. Intuitively, w_i for the i-th DCT coefficient y_i is chosen such that $w_i y_i^2$ is approximately equal to 1 if y_i is non-zero. In this way, as a surrogate, the weighted l_2-norm is roughly equivalent to the l_0-norm, leading to a sparse solution.

The optimal solution to Eq. (2.10) can be found by solving the linear system

$$(\Theta\tilde{\mathbf{V}}\Theta^{-1} + \mathbf{W}_{\lambda})y^o = \Theta\tilde{\mathbf{V}}\mathbf{x}, \tag{2.12}$$

where $\tilde{\mathbf{V}} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{V}$.

Iteratively updating the weights in $\mathbf{W}_{\lambda}$ and solving the linear system in (2.12) can achieve transform domain sparsity [12] [24] and minimum rate-distortion cost given λ. The detailed procedure is written in Algorithm 1. After the algorithm converges, the optimal transform coefficients are quantized and entropy coded. Finally, we note that the parameter τ in the iterative weight computation process (see Algorithm 1) can control the speed of the algorithm, *i.e.* the complexity of encoder, while at decoder side we only need to inverse-transform the received coefficients.

Algorithm 1 Iterative re-weighted least squares (IRLS) for transform domain sparsification

1. Initialize weights: $w_i = 1/(|y_i^t| + \epsilon)^2$, where $y_i^t = (\Theta \mathbf{x})_i$, the i-th transform coefficient of the ground truth input signal x.
2. Find the solution to Eq. (2.12): y_i^o.
3. Update weights: $w_i = (|y_i^o|^2 + \epsilon^2)^{-\frac{2-\tau}{2}}$ if $|\frac{y_i^o}{Q}| \geq 0.5$; $w_i = \epsilon^{2-\tau}$ otherwise, where Q is quantization step.
4. Repeat Step 2 to 3 until convergence.

2.5 "Hard-to-Easy" Order for Target Patch Selection

In this section, we address the following problem: how target patches Ψ_p in the target region Ω should be selected for completion. The order of target patches to be filled can be denoted by a sequence of positions p on the boundary between known and target regions, *i.e.*, $p \in \delta\Omega_t$, where Ω_t denotes the target region that contains all missing pixels at iteration t, until all missing pixels in the image are filled. Our goal is to find a patch order so that the overall RD cost to fill all missing pixels in Ω_0 is minimized. The total number of possible orders is, in the worst case, exponential in the number of missing pixels in Ω_0, so clearly an exhaustive search for the optimal order is not practical.

We discussed earlier that the Criminisi's inpainting algorithm [21] proposed an "easy-to-hard" order for patch selection to minimize the chance of error propagation from hard-to-fill patches to other spatial regions. Given that the encoder can transmit auxiliary information to guide decoder in completing missing pixels, the error propagation from hard-to-fill patches can be contained proactively. Hence, Criminisi's order is clearly not an optimal order[5] in the general case. In this section, we first show that finding the optimal order is an NP-hard problem. We then propose our heuristic "hard-to-easy" order, which can be computed in polynomial time.

2.5.1 NP-Hardness Proof for Patch Selection Order

In the most general setting, the optimal patch ordering problem can be formulated as follows. Let $\mathbf{P}_t = \{p_t, \ldots, p_1\}$ be the first t selected patch centers, and let $\Upsilon_t = \{\varphi_t, \ldots, \varphi_1\}$ be the t selected AI modes for the first t selected patches $\mathbf{P}_t$. Assuming that it requires T selected patches before all the missing pixels are filled in the initial target region Ω_0, the optimal patch order, expressed in patch centers and AI modes $\mathbf{P}_T^*$ and Υ_T^*, is defined as:

$$(\mathbf{P}_T^*, \Upsilon_T^*) = \arg\min_{\mathbf{P}_T, \Upsilon_T} \sum_{t=1}^{T} d\left(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1}\right) \; + \; \lambda \cdot r\left(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1}\right) \tag{2.13}$$

[5] Interestingly, one can argue that at zero rate, the Criminisi's "easy-to-hard" order is a good solution in RD performance. We will in fact show our proposed order defaults to the Criminsi's order when the rate constraint is extremely tight.

where $d(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1})$ and $r(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1})$ are respectively the distortion and rate of completing the patch centered at p_t using mode φ_t, given previous selected patch centers and modes $\mathbf{P}_{t-1}$ and Υ_{t-1}. The selected patch centers and modes $\mathbf{P}_T^*$ and Υ_T^* must satisfy two conditions. First, each center p_t must lie on the boundary $\delta\,\Omega_t$, where the target region of missing pixels Ω_t for each iteration t is updated as follows:

$$\Phi_t = \Phi_{t-1} \cup \Psi_{t-1}, \qquad \Omega_t = I \setminus \Phi_t \tag{2.14}$$

In words, the known region Φ_t in iteration t is updated with completed pixels in patch Ψ_{t-1}, and Ω_t is the target region of remaining missing pixels.

Second, by completing all T patches, there should be no remaining pixels:

$$\Omega_{T+1} = \emptyset \tag{2.15}$$

The optimal patch order problem in (2.13) is hard for two reasons. First, the distortion term $d(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1})$ depends on the history of previously selected modes Υ_{t-1}, where each mode φ_t can be selected from a discrete AI mode set $\{\varphi\}$. This means that the total number of these distortion terms $d()$ is *at least* on the order of $|\{\varphi\}|^T$, *i.e.*, exponential in the number of selected patches T. Thus, the time required just for data collection of these terms is time-consuming. This is analogous to the dependent quantization problem for video coding [79], where the distortion $d_t(Q_t \,|\, Q_{t-1}, \ldots, Q_1)$ of a differentially coded frame t depends on not only its own quantization parameter (QP) Q_t, but also QPs of all previous frames in the dependency chain as well.

Second, we have the difficulty of choosing an appropriate patch order for mode selection in our problem. This means that, in addition to the set of patch centers $\mathbf{P}_{t-1}$ selected in previous iterations, the order in which these patch centers have been selected also influences the rate term $r(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1})$ and the distortion term $d(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1})$. To illustrate this second difficulty, let us consider the simple case where the rate term $r(p_t, \varphi_t \,|\, \mathbf{P}_{t-1}, \Upsilon_{t-1})$ depends *only* on the current patch center and the lone previous patch center, *i.e.*, $r(p_t \,|\, p_{t-1})$. This corresponds to the case where the location of the next patch center p_t is differentially coded from the previous center p_{t-1}, while the AI mode coding cost is negligible, resulting in a rate cost $r(p_t \,|\, p_{t-1})$. We will assume that the rate cost for the first patch center $r(p_1)$ is the same for all centers, and therefore can be safely ignored in the optimization. To further simplify our complexity analysis, we also assume that the distortion cost is negligible; this will correspond to the case when λ in (2.13) is set so large that the rate term dominates. We now show that even in this special case, the optimal patch order problem is NP-hard via a reduction from a well-known NP-hard problem—the *traveling salesman problem* (TSP) [43].

TSP is formulated as follows. There exists a finite set of cities $C = \{c_1, \ldots, c_M\}$ and a distance $l(c_i, c_j)$ between each pair of cities $c_i, c_j \in C$. The question is how to find a tour of all the cities,

$\pi = \{\pi(1), \ldots \pi(M)\}$, where $\pi(m) \in C$, such that the total distance traveled is minimized:

$$\pi^* = \arg\min_{\pi} L(\pi) = \sum_{i=1}^{M-1} l(c_{\pi(i)}, c_{\pi(i+1)}) + l(c_{\pi(M)}, c_{\pi(1)}) \tag{2.16}$$

TSP remains NP-hard if we do not require a cycle and remove the last distance term $l(c_{\pi(M)}, c_{\pi(1)})$.

We now argue that the above simple case of patch selection includes TSP as a special case. First, we construct M non-overlapping patches that require separate filling in the target region Ω_0; each patch i will correspond to a city $c_i \in C$ in the TSP problem. Then we set the rate cost $r(i \mid j)$ of selecting patch center i after previous selection of patch center j, as well as the reverse $r(j \mid i)$, to be $l(c_i, c_j)$ in TSP. It is now clear that the optimal patch order in our simplified problem—one that minimizes the total rate $\sum_{t=2}^{M} r(p_t \mid p_{t-1})$—maps to a minimum distance tour in TSP. Hence, our optimal patch order problem is at least as hard as TSP, which means that our optimal patch order problem is actually NP-hard.

2.5.2 "Hard-to-Easy" Order

Given that the optimal patch order problem in (2.13) is NP-hard, we propose a simple "hard-to-easy" heuristic to determine a good patch order. The key idea is that, if all the difficult-to-fill patches are first filled, then the missing pixels in the remaining easy-to-fill patches can be self-discovered at the decoder exploiting self-similarity in images, such that no more AI is required. Further, bundling the difficult-to-fill patches in the beginning of AI coding means that there is stronger statistical correlation among chosen modes for these patches, resulting in coding gain when the chosen modes are compressed using arithmetic coding, as described in Section 2.3.

In order to determine the "hard-to-easy" order, for each iteration t of the inpainting algorithm we compute a metric for each candidate target patch Ψ_p centered at $p \in \delta\Omega_t$ using known pixels in Φ_t. The metric is the distortion between candidate patch Ψ_p and the best matching block Ψ_q in Φ_t chosen via template-matching, see Eq. (1.6), and computed using only the known pixels in Ψ_p, *i.e.*, $\Psi_p \cap \Phi_t$. The candidate patch Ψ_p with the largest metric value will be deemed the hardest and is selected as the next target patch for filling. The intuition here is that the candidate patch Ψ_p with no good matching patch Ψ_q in the known region Φ_t likely lacks the self-similiarity characteristics that are required for nonlocal template matching to recover missing pixels. Hence this patch is deemed difficult-to-fill. Note that using this method, there is no need to explicitly inform the decoder about the location of the next target patch center p, as it can perform the exact same operations as the encoder to deduce the same target patch location.

The patch selection process is first computed until there are no missing pixels left. Then, a binary search is performed to identify the best end point, at which the encoder stops all AI transmission and the decoder is left to fill the remaining holes on its own via Criminisi's algorithm in the default "easy-to-hard" order. At each candidate end point, the RD cost including both the AI-assisted patches

and the decoder's self-discovered patches is computed, and the candidate point with the smallest RD cost is selected as the optimal end point. In practice, an "end-of-transmission" flag is coded to signal to the decoder the end of AI information and the start of the classical Criminisi's algorithm to fill in the remaining holes.

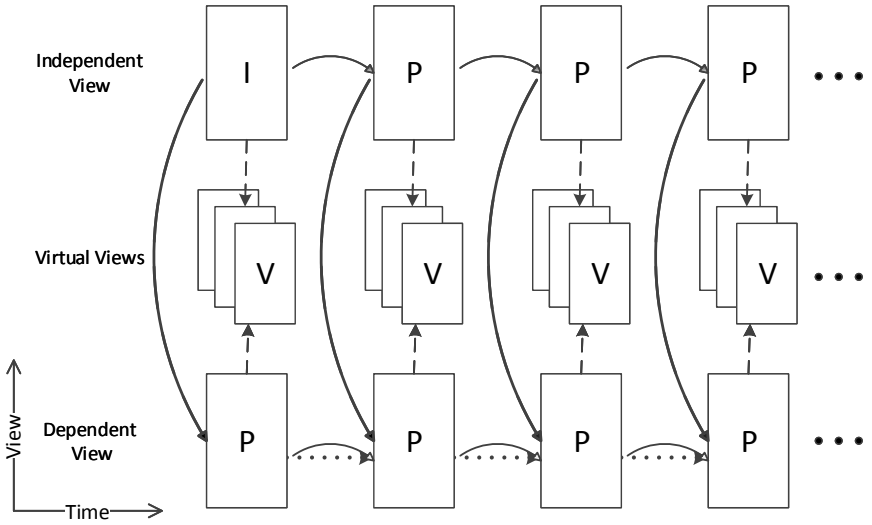
2.6 Experimentation

2.6.1 Experimental Setup

To test the coding performance of our proposed encoder-driven inpainting strategy, we performed the following experiments. We selected the following four JCT-3V standard test sequences [71]: `Undo_Dancer` and `GT_Fly` from Nokia at 1920×1080 pixels resolution, and `Kendo` and `Balloons` from Nagoya University at 1024×768 pixels resolution. `Undo_Dancer` and `GT_Fly` are synthetic multiview video sequences with very accurate depth values, while `Kendo` and `Balloons` are video sequences captured with multiple cameras where depth values have been estimated using stereo matching algorithms. The camera viewpoints of `GT_Fly` and `Undo_Dancer` move over time. In contrast, the camera positions of `Kendo` and `Balloons` are fixed. The foreground objects (the two swordsmen) in `Kendo` consume a larger spatial area and have larger movements than the content of the other three sequences. The first 200 frames of each sequence are used in our experiments.

As one of the advantages of texture-plus-depth representation resides in the possibility for users to synthesize viewpoints at decoder, we evaluate the performance of our algorithm not only on pure compression results for camera views, but also in terms of benefits for view synthesis. Hence, besides the camera-captured views of the testing sequences, we also synthesize several equally-spaced intermediate virtual views between the camera-captured views, and take the average distortion of camera-captured and virtual views as our performance measure. The distortion is measured by the average Peak-Signal-to-Noise-Ratio (PSNR) of the two coded views with and without virtual views, respectively. Note that the PSNR in our experiments is a combination of Y, U, V channels, *i.e.*, $\mathrm{PSNR} = (4 \times \mathrm{YPSNR} + \mathrm{UPSNR} + \mathrm{VPSNR})/6$, because all the testing YUV sequences are in 4:2:0 format. To conduct a fair comparison, similar prediction structures are used for our scheme and a competing scheme, 3D-HEVC [91] as shown in Fig. 2.5. The independent view of our scheme is coded using 3D-HEVC in IPPP structure, and the reconstructed texture and depth images of the independent view are used to synthesize texture and depth maps of the dependent view via DIBR.

Then, to evaluate the coding performance for a wide range of bitrates, we use four quantization parameters (QP) (25, 30, 35, and 40) for texture and depth images in both 3D-HEVC and our AI "intra" mode. Lossy coding of the independent view images has large influence on the quality of warped images in dependent view. In order to reconstruct quality-consistent video, QP of AI "intra" is equivalent to the one used to quantize the independent view.



(a) 3D-HEVC

(b) The proposed coding strategy

Figure 2.5: Prediction structures used in our experiments, where I and P denote I- and P-frames respectively, and V denotes equally-spaced intermediate virtual view images. In (b), W symbolizes the warped region and H the hole region. Solid lines in both figures denote block-based inter/inter-view prediction. Dashed lines denote view synthesis via DIBR. Dotted lines in (b) indicate the reference frame used by template matching of temporally nonlocal “skip” and the motion estimation of inter-frame AI “vec”.

The patch size in our proposed strategy is chosen to be 8×8. In theory we could also make our patch variable sizes in quadtree structures in the hope for better coding performance, but in this chapter we focus on the point of encoder-driven inpainting, which can be illustrated with fixed size patch without introducing too much complexity. Accordingly, the configuration of 3D-HEVC in our experiments generally follows the common test condition [71], except that we set the size of coding tree unit (CTU) to be 16×16, which is smallest option of CTU and the closest to the patch size that we choose. Since 16×16 CTU of 3D-HEVC includes 8×8 coding unit as one of its candidate modes, 16×16 CTU probably leads to better compression performance than the fixed 8×8 coding patch. However, we will show that with better management of redundancies our method using 8×8 coding patch nevertheless outperforms 3D-HEVC using 16×16 CTU.

2.6.2 Experimental Results

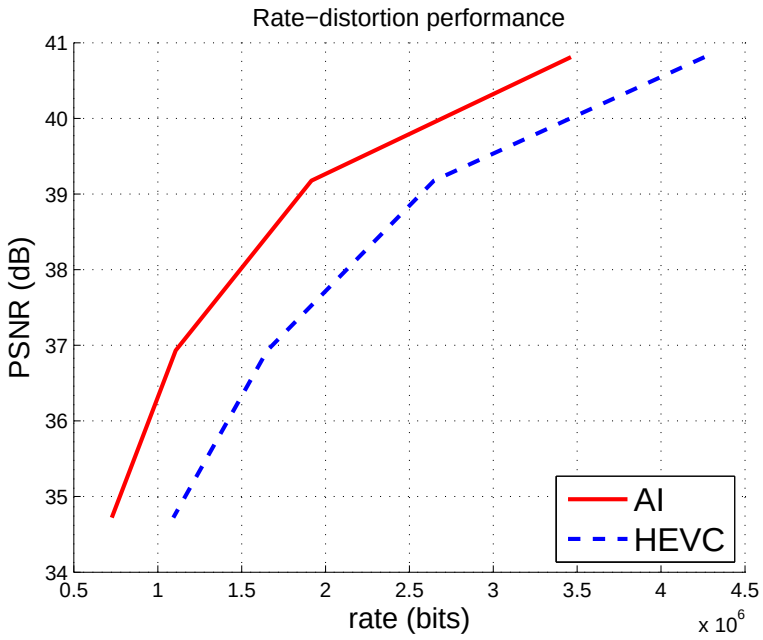
Comparison of Rate-Distortion Performance

We first compare the RD performance of our approach[6] with 3D-HEVC. Recall that the rates and distortions of the independent view are identical for the two competing schemes since both use the same coding scheme. The differences are spotted on the dependent view and the synthesized intermediate virtual views. The rates of the dependent view in our inpainting-based scheme are those used to complete the holes (H region in Fig 2.5(b)), while the warped pixels (W region in Fig 2.5(b)) actually cost zero bits. As mentioned in Section 2.2, the reconstruction of depth maps of the dependent view does not need any coding bits in our scheme. In this experiment, the total rate is the summation of the independent and dependent view including texture and depth images. The comparison of rate-distortion performance is shown in Fig. 2.6, 2.7, 2.8, and 2.9. We can find large gains of the proposed encoder-driven inpainting based coding over 3D-HEVC for different testing sequences, which mainly comes from the compact representation of 3D scene. The inter-view redundancy is exploited via DIBR: we do not code rendered regions whereas 3D-HEVC needs to code every block. The inter-patch redundancy is exploited via AI "skip" and "vec" while 3D-HEVC does not have efficient tools designed for nonlocal image-similarity. Further, the inter-pixel redundancy is exploited via image inpainting and novel transforms that are more suitable to our scenario than intra-prediction and DCT used in 3D-HEVC.

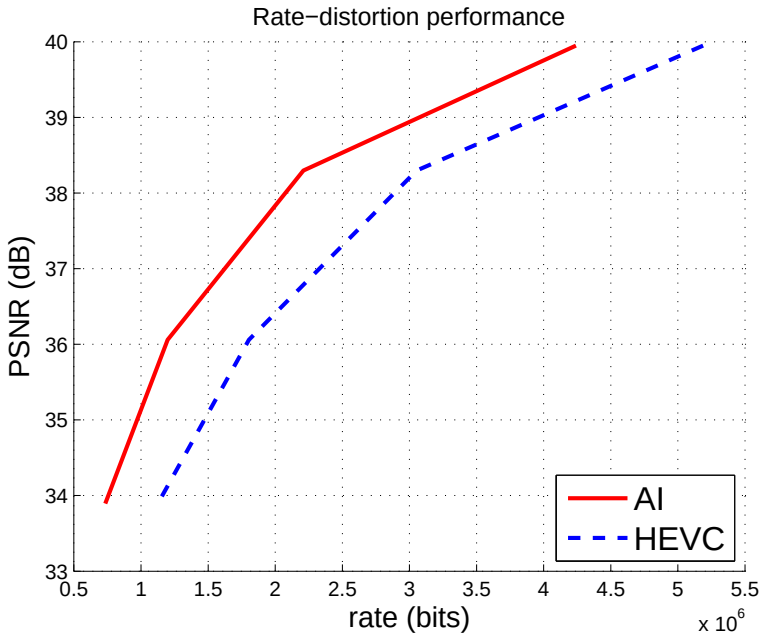
Comparison of Criminisi's "Easy-to-Hard" Order and the Proposed "Hard-to-Easy" Order

We next examine the effectiveness of our proposed "hard-to-easy" patch order. First, we show in Fig. 2.10 that the "hard-to-easy" order can be achieved by iteratively picking the patch that has the

[6] Here we use all the proposed coding tools including the new transforms and "hard-to-easy" patch order. Some components will be examined individually later.

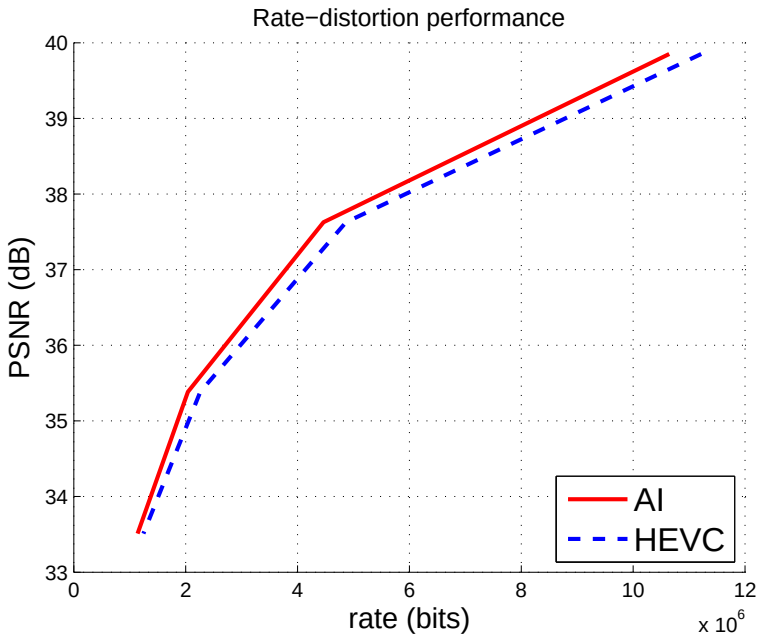


(a) Kendo, BD gain: 0.84 dB

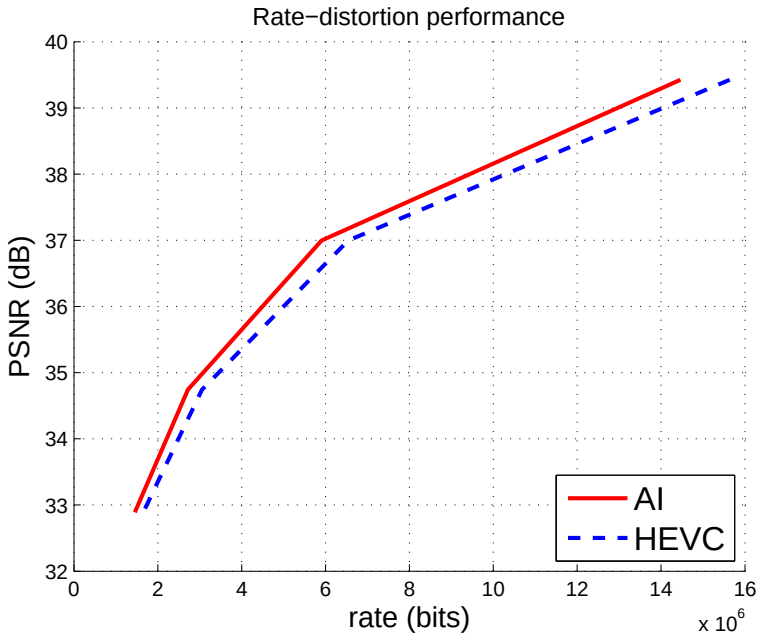


(b) Balloons, BD gain: 0.80 dB

Figure 2.6: Rate-distortion performance comparison: the total rate vs. the average distortion of two coded views. BD gain denotes the increment of Bjontegaard PSNR [7] of our scheme over 3D-HEVC.

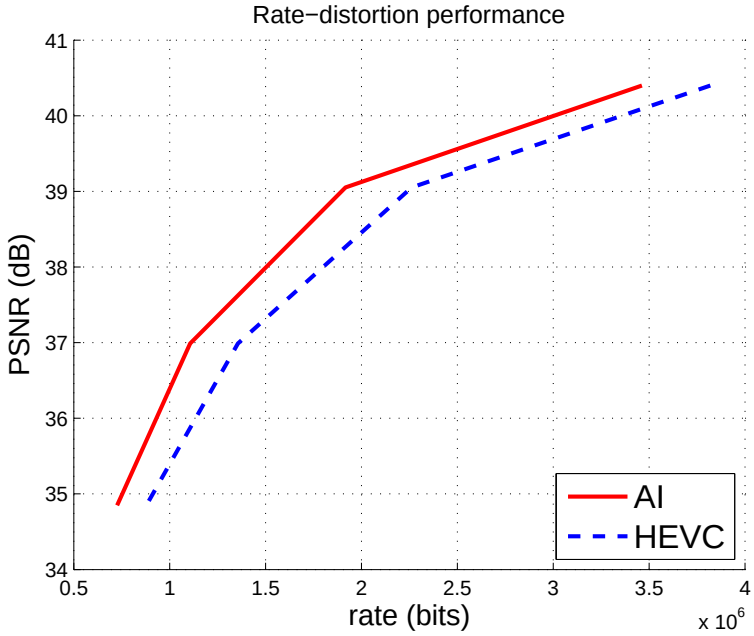


(a) GT_Fly, BD gain: 0.27 dB

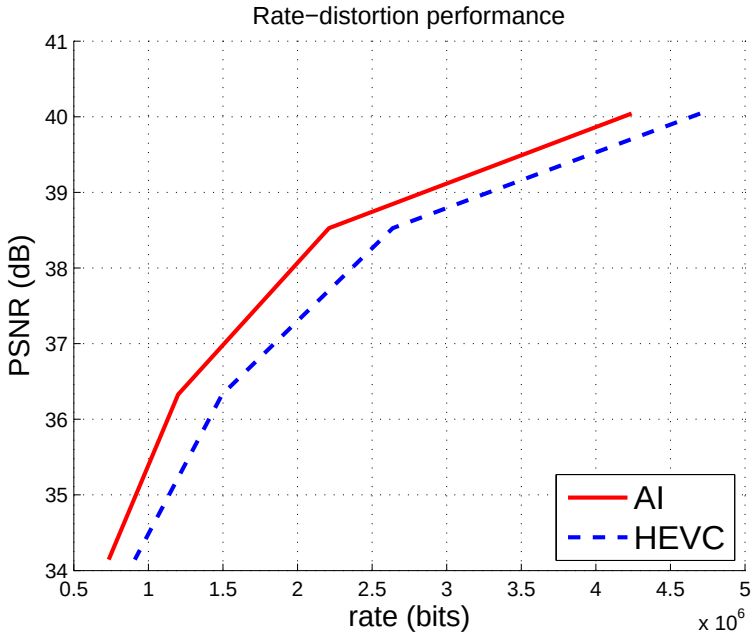


(b) Undo_Dancer, BD gain: 0.33 dB

Figure 2.7: Rate-distortion performance comparison: the total rate vs. the average distortion of two coded views. BD gain denotes the increment of Bjontegaard PSNR [7] of our scheme over 3D-HEVC.

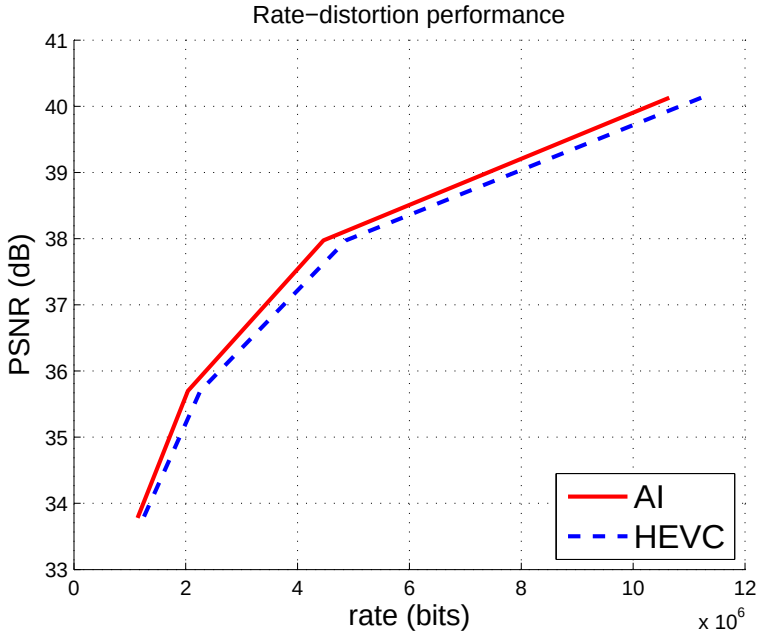


(a) Kendo, BD gain: 0.79 dB

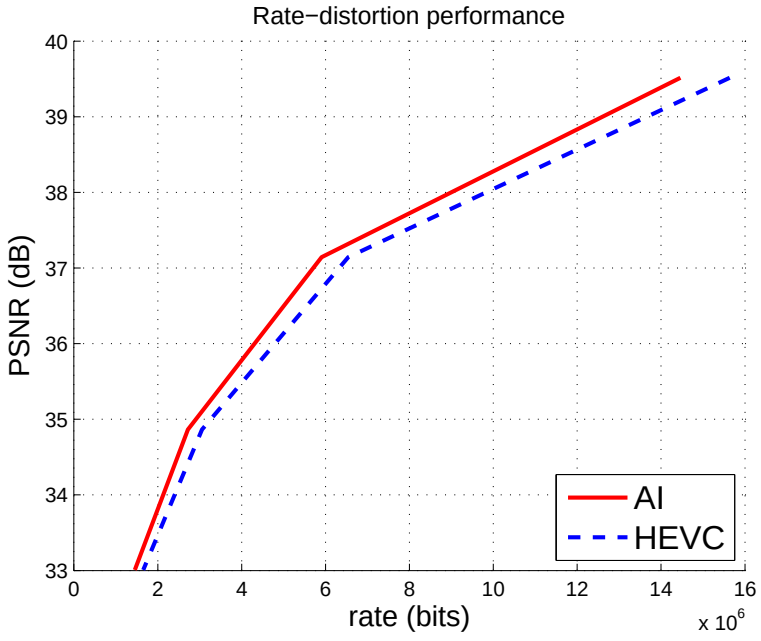


(b) Balloons, BD gain: 0.80 dB

Figure 2.8: Rate-distortion performance comparison: the total rate vs. the average distortion of two coded views and seven synthesized intermediate views. BD gain denotes the increment of Bjontegaard PSNR [7] of our scheme over 3D-HEVC.



(a) GT_Fly, BD gain: 0.27 dB



(b) Undo_Dancer, BD gain: 0.33 dB

Figure 2.9: Rate-distortion performance comparison: the total rate vs. the average distortion of two coded views and seven synthesized intermediate views. BD gain denotes the increment of Bjontegaard PSNR [7] of our scheme over 3D-HEVC.

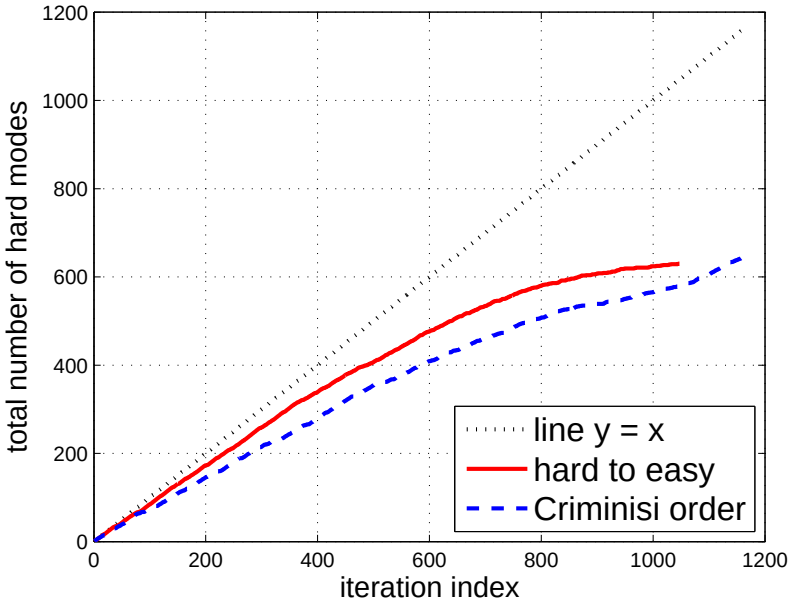
largest distortion with its best match in source region. The x-axis of Fig. 2.10 indicates the iteration index. We compare our patch order with Criminisi's "easy-to-hard" order, where in each iteration a template-matching fills a selected patch. The y-axis denotes the total number of hard modes selected before the current iteration. The hard modes include AI "vec" and "intra", because the corresponding patches cannot be properly reconstructed without any explicit instructions. These figures depict the accumulation process of hard modes over iterations, using our patch order and Criminisi's order. Note that we also draw a diagonal line $y = x$ in the figure as reference, which represents the situation where all modes up to current iteration are hard. Since we expect that a "hard-to-easy" order can first pick hard-to-fill patches, the early part of the curve should approximate $y = x$. At some point, when most of hard patches have been already coded, the curve should grow as slow as possible with only a few hard modes left. In this sense, our heuristic for "hard-to-easy" order performs as expected. Recall that, using the proposed "hard-to-easy" patch order, we terminate the signaling of AI at an optimal RD point; the remaining holes can be inpainted by the decoder, so that the number of iterations in our strategy are often less than a strategy using in Criminisi's order as shown in Fig. 2.10.

The next experiment shows the performance improvement using our proposed "hard-to-easy" order. As shown in Fig. 2.11, our proposed "hard-to-easy" order outperforms Criminisi's order. Larger gains are observed at low rate, because more easy modes and less hard modes are chosen and the bitstream is likely to be truncated earlier. We also observe larger coding gains for the `Kendo` sequence than for `Undo_Dancer`, because the growth of the number of hard modes in the `Kendo` sequence gets slow earlier (see the saturation of the two red solid curves in Fig. 2.10), so there is more room for `Kendo` to perform truncation, resulting in more bits-saving.

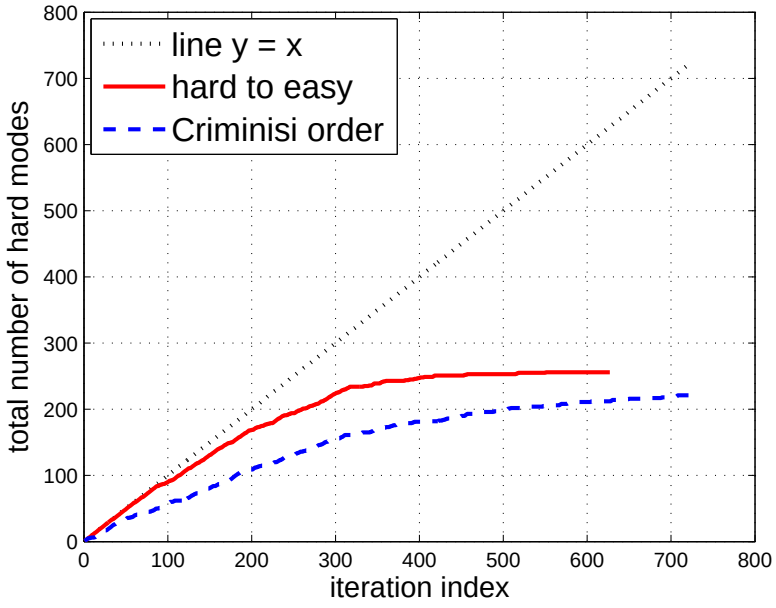
Finally, we note that the arithmetic coding of mode index can take advantage of the proposed "hard-to-easy" order: "hard" modes such as AI "intra" or "vec" are chosen more frequently at the beginning, while "easy" mode AI "skip" is likely chosen at the middle and end of the coding process. When we evaluate the probabilities of each mode over $W = 100$ last mode decisions, we observe a rate reduction of 15.51% (`Undo_Dancer`) and of 14.45% (`Kendo`) for the coding of mode indices, compared to the arithmetic coding without a limited-length window whose probabilities are estimated by all previous mode decisions.

Evaluation of Graph Fourier Transform and Sparsification Procedure Using DCT

One key difference of our coding strategy with 3D-HEVC lies in the transform coding step. Instead of regular DCT, we propose GFT and a sparsification procedure using DCT (*i.e.*, transform domain sparsification (TDS)) to take advantage of the particular feature of our coding patch, namely parts of patches are known and do not need to be coded. The resulting gain using the proposed transforms can be found in Fig. 2.12, where we compare the performance of our strategy using 1) DCT; 2) TDS and DCT; 3) GFT and DCT; 4) all transforms (DCT, TDS, and GFT). For case 2) to 4), we pick the

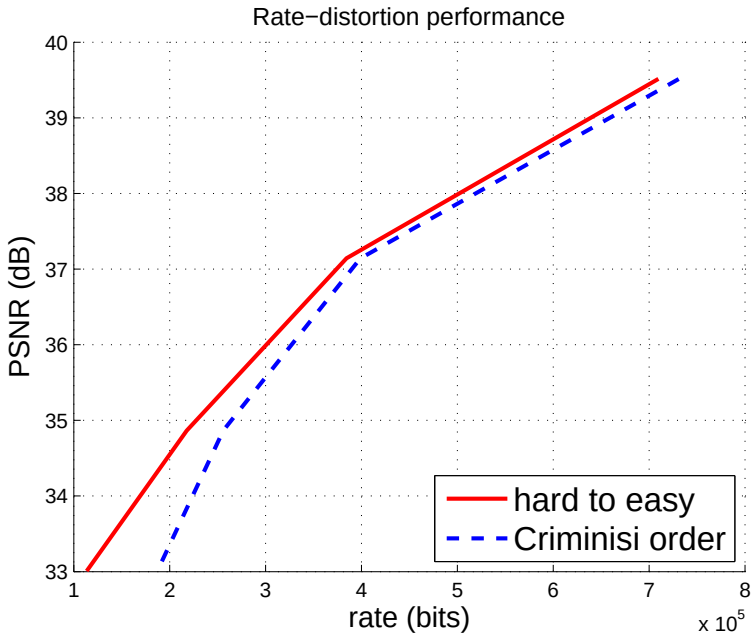


(a) Undo_Dancer

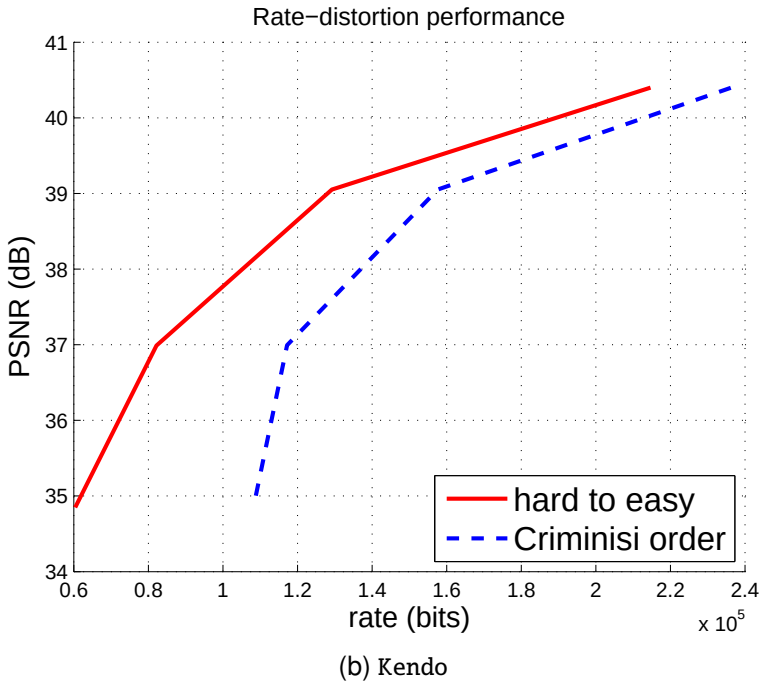


(b) Kendo

Figure 2.10: Iterative accumulation of hard modes along the iteration of the coding scheme, for two different patch orders.



(a) Undo_Dancer



(b) Kendo

Figure 2.11: Rate-distortion performance of our inpainting-based coding strategy using two different patch orders.

best transform in RD sense for each patch.

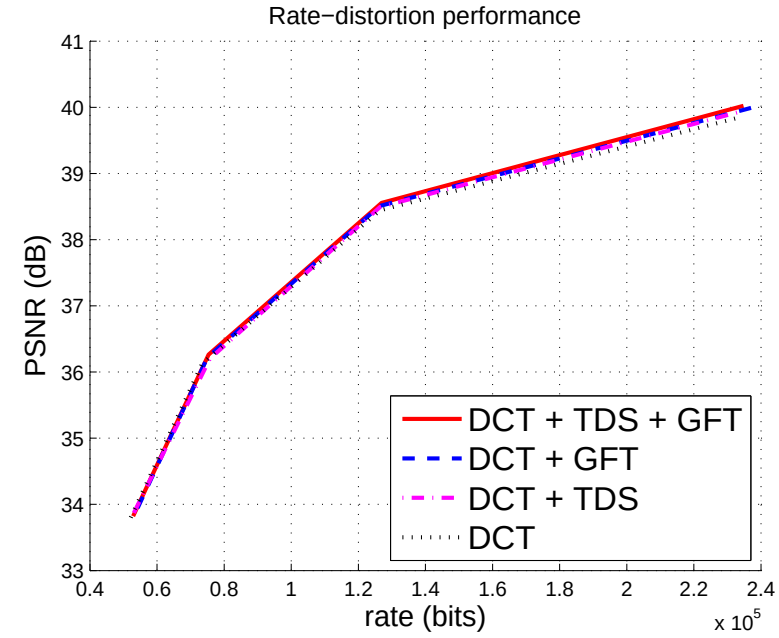
Since the proposed transforms are only for AI "intra", coding gain is observed mostly at high rate, where more AI "intra" modes are selected to achieve better reconstruction quality. As shown in next part, our coding strategy prefers low-cost AI "skip", and AI "vec" at low rate. Thus when the proportion of selected AI "intra" is already small, the performance improvement using our proposed transforms is limited. Fig. 2.12 shows that, using GFT can get more gain over the sparsification procedure, but the latter only needs to iteratively solve least squares at encoder, while at decoder the regular inverse DCT is only performed once, thus is less expensive in computation compared to GFT. The performance over high rate is emphasized in Fig. 2.12 (b), where the combination of TDS and GFT is better than using them individually.

Statistics of AI Modes

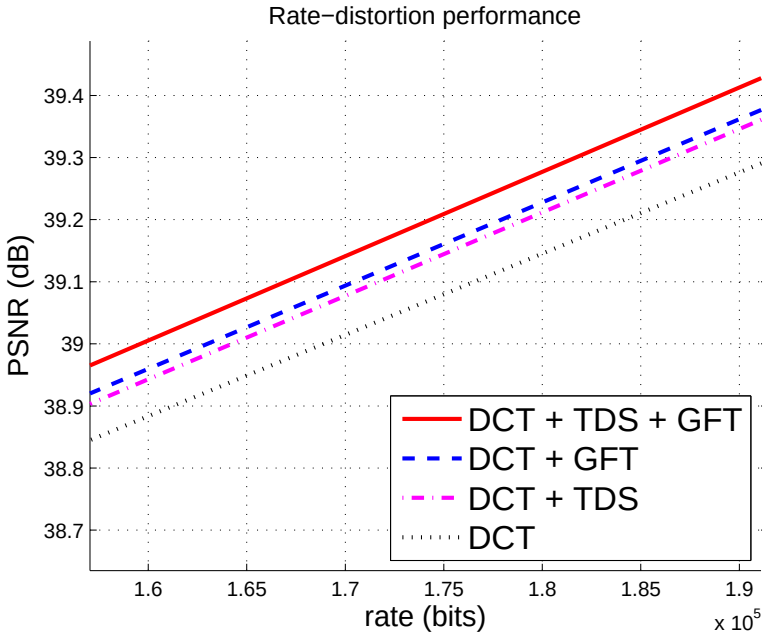
Each AI mode is proposed with its own purpose. To better understand why they are introduced and how they work, we show the statistics of AI modes in different circumstances. As shown in Fig. 2.13 and 2.14, we can find that 1) AI "skip" and "vec" dominate, meaning that there is a great amount of similarity to exploit from the known region and reference frame; 2) with larger QP, *i.e.*, lower rate budget, the number of the cheapest "skip" mode increases while the expensive "intra" and "vec" mode decreases; 3) the difference between I-frame and P-frame in the coding structure of Fig. 2.5 is not obvious, but obviously the chroma component requires less "intra" mode due to simpler texture; 4) the proposed coding strategy can adapt to the feature of different sequences by choosing different sets of AI modes.

2.7 Summary

Compression of texture and depth maps from multiple closely-spaced camera viewpoints is important for 3D imaging applications and new free viewpoint video communication. In this chapter, we propose an encoder-driven inpainting strategy to complete disocclusion holes in the DIBR-synthesized image in an RD optimal manner. Missing pixel regions that are difficult-to-inpaint are first completed following instructions from the encoder in the form of auxiliary information (AI). The remaining easy-to-fill holes are then completed without encoder's help via nonlocal template matching, which is effective due to the self-similarity characteristics in natural images. Finally, we propose two patch-based transform coding techniques (graph Fourier transform and DCT sparsification), so that only missing pixels in a target patch are encoded, avoiding representation redundancy. In doing so, our coding strategy successfully exploits the three kinds of redundancy inherent in the texture-plus-depth representation for coding gain: i) inter-view redundancy via DIBR-based 3D warping; ii)

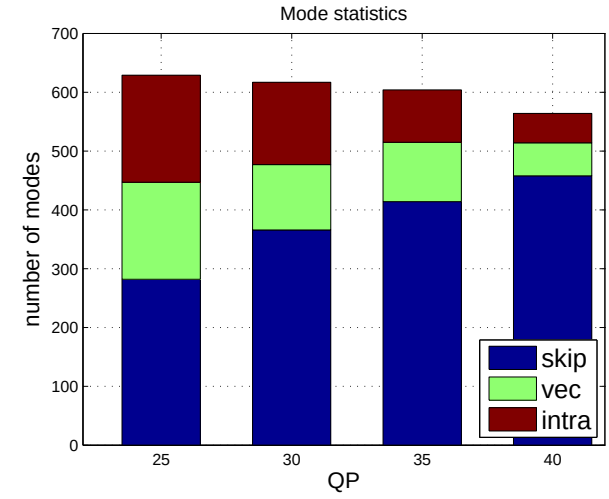


(a) `Balloons`

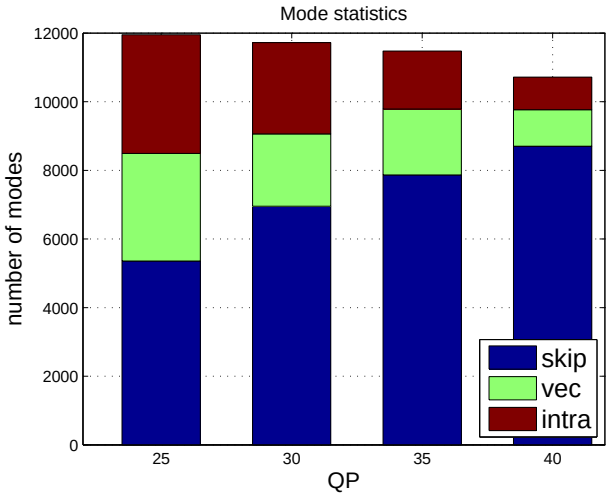


(b) `Balloons`, zoom in on high rate

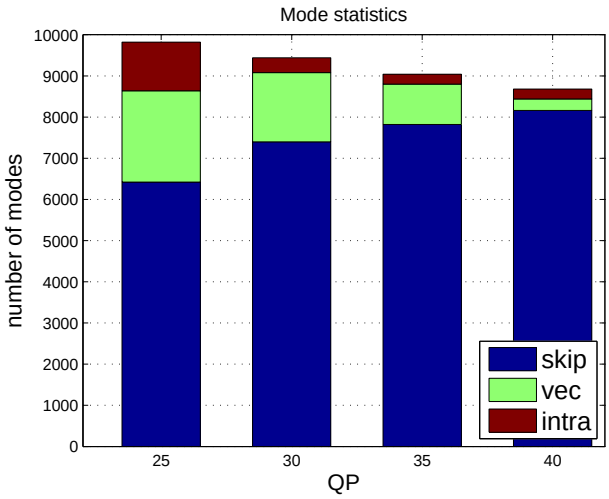
Figure 2.12: Performance of the different transforms to code unknown pixels in our coding scheme.



(a) I-frame

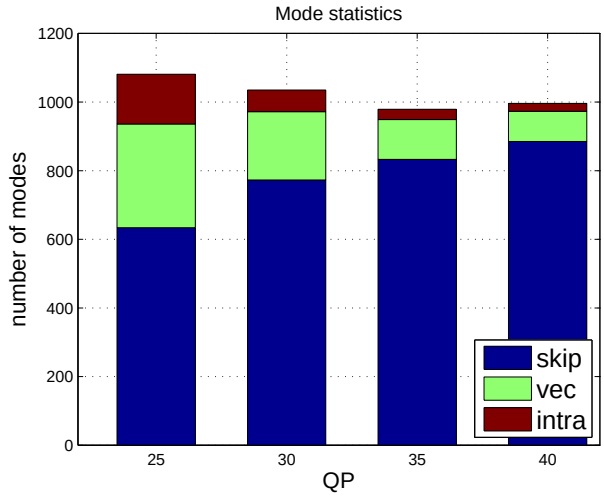


(b) P-frames

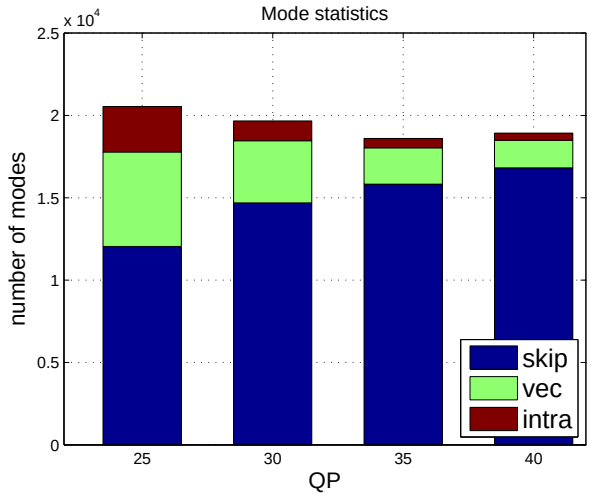


(c) Chroma component

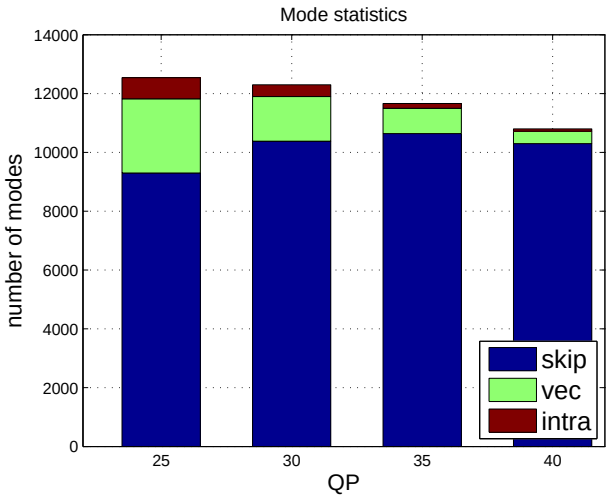
Figure 2.13: Mode statistics for coding of Kendo with the proposed scheme.



(a) I-frame



(b) P-frames



(c) Chroma component

Figure 2.14: Mode statistics for coding of `GT_Fly` with the proposed scheme.

inter-pixel redundancy via patch-based transform coding; and iii) inter-patch redundancy via nonlocal template matching.

The experimental results show that the proposed encoder-driven inpainting strategy is more effective than conventional multiview video coding in RD sense. The inter-view, inter-patch, and inter-pixel redundancies are greatly reduced by the efficient hole filling using the well-designed AI modes. The proposed novel transforms boosts RD performance at high rate and the “hard-to-easy” patch order improves RD performance mainly at low rate such that our overall scheme can get noticeable coding gain over a large range of rate. Since we use DIBR to construct a starting image to inpaint, the spacing of coding views and the quality of depth maps limit the performance of our approach. Although the proposed method outperforms conventional multiview video coding like 3D-HEVC, we expect that, in addition to regular block-based decoding, a contribution from the decoder to be able to handle the new introduced coding modes.

Chapter 3

Fast Estimation of the JND Maps for Perceptual MVC

In this chapter, we consider perceptual MVC, where the just noticeable distortion (JND) is used to cheat human visual system so that lower rate can be used in video coding. The direct calculation of the JND map incurs high complexity, and the problem is aggravated in MVC. We propose two fast methods to generate the JND maps of multiview videos. In the first method, the JND maps of some reference views are used to synthesize the JND maps of other views via the depth image based rendering (DIBR), which can be much faster than the direct JND computation. In the second method, the motion and disparity vectors obtained during the video coding are employed to predict the JND maps. If the prediction is not satisfactory, the JND block will be refreshed by calculating the JND directly. This method does not need any camera parameters and depth maps. The performances of the two fast JND map generation methods are evaluated in a perceptual MVC framework, where the residuals after spatial, temporal, or inter-view prediction are tuned according to the JND thresholds to save the bits without affecting the perceptual quality. Experimental results show that the JND prediction method has better accuracy and lower complexity. In addition, both fast JND methods lead to negligible degradation of the coding performance, compared to the direct JND computation.

3.1 Introduction

The main goal of video coding is to achieve the best possible reconstruction quality for a given bit rate. Since human eyes are the ultimate receivers of video signals, it is reasonable that the video quality should be judged by human opinions. However, in the block-based video coding schemes, it is impractical to bother people to assess the quality of image blocks and help to choose a prediction

mode. Therefore, objective measurements such as mean squared error (MSE) are widely adopted. Nevertheless, simple measurements like MSE bear some shortcomings, one of which is that they could not satisfactorily match the human's perception. Therefore, the properties of the human visual system (HVS) have been extensively studied, with the hope to find useful clues to overcome the disadvantages of current coding schemes.

One achievement of the research on HVS is the JND model [53], which shows that any error around a pixel below a JND threshold cannot be perceived by the human being. Therefore if we hide some errors according to the JND model, the perceived coding performance could be improved. As a result, in addition to the temporal, spatial and statistical redundancies, the perceptual redundancy can also be removed in the JND-based perceptual video coding. The JND threshold can be determined in either pixel domain or subband domain. The pixel-domain JND thresholds are estimated by luminance adaptation and texture masking, whereas the subband-domain methods exploit the varying spatio-frequency sensitivity of the HVS to tune transform coefficients [54].

Recently, there have been significant progresses in the development of the next generation visual communication services such as 3DTV and free viewpoint video (FVV), and efficient MVC is critical to their successful deployments [30]. In this chapter, we consider the applications of multiview videos where at any time only a single view of the multiview video dataset is displayed by a traditional 2-D monitor, although it is possible that the users can switch from one view to another at certain switching points. In this case, the traditional single-view JND model can still be used to reduce the bit rate of the system, while maintaining its perceptual quality.

The pixel-domain JND model is adopted in this chapter. A naive way of achieving this goal is to directly generate JND maps [1] for every frame and then apply the perceptual video coding to each frame. However, obtaining a JND map directly is computationally expensive, and the problem becomes more aggravated in MVC.

Since the multiview videos discussed in this chapter are taken in the same scene, their inherent similarities can be utilized to reduce the complexity of generating the JND maps. Two fast JND map estimation methods are proposed in this chapter. The first method renders the pixel-domain JND map of target view from a JND map of its neighboring view. The view synthesis can be employed because the pixel-domain JND map can be viewed as a gray level image, which is assumed to follow the pinhole camera model in multiple view geometry [47]. Different from the first method, our JND prediction method not only uses the inter-view correlation but also exploits the temporal similarity among JND maps. Specifically, the JND values of a block are copied from the best matched block in either a temporal reference frame or a neighboring view. The motion or disparity vectors and indices of reference frame employed in the JND prediction are obtained directly from the motion or disparity estimation in the coding procedure, so the complexity is very low. As both intra-view and inter-view correlations are exploited, the JND prediction method is expected to outperform the JND

[1] A JND map is a collection of per-pixel JND thresholds for an image.

synthesis method. The proposed methods are evaluated in a perceptual MVC framework, where the prediction residuals are tuned according to the JND thresholds to save the bits without affecting the perceptual quality.

3.2 Block DIBR Based Fast JND Map Synthesis

Image-based rendering refers to the creation of new views without a 3-D scene model. This is essentially a multidimensional sampling problem, which synthesizes new views using the reference textures in adjacent views. Besides the reference image $I[c,t,x,y]$ from camera c at time t, we assume in this section that the camera calibration parameters, *i.e.*, the intrinsic matrix $A(c)$, rotation matrix $R(c)$, translation vector $T(c)$, and the depth map $D[c,t,x,y]$ are also known.

In this chapter we synthesize the JND map of the target view in order to avoid directly calculating the map. A JND map generated by pixel-domain method can be considered as a gray level image. It can be much more efficiently warped from the available JND maps of neighboring views than direct computation. Experimental results in Sec. 3.5 show that the fidelity loss caused by the synthesized JND map can be tolerated in multiview video coding.

According to the pinhole camera model, the JND threshold at location (x,y) in camera c can be projected into the 3-D world coordinate $[u,v,w]$ by [65]

$$[u,v,w] = R(c)\cdot A^{-1}(c)\cdot[x,y,1]\cdot D[c,t,x,y] + T(c). \tag{3.1}$$

In order to get the corresponding JND threshold around $[x',y',z']$ in camera c', the world coordinate $[u,v,w]$ is then back-projected into the image plane of target camera c' by

$$[x',y',z'] = A(c')\cdot R^{-1}(c')\cdot[u,v,w] - T(c'). \tag{3.2}$$

The corresponding inhomogeneous coordinate is $[x'',y''] = [x'/z', y'/z']$.

To further reduce the complexity, we only synthesize a JND map block by block. That is, all pixels in the same block have the same 3-D projection. Obviously, this involves a tradeoff between the fidelity of the synthesized JND map and the speed of DIBR, which will be studied in Sec. 3.5.

Note that although some potential applications of this chapter like FVV may require depth information for view synthesis, the depth maps used in this chapter are not compressed or transmitted, as the decoder does not need any side information (depth or JND) to reconstruct from the coded bitstream. The proposed method does not change any syntax or decoding process of H.264 MVC. Thus, the compression of depth map is out of the scope of this chapter. Also, the depth maps are assumed to be available at the encoder side and the computational complexity of depth estimation is not considered.

There are some limitations of the DIBR based JND synthesis. First, to get good performance, the DIBR requires accurate calibration of cameras, high-quality depth maps, and small baseline to

reduce occlusion areas. Second, even if the requirements in the first point are satisfied, it is not guaranteed that adjacent pixels in the current view are adjacent in another view, because this depends on the camera positions and orientations and on the characteristics of the scene. Therefore, the JND value which relies on its neighboring pixels will be influenced and an offset away from the true value will be inevitable.

3.3 Block Matching Based Fast JND Map Prediction

In this section, we propose another method, the block matching based fast JND map prediction, which can be viewed as a generalization of DIBR based JND map synthesis discussed in Sec. 3.2. In addition to the inter-view similarity we have exploited in DIBR based method, the block matching based method takes advantage of the temporal correlation among frames taken by a certain camera. A JND block copying scheme will be introduced in Sec. 3.3.1. In Sec. 3.3.2 we analyze the error propagation in this scheme. In Sec. 3.3.3, the error propagation is resolved by a JND map refresh approach.

3.3.1 JND Block Copying Scheme

The block DIBR based JND map synthesis copies the whole JND block without any change from the reference JND map of a neighboring view. The method is essentially a kind of disparity compensation: the disparity vector is found by geometric projection instead of block matching, and there is no residual to compensate, as the original JND map of the current view is not available (if it were, we would not have bothered with the synthesis). Observed that the inter-view similarity has been exploited in the JND map synthesis, it is natural to study how to use the temporal correlation which is generally stronger than that between views [68].

In H.264 JMVC [1], the optimal motion vector and disparity vector of a block are estimated before the transform and entropy coding, which are used to find an inter-frame or inter-view block matching. In this section, we propose to take advantage of those readily available vectors to find the estimate of the current JND block from the JND maps in reference lists. The reference JND maps can be either from the current or neighboring views. It is reasonably assumed that the similar process of inter-frame or inter-view matching can be also performed for each JND map because the JND is derived from an image. Since we do not directly compute the original JND block, the lack of the ground truth leads to the compromise that the prediction errors of the current JND block cannot be compensated. Therefore, unlike the motion or disparity compensation of color image, only direct copying is performed for JND blocks. We name it block matching based JND copying scheme, whose complexity is very low.

In the context of this chapter, we attempt to improve the perceptual quality of reconstructed

video and in the meantime avoid increasing computational burden too much. In addition, JND map is basically a field of thresholds and will not be directly watched by observers. Therefore, only integer-pel motion vector or disparity vector is employed when copying a JND block. This can reduce the complexity by avoiding interpolating the JND maps, which does not contribute much to the visual quality.

Besides its low complexity, another advantage of this method is that depth maps and camera calibration parameters are not required any more. We simply use the available motion vectors or disparity vectors estimated for color image rather than the geometric projections to locate the matched JND blocks. The negative effect of quantized depth map and inaccurate camera calibration parameters in DIBR can thus be avoided.

3.3.2 Error Accumulation and Propagation of JND Block

Note that a block is not always coded by an inter mode even in a P or B slice. Sometimes, intra coding achieves better rate-distortion performance than inter coding, *e.g.*, when no good matching exists. In some smooth regions, even if good inter-frame matching can be found, intra mode is still likely to be chosen in the sense of rate-distortion cost. In this chapter, we propose to refresh (calculate directly as discussed in Sec. 1.2.3) the JND block when the matching of its associated color image block is so poor that large distortion is possible to be introduced if the JND block is copied from its predictor. Even if the matching is always acceptable, the errors caused by the JND copying scheme will be accumulated and propagated over time, which will be studied in the sequel.

Suppose frame are numbered by its prediction order from 0. Let $s_i(x,y)(i = 0,1,...)$ be a JND block in frame i at location (x,y) and $(d_{i,x},d_{i,y})$ be the ideally accurate motion vector (or disparity vector) pointing from JND block s_{i+1} to block s_i. If s_i is used to represent s_{i+1}, we have

$$s_{i+1}(x,y) = s_i(x-d_{i,x},y-d_{i,y}) + n_i(x-d_{i,x},y-d_{i,y}), \tag{3.3}$$

where n_i is the temporal or inter-view difference between the two correlated JND blocks. Since the accurate motion vector $(d_{i,x},d_{i,y})$ does not affect the error propagation of the JND copying scheme, to simplify the following mathematical analysis, we remove $(d_{i,x},d_{i,y})$ from the following equations, which is equivalent to a translation of the coordinate system of s_i. Therefore, Eq.(3.3) can be simplified as

$$s_{i+1}(x,y) = s_i(x,y) + n_i(x,y).$$

In this analysis, the JND map of frame 0 is refreshed entirely as the reference of the JND maps of frame $1,2,....$ Except frame 0, no JND block will be refreshed. It is convenient to observe the error accumulation and propagation in this setup. Hence, the estimate of s_1 can be viewed as an output of system h_{Δ_0}, given input s_0,

$$\hat{s}_1(x,y) = s_0(x,y) * h_{\Delta_0}(x,y). \tag{3.4}$$

where $*$ denotes convolution. $\hat{s}_1$ is basically a shift version of s_0 by Δ_0 which is an offset representing the inaccuracy of the integer-pel motion estimation employed in the JND copying scheme. The prediction error of s_1 can be written as

$$\begin{aligned} e_1(x,y) &= s_1(x,y) - \hat{s}_1(x,y) \\ &= s_0(x,y) + n_0(x,y) - s_0(x,y) * h_{\Delta_0}(x,y) \\ &= s_0(x,y) * [\delta(x,y) - h_{\Delta_0}(x,y)] + n_0(x,y). \end{aligned}$$

Similarly, since there is no compensation, the estimate of JND block s_2 can be found by

$$\begin{aligned} \hat{s}_2(x,y) &= \hat{s}_1(x,y) * h_{\Delta_1}(x,y) \\ &= s_0(x,y) * h_{\Delta_0}(x,y) * h_{\Delta_1}(x,y). \end{aligned}$$

And its accumulated prediction error is

$$\begin{aligned} e_2(x,y) &= s_2(x,y) - \hat{s}_2(x,y) \\ &= s_0(x,y) * [\delta(x,y) - h_{\Delta_0}(x,y) * h_{\Delta_1}(x,y)] + n_0(x,y) + n_1(x,y). \end{aligned}$$

For frame N, the accumulated error becomes

$$e_N(x,y) = s_0(x,y) * [\delta(x,y) - h_{\Delta_0}(x,y) * h_{\Delta_1}(x,y) * ... * h_{\Delta_{N-1}}(x,y)] + \sum_{i=0}^{N-1} n_i(x,y).$$

The convolution of the N space-shift systems can be simplified as

$$h_{\sum_{i=0}^{N-1}\Delta_i}(x,y) = h_{\Delta_0}(x,y) * h_{\Delta_1}(x,y) * ... * h_{\Delta_{N-1}}(x,y),$$

If we model $\{\Delta_i\}$ and $\{n_i\}$ $(i = 1,2,...,N)$ as independent zero mean Gaussian random variable sequences respectively, it can be observed that the variances of both $\sum_{i=0}^{N-1}\Delta_i$ and $\sum_{i=0}^{N-1} n_i(x,y)$ will become larger by increasing N, as the variance of the summation of independent Gaussian random variables is the summation of all the variances of those random variables. Therefore, the variance of the accumulated prediction error grows in the JND copying scheme.

3.3.3 JND Block Refresh Scheme

In the previous analysis, it can be seen that the prediction error of the JND copying scheme will be inflated along with time, so the JND block refresh will play a key role to control the propagation of the error. The remaining problem is whether or not a JND block should be refreshed. In this chapter, a simple but very effective method is proposed. If the sum of absolute difference (SAD) of a color picture block calculated in motion estimation stage is larger than a threshold T, that is, the inter prediction is not working well, the associated JND block will be refreshed.

The JND block refresh is basically a set of pixel filtering operations, whose computation costs do not strongly depend on the position of block, if the size of blocks is the same. Therefore we can assume there is approximately a linear relation between the times of refresh and the extra complexity introduced by JND. Let α be the proportion of refreshed blocks N_{rf} to the total number of blocks N_{tot}

$$\alpha = \frac{N_{rf}}{N_{tot}}.$$

Since the integer-pel motion (or disparity) vector and reference frame index for a macroblock are available in the process of inter-frame coding and little computation is entailed to the JND prediction obtaining and using the motion information, α is consequently an indicator of computational complexity of the JND prediction method. For example, when $\alpha = 0.5$ the JND prediction has about double computational complexity compared to the case with $\alpha = 0.25$.

Given a refresh proportion α (*i.e.* a certain computational complexity), we want to determine the threshold T. Let the SAD ϵ be a random variable different block by block following the probability density function $p_\epsilon(x)$. It is straightforward that α percentage of blocks with larger ϵ will be refreshed, so the α-percentile point ϵ_α can be found by

$$\mathbb{P}\{\epsilon > \epsilon_\alpha\} = \int_{\epsilon_\alpha}^{\infty} p_\epsilon(x)\,\mathrm{d}x = \alpha$$

As a result, we have $T = \epsilon_\alpha$. If we find a good probability density function to model the stochastic character of ϵ, the percentage α and the complexity could be precisely controlled, but this topic is out of the scope of the chapter. The refresh thresholds used in experiments are obtained empirically by the histogram of a collection of SAD's from a subsequence of each testing video.

Note that the terminologies JND block copying and JND block prediction are distinguished in this chapter. If the JND block refresh scheme is employed, *i.e.*, when $\alpha > 0$, we call it JND block prediction scheme. If there is no refresh in any block of a frame, we call it JND block copying scheme.

3.3.4 Steps of JND Synthesis and Prediction

The steps of JND synthesis method are:

1. Generate the JND map for view c' by using the direct method discussed in Section 1.2.3;

2. Project the top-left corner (x, y) of a 16×16 block in view c to view c' and obtain the corresponding pixel position (x', y');

3. Copy the JND map from the JND block whose top-left corner is at (x', y') in view c' to the JND block buffer at (x, y) in view c;

4. Perform step 2 and 3 for every block in view c until a synthesized JND map for view c is obtained.

The steps of JND prediction method are:

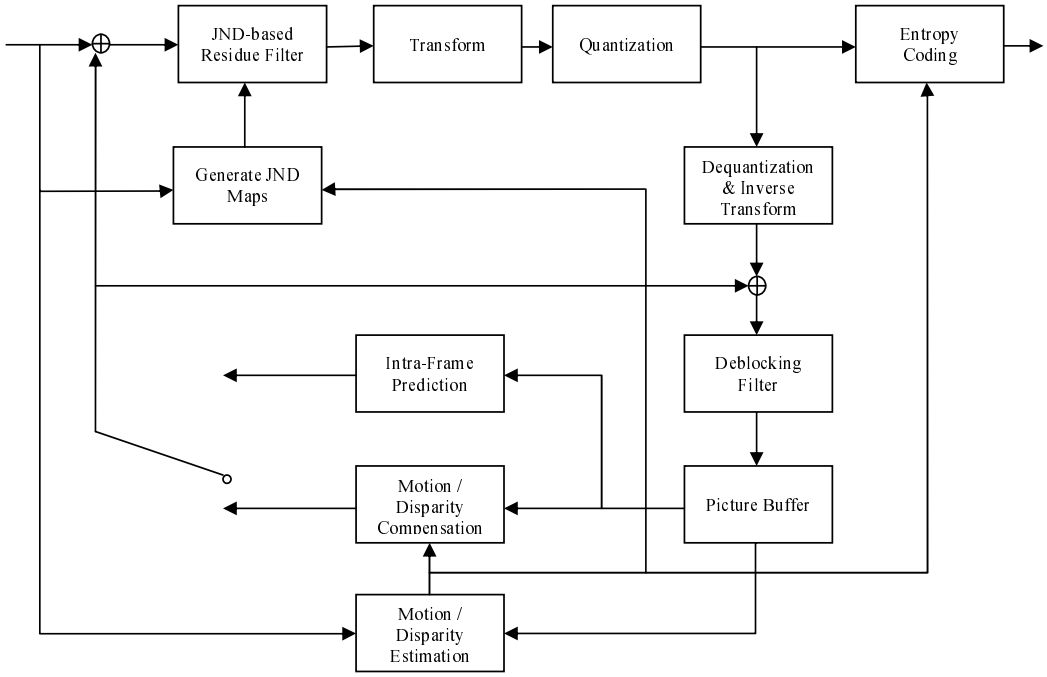


Figure 3.1: Diagram of the proposed perceptual MVC encoder.

1. Generate the JND map for I-frame by using the direct method discussed in Section 1.2.3;

2. For P-frame or B-frame, in the mode decision stage, obtain the integer-pel motion (or disparity) vector and reference frame index from 16×16 skip/inter mode, and record the SAD caused by the integer-pel motion estimation;

3. If the SAD is greater than a threshold, the JND for current block is refreshed;

4. If the SAD is not greater than the threshold, use the motion (or disparity) vector and reference frame index obtained in step 2 to find the best matched JND block, and copy the JND block to the current JND block buffer without compensation;

5. Perform step 2, 3, and 4 for every block until a predicted JND map for current frame is obtained.

3.4 JND-based Perceptual Multiview Video Coding Scheme

In this section, a perceptual MVC framework is developed, which will be used in the next section to compare the performances of the two fast JND map generation methods with the direct JND method. Note that the proposed fast JND synthesis and prediction methods can be applied to any JND based perceptual MVC, not just the scheme described in this section. Fig. 3.1 shows the block diagram of the encoder of the perceptual MVC. It can be seen that most parts of the encoder are

the same as the H.264 JMVC framework except for the JND-based residual filter, whose purpose is to tune residual toward the mean value of the 16 entries in a 4×4 block, subject to the constraint of the JND threshold at each pixel. The definition of the filter is modified from that in [110]. Let the average of the residuals in a 4×4 block be $\bar{e}$. When $\bar{e} \geq 0$, the operation of the filter is

$$e(x,y) = \begin{cases} \bar{e}, & \bar{e} < e(x,y) \leq \mathrm{JND}_s(x,y) \\ 0, & -\mathrm{JND}_s(x,y) \leq e(x,y) < 0 \end{cases} . \tag{3.5}$$

When $\bar{e} < 0$, the operation is

$$e(x,y) = \begin{cases} \bar{e}, & -\mathrm{JND}_s(x,y) \leq e(x,y) < \bar{e} \\ 0, & 0 < e(x,y) \leq \mathrm{JND}_s(x,y) \end{cases} , \tag{3.6}$$

where $e(x,y)$ is the prediction residual at pixel (x,y). One difference between (3.5) - (3.6) and that in [110] is that all residuals are tuned in [110], whereas our method only tunes the residuals whose absolute value are below the corresponding JND thresholds. Another difference is that the method in [110] is more complex than the one proposed in this chapter. The weight λ of the JND threshold in [110] has to be carefully tuned to achieve a good performance. In contrast, our method does not have any parameters to decide. This simplified perceptual MVC encoder is good enough to evaluate the performance of our fast JND generation methods.

After tuning, the variance of the residuals is smaller, which leads to the reduction of the signal's entropy; hence statistically the number of bits used to code the residuals will be reduced. Since the filter works under the constraint of the corresponding JND model, human eyes are not able to perceive the differences after the residuals are adjusted. Consequently the perceptual quality of the reconstructed video is not degraded. This conclusion can be better understood from the definition of PSPNR, which is a widely used perceptual distortion metric when the pixel-domain JND map is available. The PSPNR is given in [19],

$$\mathrm{PSPNR} = 10 \log_{10} \frac{255 \times 255}{\frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} (\mathrm{err}(x,y))^2 \delta(x,y)}$$

where

$$\mathrm{err}(x,y) = |I(x,y) - \hat{I}(x,y)| - \mathrm{JND}_s(x,y)$$

$$\delta(x,y) = \begin{cases} 1, & \text{if } |I(x,y) - \hat{I}(x,y)| \geq \mathrm{JND}_s(x,y) \\ 0, & \text{otherwise} \end{cases}$$

where $I(x,y)$ represents the uncompressed frame, $\hat{I}(x,y)$ denotes the reconstructed frame, W and H are the width and the height of the image, respectively. $\mathrm{JND}_s(x,y)$ is the JND threshold of $I(x,y)$ around pixel (x,y) and it is used as ground truth for all the PSPNR-based comparison in Sec. 3.5. It can be seen from the equations above that the tuning in (3.5) and (3.6) does not change the value of the PSPNR.

Table 3.1: Speedup factor and MSE of the synthesized JND, compared to the directly computed JND

Block Size	Speedup Factor	MSE
1×1	0.16	2.89
2×2	0.61	2.11
4×4	2.45	1.68
8×8	9.28	1.48
16×16	35.1	1.47
32×32	100.0	1.68

3.5 Experimental Results

In this section, we compare the performances of the proposed fast JND methods and the direct JND method in the perceptual MVC framework. Before that, the setup of DIBR based JND synthesis and the error propagation of the JND copying and prediction schemes will be studied.

3.5.1 Block Size of DIBR Based JND Synthesis

We first investigate the impact of block size on DIBR quality and complexity. We define speedup factor to be the time used by the direct JND method (see Sec. 1.2.3) over the time used by the synthesized JND method. Table 3.1 lists the speedup factors with different DIBR block sizes. In addition, the corresponding MSE per pixel for each case is reported, using the directly computed JND map as reference.

Surprisingly, the MSE performance of the block-based DIBR is not as good as expected when the block size is smaller than 16×16. This suggests that the impacts of the geometrical projection errors caused by the quantized depth maps and the imperfect calibration parameters might be less pronounced in large blocks. In terms of the complexity, the table shows that when the block size is larger than or equal to 4×4, the DIBR is computationally more efficient than the direct method. Taking both speed and accuracy into consideration, we choose the block size of JND synthesis to be 16×16 hereafter in this chapter.

3.5.2 Baseline Distance of DIBR Based JND Synthesis

To investigate the impact of the baseline distance between the target and the reference camera on the accuracy of the synthesized JND map, we use the JND maps of View 1 to View 7 of the sequence Breakdancers [113] to synthesize the JND map of View 0, respectively, where View 1 is the closest to View 0, and View 7 is the farthest. View 0 is then encoded using hierarchical B structure with GOP size of 4, and 97 frames are coded. The influence of the baseline distance to the R-D performance of perceptual coding View 0 is measured by PSPNR.

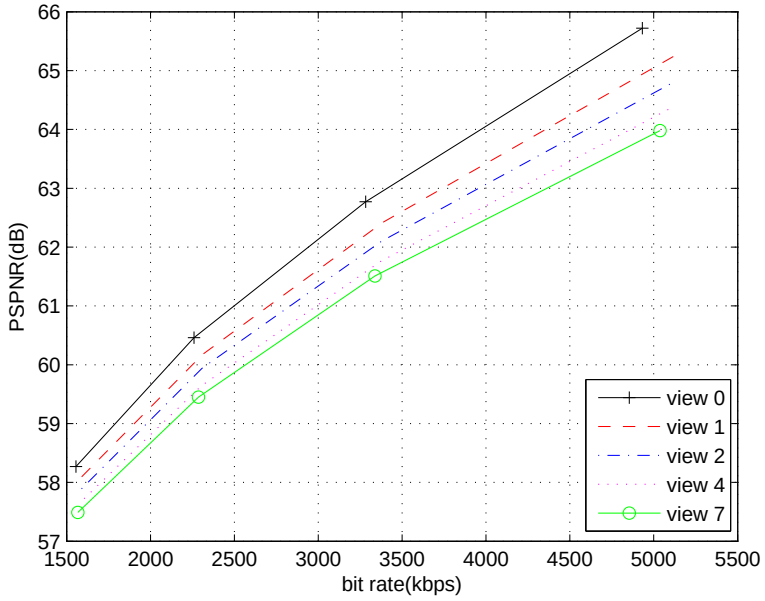


Figure 3.2: Coding performance loss of view 0 using synthesized JND maps with different baseline distances.

The results are summarized in Fig. 3.2, which suggests that the performance loss is less than 1dB when the JND map is synthesized from the immediate neighbor. In particular, the degradation is smaller at low rates.

3.5.3 Error Propagation and JND block Refresh

In Sec. 3.3.2, the error accumulation and propagation of the JND copying scheme are studied. To fight the propagation of the error, the JND block refresh is proposed, which will be empirically verified in this part. In Fig. 3.3, the curves with the notation $\alpha \to 0$ represent the case that the JND map of the first frame is refreshed and there is no refresh any more in the following frames, so when the number of frames is large enough the refresh percentage α goes to 0. In the synthesized JND method, no temporal prediction is involved, so the error will not be propagated over time. For the JND prediction method, two different error suppression intensities are tested, $\alpha = 0.25$ and $\alpha = 0.5$, respectively. The SNR in Fig. 3.3 is obtained by

$$\mathrm{SNR} = 10 \log_{10} \frac{\sigma^2_{\mathrm{JND}}}{\mathrm{MSE}}$$

where σ^2_{JND} is the variance of direct computed JND values (as ground truth), and MSE is the mean squared error between the ground truth and the synthesized or predicted JND values. From Fig. 3.3, it can be found that the propagation error is effectively under control when $\alpha = 0.25$. If the refresh

Table 3.2: Average SAD of JND blocks when the SAD of corresponding color blocks is greater or less than the thresholds

Sequence	QP	SAD $< \epsilon_{\frac{1}{2}}$	SAD $\geq \epsilon_{\frac{1}{2}}$	SAD $< \epsilon_{\frac{1}{4}}$	SAD $\geq \epsilon_{\frac{1}{4}}$
Breakdancers	18	53	131	64	178
Breakdancers	24	51	120	61	158
Ballet	16	15	49	23	61
Ballet	22	15	46	22	58

rate is set to $\alpha = 0.5$, the fidelity of predicted JND maps will be much better than that of synthesized JND maps, so we can expect a superior rate-distortion performance by using the predicted JND maps.

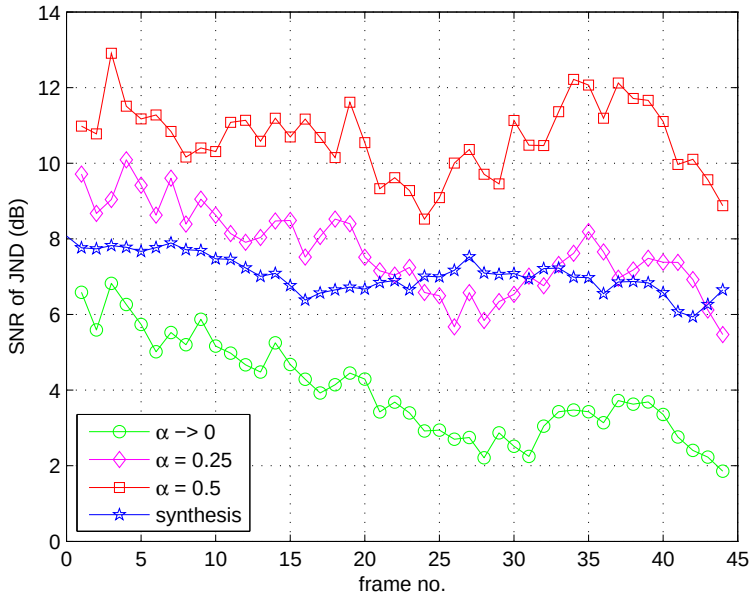
Note that the synthesized JND maps of sequence Ballet [113] are not so good as those of Breakdancers, as the occlusion areas of Ballet among different views are much larger than those of Breakdancers.

There is another way to justify the proposed JND block refresh scheme. As shown in Table 3.2, the SAD's of JND blocks are collected, when the SAD's of color blocks are greater than $T = \epsilon_\alpha$ and when they are smaller than the threshold. 1/2-percentile point $\epsilon_{\frac{1}{2}}$ and 1/4-percentile point $\epsilon_{\frac{1}{4}}$ are chosen as the thresholds in this experiment. For all the tests with various sequences and QP's, the conclusion is consistent. That is, when the prediction of color blocks are not satisfactory, *i.e.*, their SAD's are beyond the threshold, the average distortion of the corresponding JND blocks will become much larger, so it is reasonable to refresh them.

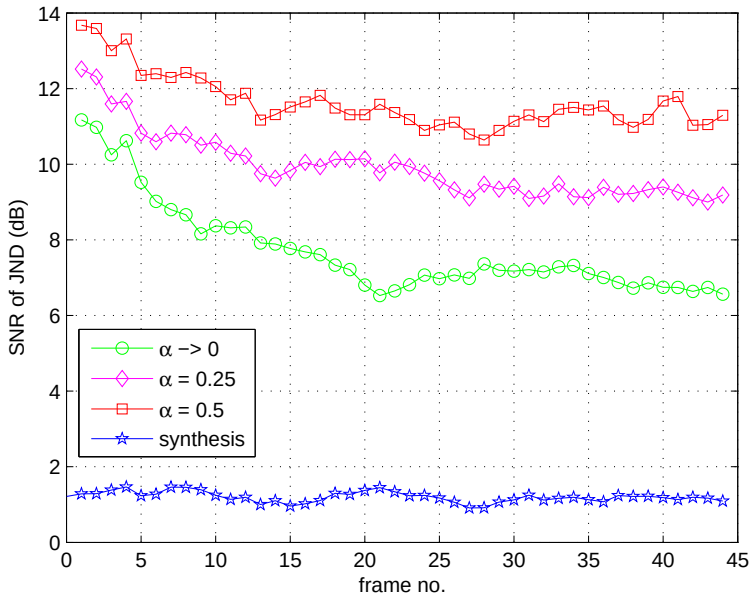
3.5.4 Performances of Fast JND Methods

We next compare the R-D performance of the perceptual MVC using the fast and direct JND methods. We encode the multiview sequences Breakdancers (1024×768), Ballet (1024×768), Champagne (1280×960), and Akko (640×480) [2] with hierarchical B structure and GOP size of 8. Twelve GOP's together with the first and the only I frame (97 frames in total) for each sequence are coded. For sequence ballet, the QP values tested are 16, 18, 20, and 22. For other sequences, the QP values tested are 18, 20, 22, and 24. In the synthesized JND method, the JND maps of Views 1 are warped from those of Views 0. For the JND prediction method, refresh ratios $\alpha \to 0$, $\alpha = 0.25$, and $\alpha = 0.5$ are chosen. Note that we can consider that 50% JND blocks are refreshed in the setup of the JND synthesis, because the JND of View 0 is entirely refreshed as inter-view reference and the JND of View 1 is entirely synthesized to reduce complexity. Hence, the JND synthesis and the JND prediction with $\alpha = 0.5$ have approximately the same computational cost, if the computation of depth estimation is not taken into account. The JND prediction with $\alpha = 0.25$ requires only a half of time to generate the JND maps compared to the JND synthesis method. Additionally, the perceptual MVC

[2] Champagne and Akko are provided by http://www.tanimoto.nuee.nagoya-u.ac.jp/~fukushima/mpegftv/



(a)



(b)

Figure 3.3: Error propagation of the JND prediction methods and the projection errors of the JND synthesis method (49 frames; QP=20; GOP size is 4; View 0 is independently coded; View 2 can be predicted by View 0; View 1 can be predicted by either View 0 or View 2). (a) View 1 of Breakdancers. (b) View 1 of Ballet.

with all JND maps refreshed can be viewed as a JND prediction method with $\alpha = 1$, *i.e.*, the direct JND method.

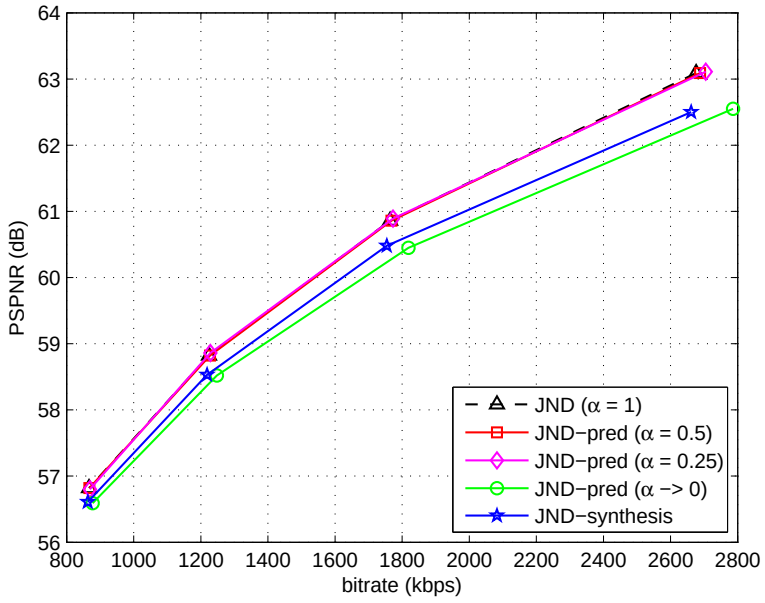
The R-D performances of View 1 of Breakdancers and Ballet are reported in Fig. 3.4. Since the JND synthesis method calculates the JND maps of View 0 directly and warps those of View 1 without any refresh, for a fair comparison, the average performances of View 0 and View 1 are illustrated in Fig. 3.5. The JND prediction scheme shows its superior performances in all experiments. At both $\alpha = 0.5$ and $\alpha = 0.25$, the rate-distortion curves of this method are very close to the curve of the all-refreshed method, but the complexity is only a half and a quarter of the direct JND method, respectively.

Moreover, the Bjontegaard Delta (BD) rate increment [7] of the fast JND-based MVC over the direct JND-based MVC is summarized in Table 3.3. Again, only negligible performance degradation is found in the JND prediction method. For $\alpha = 0.25$, the BD rate increment is no more than 0.6%; for $\alpha = 0.5$, the highest increment is only 0.26%. However, if we do not control the error propagation ($\alpha \to 0$), obvious performance loss for Breakdancers will be observed. The reason is that Breakdancers have dramatic and complex motions. This justifies the necessity of JND refresh. For the JND synthesis scheme, although the computational cost is reduced by a half, the bitrate increment is in general larger than that of the JND prediction method. The most obvious degradation is found in sequence Ballet, because the DIBR quality of Ballet is hurt by large occlusion areas.

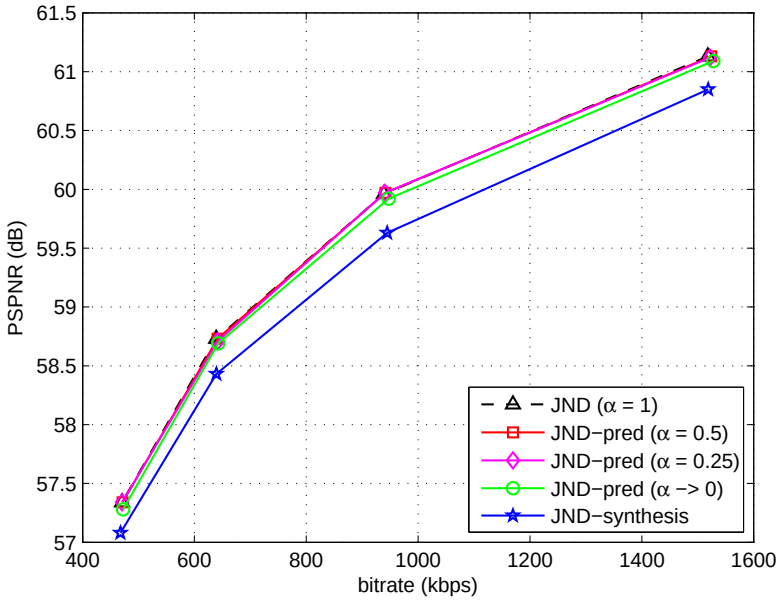
Note that there is no guarantee that the JND copied from the temporal reference is always better than the JND warped from the neighboring view, due to the following reasons. Firstly, as shown in Section 3.3.2, the block-based JND prediction method leads to error accumulation. If the refresh ratio α is low enough and there are many frames to code, the performance of block matching for JND copy could be worse than JND warping. Secondly, in video coding the temporal reference block is chosen by rate-distortion optimization. Thus the selected reference block is not necessarily the one with the minimal difference from the current block, as the motion vector also plays a role. Thirdly, if the multiview video system has very small baseline and very accurate depth information, the JND warping might be better than the JND prediction. Also, at low rate, the prediction is not very accurate; hence the JND prediction method based on motion vector or disparity vector could be affected. In contrast, the JND warping is not influenced by low rate at all.

3.6 Summary

In this chapter, two fast methods are proposed to generate the JND maps of multiview videos. In the JND synthesis method, for some views, the JND maps are obtained by exploiting the properties of luminance adaptation and texture masking of frames. The JND maps of other views are synthesized by utilizing existing neighboring JND maps. In the JND copying method, the motion and disparity information yielded in JMVC encoder is utilized to predict JND block from reference JND maps. The

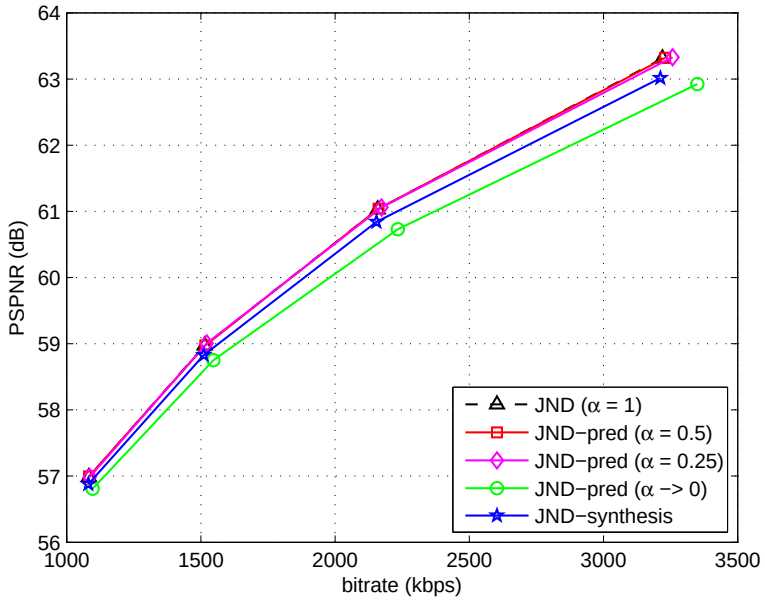


(a)

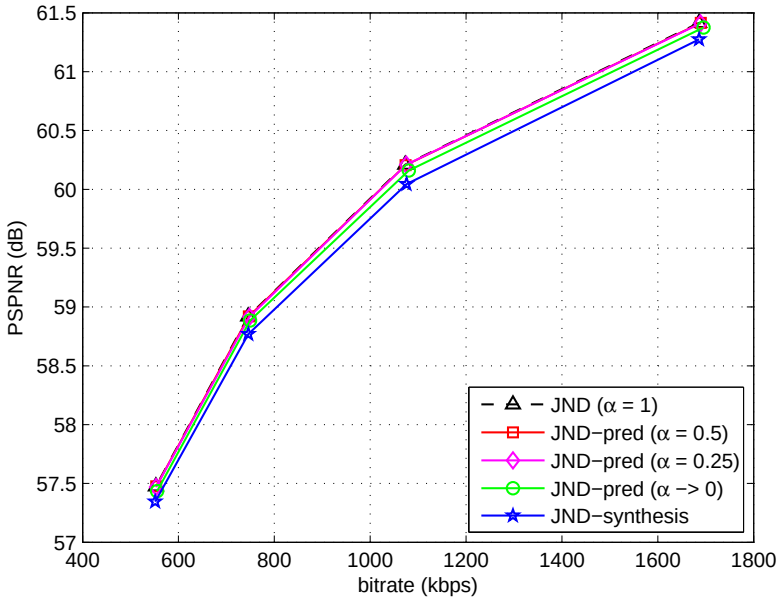


(b)

Figure 3.4: R-D performance comparison of the direct JND method and the fast JND methods. (a) View 1 of Breakdancers. (b) View 1 of Ballet.



(a)



(b)

Figure 3.5: R-D performance comparison of the direct JND method and the fast JND methods. (a) View 0 & View 1 of Breakdancers. (b) View 0 & View 1 of Ballet.

Table 3.3: BD rate increment of the proposed methods (V stands for View). N/A means there are no depth map and camera calibration parameters for the DIBR.

Sequence	synthesis	$\alpha \to 0$	$\alpha = 0.25$	$\alpha = 0.5$
Breakdancers (V0+V1)	3.0%	8.0%	0.19%	0.05%
Breakdancers (V1)	6.6%	10%	0.02%	0.24%
Ballet (V0+V1)	4.5%	1.8%	0.30%	0.19%
Ballet (V1)	10%	2.0%	0.47%	0.23%
Champagne (V0+V1)	1.6%	1.4%	0.04%	0.03%
Champagne (V1)	4.2%	1.4%	0.24%	0.02%
Akko (V0+V1)	N/A	2.8%	0.56%	0.12%
Akko (V1)	N/A	3.7%	0.60%	0.26%

error propagation of the method is studied and a simple JND block refresh approach is proposed to alleviate the influence of the error propagation.

A perceptual MVC scheme is developed based on the synthesized and predicted JND maps, where the residuals after intra or inter prediction are adjusted within the range of the corresponding JND thresholds. Experimental results show that the JND prediction method can be much faster than the JND synthesis method and has a comparable performance with the expensive direct JND method.

Chapter 4

MVC Based 3D Geometry Compression

In this chapter, we consider 3D geometry compression, which is important to graphical applications. The compression of dynamic 3D geometry obtained from depth sensors is challenging, because noise and temporal inconsistency inherent in acquisition of depth data means there is no one-to-one correspondence between sets of 3D points in consecutive time instants. Instead of coding 3D points (or meshes) directly, we propose to represent an object's 3D geometry as a collection of tile images. Specifically, we first place a set of image tiles around an object. Then, we project the object's 3D geometry onto the tiles that are interpreted as 2D depth images, which we subsequently encode using a modified MVC tuned for piecewise smooth signals. The crux of the tile image framework is the "optimal" placement of image tiles—one that yields the best tradeoff in rate and distortion. We show that if only planar and cylindrical tiles are considered, then the optimal placement problem can be mapped to a tractable piecewise linear approximation problem. We propose an efficient dynamic programming algorithm to find an optimal solution to the piecewise linear approximation problem. Experimental results show that optimal tiling outperforms naïve tiling by up to 35% in rate reduction, and graph Fourier transform (GFT) can further exploit the smoothness of the tile images for coding gain.

4.1 Introduction

The advent of depth sensing technologies like Microsoft Kinect means (partial) 3D geometries of objects in a dynamic scene can now be captured with relative ease. If an object's geometrical information can be accurately and compactly represented at the sender for network transmission, then

at the receiver it can enable a spectrum of 3D imaging applications, such as virtual image synthesis of the object from any freely chosen viewpoint. Thus compact representation of an object's 3D geometry is an important research problem.

Unlike computer-generated objects, dynamic 3D geometry periodically captured by depth sensors is subject to acquisition noise and temporal inconsistencies, which means there is no one-to-one correspondence in 3D points and edges between neighboring frames in time. Called time-varying meshes (TVM) in [75,108,109], the authors proposed to code TVM directly using predictive techniques similar to ones used in video compression algorithms like H.264 [101]. This is a difficult proposition, because: i) 3D meshes typically undergo more complicated transformations over time than the simple translational motion model assumed in motion prediction in video coding; and ii) lack of one-to-one correspondence in 3D points across frames means exact match at patch-level might not exist at all even if more complex motion models (which entail significant computation costs) are introduced.

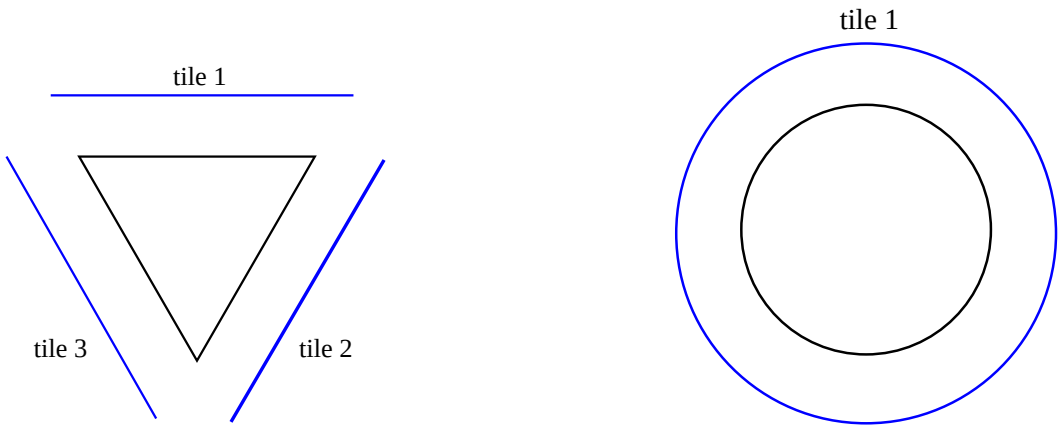
In this chapter, instead of coding 3D points (or meshes) directly, we propose an alternative to compactly represent the geometry of 3D objects using the concept of image tiling. Specifically, we first project the object's 3D geometry as images onto a set of carefully placed tiles surrounding the object, then encode the tile images using the MVC like [22] tuned for piecewise smooth signals (*e.g.*, incorporating tools like graph Fourier transform (GFT) [50, 82]). At the receiver, the decoded tile images are projected back into 3D space to reconstruct the object's geometry. The key to a compact yet accurate representation in our image tiling framework is the appropriate selection of image tiles for a given object. We show that by restricting the tile types considered to be planar and cylindrical tiles only, the optimal selection of tiles—the best-fitting tile combination given a representative cross section of the object—maps to a tractable piecewise linear approximation problem. We propose an efficient dynamic programming algorithm to solve the piecewise linear approximation problem. To the best of our knowledge, we are the first in the literature to address the optimal image tiling problem and provide a computation-efficient solution.

4.2 Image Tiles Selection

4.2.1 System Overview

We first describe the overall framework to represent an object using multiple depth images that are projections of the object's 3D geometry on selected tiles. The actual projected images on tiles can be coded using either DCT-based or GFT-based multiview image codec.

Given the geometry of an object in 3D mesh, we first place a set of K tiles surrounding the object, onto which the object's 3D geometry is projected as 2D images. By projection, we mean that for each pixel location on the tile, we trace an orthogonal line from the tile surface until we hit the



(a) linear tile (b) circular tile

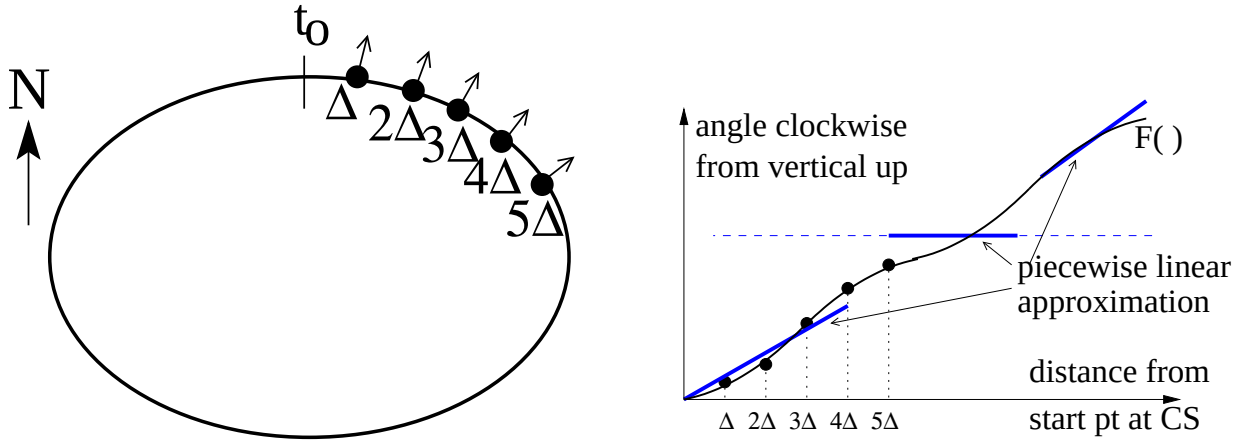
Figure 4.1: Examples of linear and circular tiles.

surface of the object, upon which we record the distance (depth) as the pixel value. Each resulting depth image on a tile is a 2D impression of the 3D geometry; the goal is to identify the optimal set of 2D impressions for a given object.

Intuitively, the best placements of tiles—in terms of accuracy in representing the object's geometry—are the ones that are parallel to the surface of the object. As simple examples, consider only the *cross sections* of two 3D objects in Fig. 4.1, which are triangle and circle in (a) and (b), respectively. For Fig. 4.1(a), if we place three tiles parallel to the three sides of the triangle, regularly sampled voxels on each side of the object can potentially be captured by the same number of evenly spaced pixels on the corresponding tile. (Evenly spaced pixels in a 2D grid on the tile becomes an image for coding.) If the three tile pixel lines are then losslessly coded, then the exact same triangle cross section (in terms of the original voxel samples) can be reproduced at the decoder, resulting in zero distortion. Similarly, in Fig. 4.1(b) we see a circular tile that is ideal for an object's circular cross section. Given this developed intuition, we next describe how we find K tiles that best match the object's surface.

4.2.2 Cross Section Selection

To simplify the tile selection problem, we first select a *representative 2D cross section* from the object's 3D geometry, on which we perform a best-fitting procedure given K tiles. We identify the representative cross section as follows. For a given 3D geometric object, we first identify an axis in 3D space where the object is longest; *i.e.*, if the object's 3D points are projected onto this dimension, the range from maximum value to minimum is the largest. As an example, if the 3D object is a person standing up, the longest dimension would be the vertical axis (range will be his height). We denote this axis as the *principal axis* or simply z*-axis*.



(a) cross section (b) surface normal function

Figure 4.2: Example of cross section and surface normal function.

Given the chosen principal axis, we next identify a 2D plane of the object orthogonal to the axis that has the *largest* cross section. In the previous example, this could be a cross section of the person's belly. We perform our 2D optimization for tiling on this representative cross section, as described in the next section.

4.2.3 Tile Selection

Given a 2D representative cross section, we first define a *surface normal function* (SNF), which is the degree (in radian) of the surface normal, computed clockwise from the vertical up direction, as one moves clockwise around the representative cross section. An example of a cross section and its corresponding SNF $F()$ is shown in Fig. 4.2.

As discussed earlier, our goal is to select tiles to approximate the representative cross section as closely as possible. However, if too many tiles or tiles of too complicated shapes are deployed, then the descriptions of the tiles themselves would require too many coding bits, leading to large representation size. Thus the goal is to select a limited number of tiles, each of which required few bits for description, to approximate the cross section as closely as possible.

Given the above considerations, we will use only two kinds of 2D tiles to approximate the representative cross section in this chapter: i) *linear tiles*, ii) *circular tiles*. When considering 3D space, linear and circular tiles extends to *planar* and *cylindrical* tiles respectively along the z-axis. Assuming a change of coordinate is performed so that x and y are axes orthogonal to the principal axis z, the location of these two kinds of tiles can be described using simple equations:

$$ax + y = b \tag{4.1}$$

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{4.2}$$

In particular, each tile can be described by at most three parameters: (a, b) for a planar tile, and (x_0, y_0, r) for a cylindrical tile. Thus, the cost of describing each tile is negligible. Further, one can map the pixels in each tile image easily onto a 2D grid for conventional image / video coding. Note that though in general a tile can be located any distance from the object's surface, for simplicity we will assume in this chapter that a tile is at average distance ξ away from the surface it is approximating, for suitably chosen ξ.

When a linear or circular 2D tile is mapped to the surface normal (SN) domain, they become a *constant* or *linear* function, respectively. Given only linear or circular tiles are used for approximating the representative cross section, the tile selection problem becomes essentially a *piecewise linear approximation* problem in the SN domain, where we constrain the number of pieces used to be K. We assume that the value K—*i.e.* the number of tiles we used—affects directly the overall rate of the coding system. Hence, for a given K, we find the best possible piecewise linear function to approximate the cross section's SNF—a proxy for distortion. By adjusting K we can induce a rate-distortion (RD) tradeoff.

Objective Function

Mathematically, given K we can define our objective function as follows. First, let the representative cross section's SNF be $F(\,)$, and the voxel sampling period on its surface be Δ (with N total samples). Let the circular tile k be represented as a linear function in SN domain as $C_k(t) = m_k t + h_k$, where m_k and h_k are the slope and y-intercept, respectively. For linear tile, its representation in SN domain is simply $L_k(t) = h_k$. Let the boundary between neighboring pieces k and $k+1$ be $\gamma_k\Delta$. The distortion function—l_2-norm between $F(\,)$ and its piecewise linear approximation—can then be written as:

$$D = \sum_{k=1}^{K} \sum_{i=\gamma_{k-1}+1}^{\gamma_k} |F(i\Delta) - \alpha_k C_k(i\Delta) - (1-\alpha_k) L_k(i\Delta)|^2 \tag{4.3}$$

where $\gamma_0 = 0$ and $\gamma_K = N-1$ is the last voxel sample on the cross section, and α_k is a binary variable to denote if a circular or linear tile is used for the kth piece.

Problem Constraints

We consider the following constraints for our optimization problem. First, the right boundary $\gamma_k\Delta$ for the kth piece must come before the next boundary $\gamma_{k+1}\Delta$ in the approximation function, hence

$$\gamma_k < \gamma_{k+1}, \qquad 1 \le k \le K-1 \tag{4.4}$$

Second, though in theory any radius r can be used to describe the circular tile in (4.2), a very large radius will create numerical instability when mapping tile pixels to the object surface. Thus,

we will lower-bound the choice of slope m_k for a circular tile as follows:

$$m_k \geq m_{\min}, \qquad 1 \leq k \leq K \tag{4.5}$$

Finally, as described earlier, α_k is constrained to be a binary variable and can take on only one of two values:

$$\alpha_k \in \{0, 1\} \tag{4.6}$$

Dynamic Programming Algorithm

The optimization is now well defined:

$$\min_{\{\alpha_k, m_k, h_k, \gamma_k\}} D \tag{4.7}$$

such that constraints (4.4), (4.5) and (4.6) are satisfied.

The optimization problem (4.7) can be solved optimally and efficiently using *dynamic programming* (DP). Let $D(i,k)$ be the minimum distortion for voxel samples i to N, if k constant / linear pieces can be used for approximating SNF $F(\,)$. If a single piece is used to approximate samples i till j (resulting in distortion $d(i,j)$), then there will be one fewer pieces $k-1$ for the remaining samples $j+1$ to N. Mathematically, we can write:

$$D(i,k) = \begin{cases} \min\limits_{j=i,\ldots,N} d(i,j) + D(j+1, k-1) & \text{if } \; k \geq 2 \\ d(i,N) & \text{o.w.} \end{cases} \tag{4.8}$$

$d(i,j)$ is the distortion if a piece k is used to approximate voxel samples i to j. We can solve for the optimal tile variables α_k, m_k and h_k using *linear regression* [6]. Specifically, we first solve for the best-fit m_k^* and h_k^*:

$$\begin{aligned} m_k^* &= \frac{\overline{\theta_k t_k} - \bar{\theta}_k \bar{t}_k}{\overline{t_k^2} - (\bar{t}_k)^2} \\ h_k^* &= \bar{\theta}_k - m_k^* - \bar{t}_k \end{aligned} \tag{4.9}$$

where $\bar{t}_k$ and $\bar{\theta}_k$ are the average surface distance and surface normal angle respectively, $\overline{t_k^2}$ is the average of the square distance, and $\overline{\theta_k t_k}$ is the average of the product of distance and angle, for voxel samples i to j.

If the best-fit slope m_k^* is smaller than $m_{\min}$, we then compare the solutions when slope is $m_{\min}$ (circular tile) and when slope is 0 (linear tile). The solution with the smaller square error with respect to $F(\,)$ will be chosen.

The complexity of the DP algorithm (4.8), with initial call $D(1,K)$, can be analyzed as follows. The size of of the DP table to contain solutions to sub-problems $D(i,k)$ is $O(NK)$. To compute each entry in the DP table, the operation in (4.8) is $O(N)$. Hence the complexity of the algorithm is $O(N^2K)$.

4.3 Experimentation

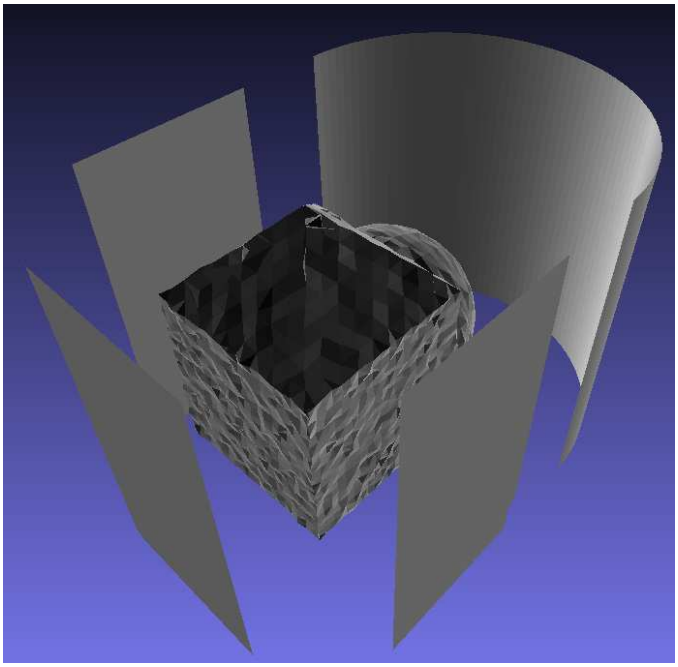
To demonstrate the value of optimizing tile placement, we conducted the following experiment. To find the optimal tile placement for a given static geometry, we use our proposed DP algorithm discussed in Section 4.2.3 for a given budget of tiles K. Fig. 4.3(a) illustrates the optimal tile placement for the 3D mesh when $K = 4$, while a naïve scheme uses four planar tiles. The prediction structure is IPPP and the quantization parameters are 5, 10, 15, 20, and 25. The generated depth maps are coded using either DCT based encoder (HEVC) or GFT based encoder[1]. Fig. 4.3(b) shows the RD performance of the two schemes, where maximum root mean square (RMS) of the metro distance [20] is used to measure 3D reconstruction distortion. The performance curve of our proposed method is the convex hull of operational points obtained by varying QP and the number pieces K used for approximation of SNF. We see that our proposed method outperformed the naïve method at all bitrates.

Fig. 4.4 shows the performance of another 3D mesh. The 3-tile competing scheme is obtained by replacing cylindrical tile with a planar tile. The 4-tile competing scheme replaces the cylinder tile by two orthogonal planar tiles, *i.e.*, the four tiles face North, East, South and West direction, respectively. From both experiments, we found that our proposed tile selection scheme outperforms the naïve tile placements. The gain is most pronounced at low rate, where up to 35% bit rate reduction can be observed. We observe also that more gain is achieved by using GFT, where piecewise smooth signals can be represented more sparsely compared to DCT.

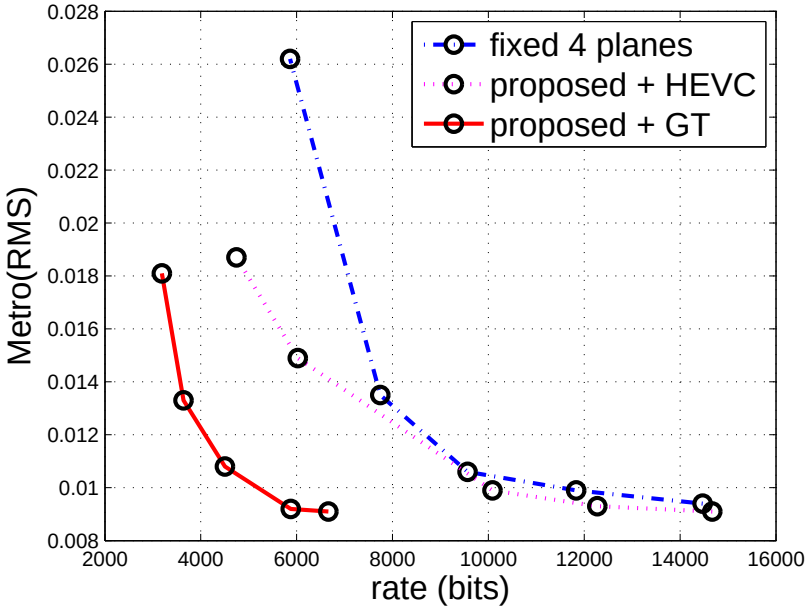
4.4 Summary

We presented an image tiling framework for representation of an object's dynamic 3D geometry, where the object's 3D mesh is first projected as 2D images to a set of carefully placed image tiles, before a multiview image codec tuned for piecewise smooth signals is deployed for coding of tile images. The key to a compact yet accurate representation is the selection and placement of image tiles. We showed that if only planar and cylindrical tiles are considered, the optimal tile placement problem (a finite tile set that best matches the object's representative 2D cross section) can be mapped to a piecewise linear approximation problem. The approximation problem can be subsequently solved efficiently using a dynamic programming algorithm. Experimental results show that optimal tile placements can outperform naïve tile placements by up to 35% in rate reduction.

[1] Implementation of GFT used was found here: http://biron.usc.edu/~kumarsun/Codes/GraphTransform_Matlab.zip

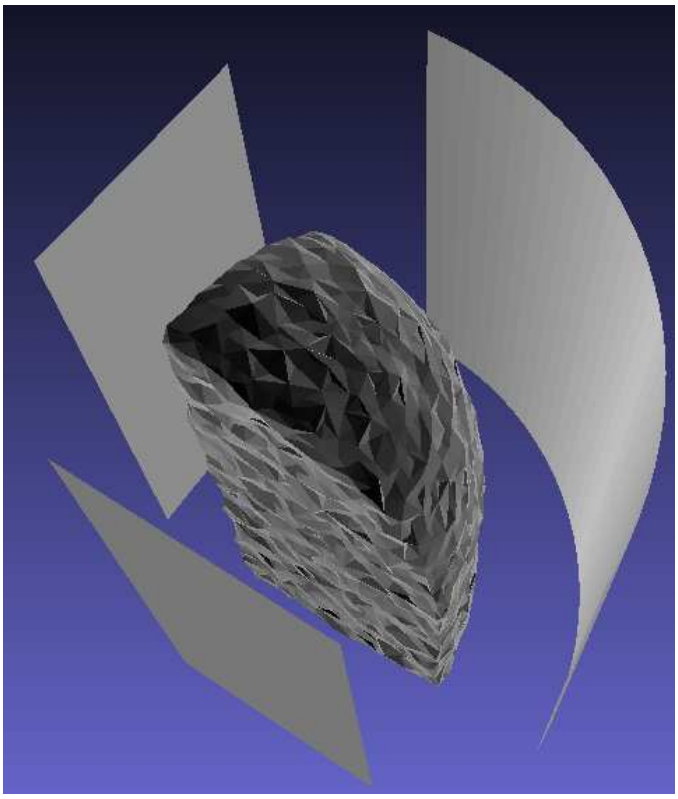


(a) selected tiles

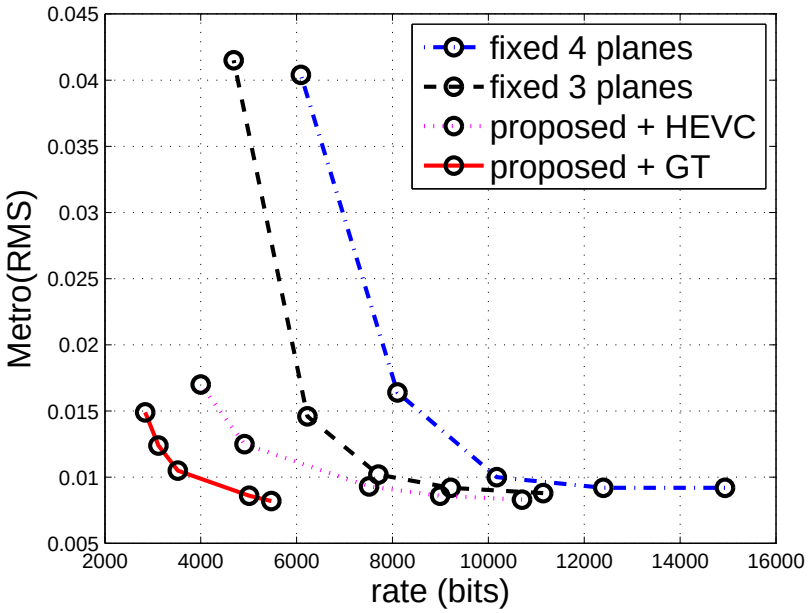


(b) RD performance

Figure 4.3: Result for the first 3D mesh.



(a) selected tiles



(b) RD performance

Figure 4.4: Result for the second 3D mesh.

Chapter 5

Optimizing Frame Structure with Real-time Computation for IMVS

In the previous three chapters, we focus on rate-distortion (RD) performance. In the next two chapters, we study applications where real-time computation and the associated complexity also need to be considered. In this chapter, we consider interactive multiview video streaming (IMVS), an application where a network client requests from server a single video view at a time but can periodically switch to other views as the video is played back uninterrupted. Existing IMVS algorithms output pre-computed frame structures that facilitate permissible view-switching while minimizing the expected transmission rate given a storage constraint. Here, we propose to use real-time computation (available at a remote powerful server or media cloud) to assist the pre-computed frame structure to satisfy users' view-switch requests. In particular, we first propose a new frame type called uni-merge frame that is computed in real-time for view-switching from one single view to one target view with low transmission rate and reasonable computation cost. Then, to enable permissible view-switches to a particular target picture, we find the optimal combination of pre-computed frames and real-time computed frames—one that minimizes streaming rate subject to both storage and real-time computation constraints—using a greedy combinatorial algorithm. Experimental results show that with real-time computation, the expected streaming rate of the IMVS system can be further decreased by 50% compared to pre-encoded frame structures without real-time computation.

5.1 Introduction

Multiview videos refer to videos of the same 3D scene captured by multiple closely spaced cameras from different viewpoints. They can enable visually immersive applications such as free viewpoint TV [94], virtual walk-through, etc. However, storage and transmission of multiview video data are

very challenging, due to the large amount of visual data involved. As a result, there has been extensive research in multiview video coding (MVC) [70], where the goal is to compress video frames of all captured views across time in a RD optimal manner.

In many applications, however, not all views are required at the client at the same time. In IMVS [15], a network client only requests from the server one captured view at a time for rendering on conventional 2D display, but can switch to other viewpoints periodically during video playback. The view-switching period is usually set very small to provide smooth view-switching visual experience.

To support frequent view-switching in IMVS without incurring large transmission cost, instead of MVC structures (where inter-view predictions create complicated inter-dependency among frames, limiting random access), previous IMVS studies [15, 104] proposed redundant frame representation, so that the expected transmission rate can be reduced at the cost of increased storage. In particular, combinations of redundant P-frames (with low transmission rate but large storage cost) and merge frames (based on distributed source coding (DSC) [18] to merge multiple decoding paths, with high transmission rate but low storage cost) that optimally trade off between the expected transmission rate and storage required to contain the redundant frame structure are sought in a combinatorial optimization.

The coding structures in the existing IMVS schemes [15, 104] are pre-computed and stored, incurring storage cost. With the advent of parallel and cloud computing, it is now possible to perform a limited amount of video processing tasks in real-time on demand [59, 93, 102] at affordable computation cost. In this chapter, we optimize the design of IMVS redundant frame structure with the help of real-time computation. In particular, we first propose a new frame type called uni-merge frame[1] that is computed in real-time for view-switching from one single view to one target view with low transmission rate and reasonable computation cost. Then, to enable permissible view-switches to a particular target picture, we find the optimal combination of pre-computed frames and real-time computed frames—one that minimizes expected streaming rate subject to both storage and real-time computation cost constraints—using a greedy combinatorial optimization.

5.2 Real-time Computation for IMVS

When a computation-intensive input-to-output information processing task needs to be performed repeatedly over time, instead of computing all possible input-to-output mappings in real-time, mappings corresponding to the more frequently occurring inputs can be pre-computed and stored in memory, so that during real-time processing, the pre-computed results can be simply looked up and returned. Finding the optimal mixture of real-time computation and pre-computing partial results in

[1] Previously proposed merge frame [15] will henceforth be called multi-merge frame.

storage for a given processing task is a fundamental problem in algorithm implementation [14], and has been successfully investigated in various problem settings such as IP address lookups in network routers [88] and scalar and vector quantization encoding [14].

With the advent of parallel and cloud computing, real-time computation for some video processing operations can become affordable. We hence revisit the IMVS structure design problem, further optimizing the streaming rate / storage tradeoff with the help of real-time computation. In particular, we study the tradeoff among streaming rate, storage and real-time computation costs for previously proposed view-switching tools [15]—redundant P-frames and multi-merge frame—and a new tools called uni-merge frame, and optimize new frame structures based on our analysis.

5.3 System Description

We consider an IMVS system that offers *dynamic* and *static* view-switching every N frames in time for streaming clients. In other words, after playing back the video of a single view for N temporal frames, a client can either switch to one of the other captured views as video continues playback in time, or freeze in time and switch to other views. To enable this view-switching functionality efficiently, at a particular view-switching point, we find the optimal structure composed of pre-computed frames and real-time computed frames to minimize expected streaming rate subject to storage and real-time computation constraints.

Let $F_{i,j}$ be a *picture group* that includes N pictures[2] of the j-th view with time instants $iN, iN + 1, \ldots, (i+1)N - 1$. A view switch from group $F_{m,n}$ to $F_{i,j}$ is denoted as $(m,n) \rightarrow (i,j)$. In this chapter, only switches within a view distance of K views are supported. Thus, a legal dynamic view-switch can be represented by $(i-1,n) \rightarrow (i,j)$, and a legal static view-switch by $(i,n) \rightarrow (i,j)$, where $j - K \leq n \leq j + K$ in both cases.

Our goal is to design a *frame structure* $\mathcal{S}_{i,j}$ to represent pictures in group $F_{i,j}$, one that facilitates all legal view-switches to $F_{i,j}$, while achieving the optimal tradeoff among transmission rate, storage and real-time computation. In each structure $\mathcal{S}_{i,j}$, the first picture of instant iN can be represented in multiple versions as *redundant P-frames*, or a single version as a *merge frame* [15]. The remaining pictures in the group are each coded as a P-frame, motion-compensated using the previous temporal frame of the same view as predictor. All coded versions of the first picture must reconstruct to exactly the same frame to avoid coding drift in the following differentially coded frames. We accomplish that using DSC frames [18], as discussed below.

[2]We will use "picture" to denote the original captured image, and "frame" to denote a coded version of a picture.

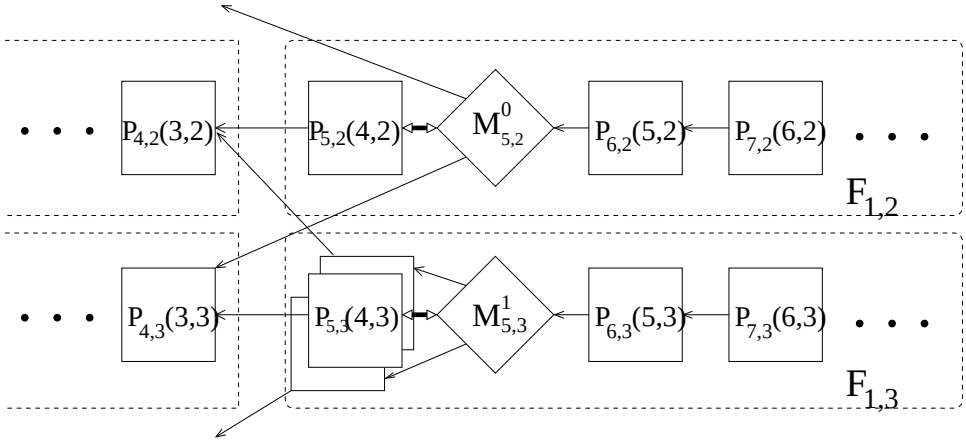


Figure 5.1: Coding structure examples with P-/DSC0-/DSC1-frames, where single-headed arrows denote prediction dependency and double-headed arrows denote merge operations using DSC0- and DSC1-frames.

5.3.1 Distributed Source Coding Frames

We employ two types of DSC frames with different tradeoffs between storage and transmission rate [18]. The first type is called DSC0, which includes a set of motion vectors (MV) and low-density parity check (LDPC) code. In particular, for each legal view-switch from a predictor frame to target frame, motion estimation is first performed, resulting in a set of MVs and an estimate of the target frame (*side information* (SI)). LDPC code is then used to remove the difference (noise) between SI and the target frame. Note that only one set of LDPC code is used for all possible predictor frames, so that no matter which SI is available from which predictor frame, the same DSC0 frame can be reconstructed. DSC0 is a *multi-merge* frame, as multiple decoding paths are merged at the target frame so that the same target picture can be reconstructed.

The second DSC frame type DSC1 is used together with redundant P-frames. Specifically, motion compensation is first performed from each legal view-switching reference frame, resulting in a set of P-frames. Then, LDPC code is generated to remove the difference between these redundant P-frames and the target frame. Though the coding mechanism is the same, DSC0 acts as a multi-merge frame, while the much smaller DSC1 (redundant P-frames are of the same view and time instant as the target frame, resulting in better quality SI with small noise) acts as a "denoising" frame. Compared to DSC0 frame, the combination of redundant P-frames plus DSC1 frame requires more storage, but it has lower transmission rate, thanks to the smaller DSC1 frame.

Usage of DSC0 and DSC1 is illustrated in Fig. 5.1 for $N = 5$. $P_{i,j}(m,n)$ denotes a P-frame (square) for instant i and view j using frame of instant m and view n for prediction. $M^0_{i,j}$ and $M^1_{i,j}$ denote DSC0 and DSC1 frame (diamond), respectively. Note that $M^1_{i,j}$ is used in combination with

redundant P-frames. Note further that the target frame for DSC frame $M_{i,j}$ is chosen to be a *re-quantized* version[3] of P-frame $P_{i,j}(i-1, j)$; i.e., the DSC frame will reconstruct to be bit-by-bit equivalent to re-quantized $P_{i,j}(i-1, j)$. This is done so that during normal video playback in the same view (the most likely view-switch), only P-frame $P_{i,j}(i-1, j)$ needs to be transmitted.

Turbo code is an alternative to implement DSC-frame, and the proposed system is possible to adopt any DSC implementations. However, LDPC code has better performance in high-rate range, and is more suitable for distributed coding [62]. Also, LDPC decoder can be practically implemented.

5.3.2 Tradeoffs in IMVS View-switching Tools

Having discussed previous view-switching tools for IMVS [15], we now overview the tradeoffs in streaming rate, storage and real-time computation for these tools for intuition. First, consider the case when real-time computation is expensive. In this case, DSC0 frame would offer the most storage-efficient view-switching solution, since no extra P-frames are stored, though it would result in a large transmission rate due to the large DSC0 frame size. Combination of redundant P-frames and DSC1 frame would offer a lower transmission rate solution, due to the smaller DSC1 frame size. However, the redundant P-frames (for large K) would lead to large storage cost.

Consider now the case when real-time computation is affordable. For the combination of redundant P-frames plus DSC1 frame $M^1_{i,j}$, instead of pre-computing and storing redundant P-frames $P_{i,j}(m,n)$'s for all legal view-switches $(m,n) \rightarrow (i,j)$, we can now store only the frequently accessed P-frames, while the less accessed P-frames are computed in real-time. The real-time computed P-frames are discarded after use to avoid storage cost.

In contrast, one can use real-time computation to change the encoding of DSC0: DSC0 can be computed in real-time on demand for a *single* view-switch $(m,n) \rightarrow (i,j)$. That means only one set of MVs for a particular predictor frame plus LDPC code strong enough to overcome the noise in this particular SI is sufficient to perfectly reconstruct the target frame. We denote this real-time computed frame as DSC0-RT $M^0_{i,j}(m,n)$. Note that because it handles only a single view-to-view switch, the size of this DSC0-RT frame is much smaller than pre-computed DSC0 frame, and smaller than $P_{i,j}(m,n)$ plus DSC1 $M^1_{i,j}$. Further, instead of using channel code like LDPC to remove noise, one can alternatively use differential coding techniques like H.264's secondary SP-frames [55] to perfectly reconstruct the target frame. This real-time computed *uni-merge* frame (DSC0-RT)—one with low transmission rate and reasonable real-time computation cost—to switch from a single view to a target view is in stark contrast to the pre-computed *multi-merge* frame (DSC0) that must necessarily merge multiple decoding paths for best tradeoff between transmission rate and storage.

[3] Re-quantization means the decoded P-frame is re-encoded as an I-frame. This is done so that LDPC code in the DSC frames only have to match the quantization bin index of each transform coefficient, lowering LDPC encoding rate.

We next discuss how the best combination of DSC0, redundant P-frames plus DSC1, and DSC0-RT can be found through a greedy optimization.

5.4 Frame Structure Optimization

5.4.1 Problem Formulation

We now formulate our objective function. We first assume a user switches from group $F_{m,n}$ to $F_{i,j}$ with view-switch probability $p_{i,j}(m,n)$, where $\sum_{i,j} p_{i,j}(m,n) = 1$, $\forall(m,n)$. The normal playback probability $p_{i,j}(i-1,j)$ will be the largest relative to other switches. Let $\pi_{i,j}$ be the steady-state probability of $F_{i,j}$.

Let $|\mathcal{S}_{i,j}|$ be the size of structure $\mathcal{S}_{i,j}$, and $S^X_{i,j}(m,n)$ and $S^C_{i,j}(m,n)$ be the transmission rate and real-time computation cost associated with the view-switching from $F_{m,n}$ to $F_{i,j}$, respectively. Given storage and computation budget $\bar{S}$ and $\bar{C}$ for the entire multiview video, the constrained optimization is written as:

$$\begin{aligned} \min \quad & \sum_{i,j}\sum_{m,n} \pi_{m,n} p_{i,j}(m,n)\, S^X_{i,j}(m,n) \\ \text{s.t.} \quad & \sum_{i,j} |\mathcal{S}_{i,j}| \le \bar{S}, \quad \sum_{i,j}\sum_{m,n} \pi_{m,n} p_{i,j}(m,n) S^C_{i,j}(m,n) \le \bar{C} \end{aligned} \tag{5.1}$$

where the optimization variables are the structures $\mathcal{S}_{i,j}$'s for groups $F_{i,j}$'s in the video.

Instead of solving the constrained problem (5.1), we solve the unconstrained version problem using Lagrange multipliers λ and μ for the two constraints:

$$\begin{aligned} \min \quad & \sum_{i,j}\sum_{m,n} \pi_{m,n} p_{i,j}(m,n)\, S^X_{i,j}(m,n) + \\ & \lambda \sum_{i,j} |\mathcal{S}_{i,j}| \; + \; \mu \sum_{i,j}\sum_{m,n} \pi_{m,n} p_{i,j}(m,n) S^C_{i,j}(m,n) \end{aligned} \tag{5.2}$$

where λ and μ need be adjusted so that the optimal solution to (5.2) meets the two constraints in original (5.1).

It is clear that (5.2) can be solved separately for each group $F_{i,j}$ without losing optimality:

$$\min \; \lambda|\mathcal{S}_{i,j}| \; + \; \sum_{m,n} \pi_{m,n} p_{i,j}(m,n) \left(S^X_{i,j}(m,n) + \mu S^C_{i,j}(m,n) \right) \tag{5.3}$$

Hence we next describe our algorithm to find the optimal structure $\mathcal{S}_{i,j}$ for each group $F_{i,j}$ in (5.3).

5.4.2 Greedy Optimization

Following our discussion in Section 5.3.2, we see that there are three logical options for structure $\mathcal{S}_{i,j}$. If real-time computation and storage costs are both expensive (weighted by $\pi_{m,n}\, p_{i,j}(m,n)\, \mu$ and λ in (5.3) respectively), then pre-computing DSC0 frame is optimal—solution with smallest storage size without real-time computation cost. If real-time computation cost is very cheap, then DSC0-RT frame is optimal—solution with smallest streaming rate and no storage cost. For other cases,

pre-computing DSC1 frame with redundant P-frames optimally divided between pre-compute and real-time computation would be a good solution. Hence we can devise our optimization strategy as follows to check for the performance of these three basic structures.

1. Pre-compute a DSC0 frame $M^0_{i,j}$ for storage and calculate objective (5.3) as cost J^0.
2. Pre-compute a DSC1 frame $M^1_{i,j}$ for storage.
3. Incrementally add the *most beneficial* redundant P-frame $P_{i,j}(m,n)$ for a legal view-switch $(m,n) \rightarrow (i,j)$, *i.e.*, one that lowers objective (5.3) the most. Stop when there are no more beneficial P-frame to add.
4. Calculate objective (5.3) as cost J^1.
5. Real-time compute DSC0-RT frame $M^0_{i,j}(m,n)$ for all legal view-switches $(m,n) \rightarrow (i,j)$. Calculate objective (5.3) as cost J^2.

The structure that corresponds to the smallest of the three costs J^0, J^1 and J^2 would be the optimal structure we choose for $\mathcal{S}_{i,j}$.

5.5 Experimentation

For intuition, we first illustrate when the three basic structures become optimal for different prices of storage and computation. Then we compare our proposed scheme to a competing scheme that does not utilize real-time computation and all frames are pre-computed and stored.

The video playback probability without view-switch $p_{i,j}(i-1,j)$ is denoted by p_{pb} for short. Let $p_1 = (1-p_{\text{pb}})/(K\cdot(K+1))$. The view-switch probability from view j to view $j \pm k$ is $(K+1-k)\,p_1$ for $k \in [1,K]$. This includes both static and dynamic view-switches. In our first experiment, the steady state probability $\pi_{i,j}$ is set to be $0.5/(2K+1)$, where K is the maximal view-switch distance.

We first tune the Lagrange multipliers λ and μ in (5.3) to show the influence of storage and computation prices on the optimal structure. The picture group $F_{2,3}$ of the multiview video sequence `Kendo` is coded into a structure generated using our proposed scheme, with $K=3$, $N=4$ (picture group size). The results are summarized in Table 5.1 for different p_{pb}.

When storage and real-time computation are both expensive, J^0 is smaller than J^1 and J^2, *i.e.*, the pre-computed DSC0 frame merging all legal view-switches is optimal, since it has the smallest storage and no real-time computation cost. When storage is very cheap, the pre-computed DSC1 with all redundant P-frames also pre-computed is the best choice, where the transmission rate is lower than the pre-computed DSC0 frame and no real-time computation cost is paid. In usual cases, DSC1 with combination of pre-computed and real-time computed P-frames has the lowest total cost (only one typical combination of λ and μ for usual case is shown in Table 5.1). When the real-time

Table 5.1: Costs of different structures.

p_{pb}	λ	μ	J_0	J_1 (computation cost)	J_2
0.5	0.1	10	**114212**	128299 (35710)	14296445
	0.0001	0.1	47655	**20593 (0)**	149730
	0.01	0.1	54251	**25821 (714)**	150113
	0.01	0.01	54251	25178 (71)	**21541**
0.9	1	200	**713824**	762087 (285800)	57185054
	0.00001	0.1	47595	**6307 (0)**	32152
	0.01	0.1	54251	**11131 (143)**	32538
	0.01	0.01	54251	11003 (14)	**6824**

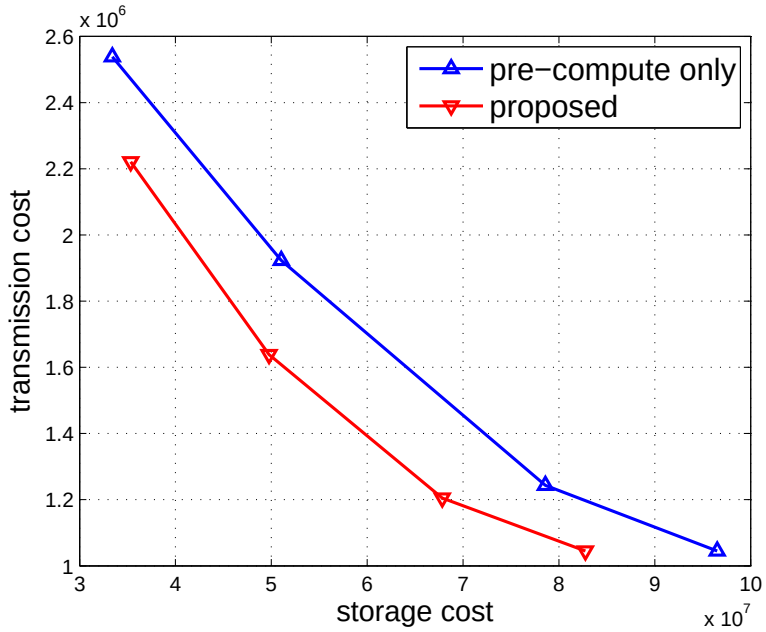


Figure 5.2: Tradeoff between storage and transmission with $p_{\text{pb}} = 0.5$.

computation is cheap, DSC0-RT is the optimal structure, since it has low transmission rate and no storage cost.

We next consider the optimization of entire multiview video sequence and show the tradeoff between storage and transmission with or without real-time computation. 50 picture groups, each of 4 pictures, from sequence `Kendo` are considered. In this experiment, the steady state probabilities are randomly generated. Note that the computation budget $\bar{C}$ for the proposed scheme is fixed.

The results are plotted in Fig. 5.2 and 5.3, showing that the proposed scheme outperforms the competing scheme without consideration for real-time computation. In particular, the streaming rate can be decreased by approximately 50%, which demonstrates the importance of introducing the real-time computation to IMVS.

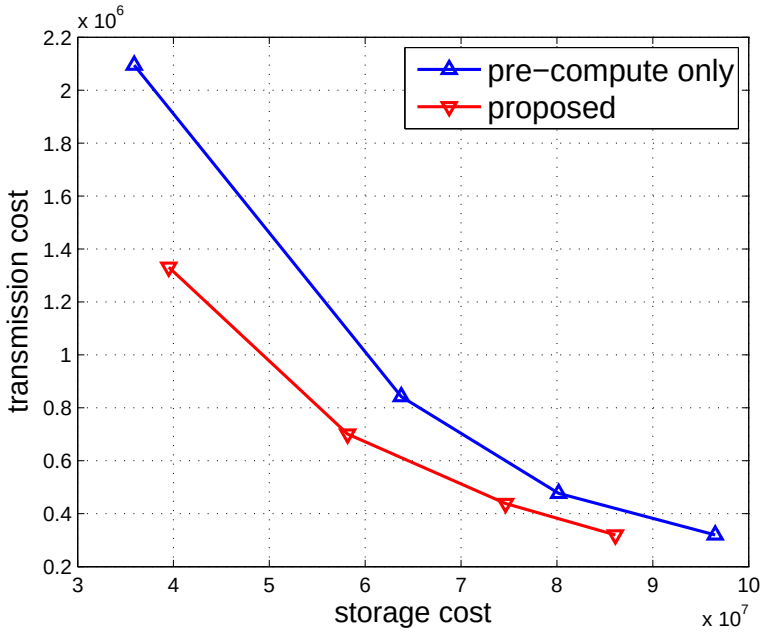


Figure 5.3: Tradeoff between storage and transmission with $p_{\text{pb}} = 0.9$.

5.6 Summary

Unlike previous work on interactive multiview video streaming (IMVS) that studied the tradeoff between expected streaming rate and storage cost when optimizing frame structures, we proposed to redesign frame structures with the help of available real-time computation, where frequently used view-switches are handled by pre-computed frames in storage, and infrequently used view-switches are handled by real-time computed frames. In contrast to multi-merge frames previously proposed that offer good tradeoff between transmission rate and storage cost, we proposed a uni-merge frame that is computed in real-time and offers good tradeoff between transmission rate and real-time computation cost. Experimental results show that with real-time computation, the expected streaming rate can be further decreased by 50% compared to pre-encoded structures without real-time computation.

Chapter 6

Rate-Complexity Tradeoff for Client-side FVV

In this chapter, we consider free viewpoint video (FVV) that enables a client to interactively choose a viewpoint from which to synthesize an image via depth image based rendering (DIBR). However, synthesizing a novel viewpoint image entails a sizable computation overhead. We study the optimal tradeoff between transmission rate and the complexity of view synthesis at the client side, so that in the event that a client device is computation-constrained, complexity of DIBR-based view synthesis can be scalably reduced at the expense of a controlled increase in transmission rate. Specifically, for standard view synthesis paradigm that requires texture and depth maps of two neighboring reference views, we design a dynamic programming algorithm to select the optimal subset of intermediate virtual views for rendering and encoding at the server, so that a client performs only video decoding of these views, reducing overall view synthesis complexity. For the new view synthesis paradigm discussed in Chapter 2 that synthesizes the second reference view itself from the first, we optimize the transmission of AI used to assist inpainting of large disocclusion holes, so that some computation-expensive exemplar block search operations are avoided, reducing inpainting complexity. Experimental results show that the proposed schemes can scalably and gracefully reduce client-side complexity, and the proposed optimizations achieve better rate-complexity tradeoff than competing schemes.

6.1 Introduction

FVV [94] enables a client to interactively choose a virtual viewpoint from which to synthesize an image via DIBR [97]. While observation of the 3D scene from different viewpoints can enhance depth perception in the viewer [112], the DIBR view synthesis process using texture and depth maps

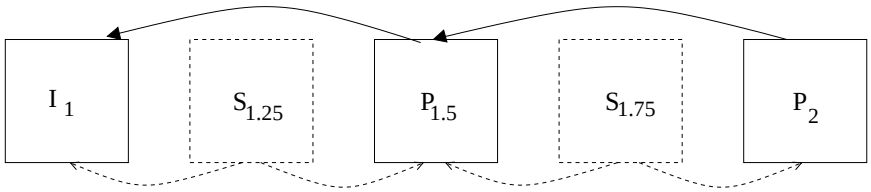


Figure 6.1: Example of rate-complexity tradeoff for low-spec devices.

captured from two nearby views entails a sizable computation overhead. Further, to reduce the transmission rate of FVV, instead of explicitly encoding the second reference view for the synthesis of intermediate views, our work in Chapter 2 call for the view synthesis of the second reference view itself using only texture and depth maps of the first reference view. Using only one reference view for view synthesis typically results in large disocclusion holes in the target image, which necessitates transmission of a small amount of auxiliary information (AI) to assist more complex inpainting algorithms [21] to complete the image satisfactorily. For computation-constrained devices like tablets, this computation load may be overwhelming.

To address the client complexity problem, we study the optimal tradeoff between transmission rate and client-side complexity, so that in the event that a client device is computation-constrained, the complexity of DIBR-based view synthesis can be gracefully and scalably reduced at the expense of a controlled increase in transmission rate. We first consider the rate-complexity (RC) tradeoff for standard virtual view synthesis paradigm that requires texture and depth maps from two neighboring reference views, as commonly done in the free viewpoint literature [94, 97]. The resulting synthesized image typically has small disoccluded regions that can be filled using simple standard procedures [97]. In this case, we design a dynamic programming (DP) algorithm to select the optimal subset of virtual views between two reference views for rendering and encoding at the server, so that a client that desires a free viewpoint "look-around" from the first reference viewpoint to the second can perform video decoding of these frames, reducing overall synthesis complexity. As an example in Fig. 6.1, in addition to reference views 1 and 2, virtual view 1.5 is chosen to be rendered and encoded as a P-frame $P_{1,5}$, so that only virtual views 1.25 and 1.75 are synthesized at client (each using two nearby encoded frames as reference, shown as dashed lines), reducing overall complexity.

For new view synthesis paradigm discussed in Chapter 2 that synthesizes the second reference view using texture and depth map of the first reference, we study the RC tradeoff for the construction of the second reference view by controlling the selection of AI to assist inpainting of large disocclusion holes. More specifically, if a missing block requires high-complexity search in the filled-in region to identify a suitable exemplar block for copying, then an intra block can be transmitted instead to complete the block at a cost of AI transmission rate increase. See Fig. 6.2 for an illustration. Virtual views 1.25, 1.5, 1.75 are synthesized at client using views 1 and 2 as reference (shown as dashed

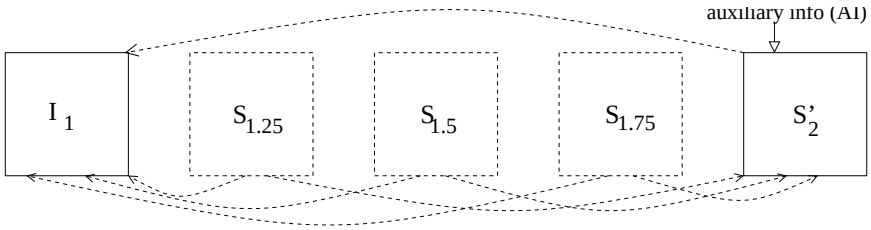


Figure 6.2: Example of rate-complexity tradeoff for high-spec devices.

lines). Reference view 2 is reconstructed via view synthesis from reference view 1, with the help of transmitted AI to inpaint disoccluded regions to complete the image. The RC-optimal sequence of AIs is selected at the sender by finding the shortest path in a trellis.

The outline of the chapter is as follows. We first overview our system model in Section 6.2. We formulate our RC optimization for two types of devices in Section 6.3. Finally, experimental results and conclusions are presented in Section 6.4 and 6.5, respectively.

6.2 System Overview

Like [104], we consider an IMVS scenario where a server pre-encodes and stores a multiview video content of V captured views, $v \in \{1, \ldots, V\}$, where v corresponds to the physical location of a camera in a 1D array. The IMVS interactivity we provide is *static view-switching*, which means a user can stop the playback of the video in time and navigate to neighboring virtual views of the paused 3D scene[1]. Specifically, a client observes one virtual view at a time denoted by $v + k/K$, where $k \in \mathbb{I}$ and $1 \leq k \leq K - 1$. In essence, the client observes a "look-around" of the static 3D scene by viewing virtual views from v to $v + 1$. Upon arriving at view $v + 1$, the client then has the option of either continuing the static 3D scene "look-around" to camera view $v + 2$, or starting temporal video playback at view $v+1$. Dependent on the baseline distance between neighboring cameras, K should be large enough to support a smooth-switching user experience.

Unlike [104] that seeks to minimize expected transmission rate while facilitating application-required periodic view-switching, the challenge in this chapter is to design *additional* coded data for transmission, so that complexity of view synthesis at client can be scalably reduced. To understand the computation complexity for the new view synthesis paradigm, we first overview the AI scheme where texture and depth maps for the second reference view $v + 1$ is synthesized using texture and depth maps of a first reference view v, with the help of transmitted AI to aid the inpainting process of large disoccluded regions. It was demonstrated that such a representation has better

[1] [63] showed that humans prefer the visual effects of static view-switching over *dynamic view-switching* [15], where the video is played back in time uninterrupted as users interactively switch to neighboring views. The latter produces effects similar to single-camera pan, which is not novel.

rate-distortion (RD) performance compared to multiview video coding (MVC) [70] and layered depth video (LDV) [81].

6.2.1 Auxiliary Information (AI) for Hole-filling

As discussed in Chapter 2, the second reference view is constructed as follows. First, texture pixels from the first reference view are mapped to their corresponding locations in the second reference view according to their disparity values[2]. These are the known pixels in the source region Φ. To fill in values in the disoccluded pixels in the target region Ω, the additional AI is transmitted, as reviewed below. We designates a filling order for code blocks with center on the boundary $\delta\Omega$ between source Φ and target region Ω. There are three types of AI to assist in block completion: `skip`, `vec`, `intra`.

1. `skip` means missing pixels on a block with center on boundary $\delta\Omega$ can be capably inpainted (copied) by the most similar block in Φ, found using the Criminisi's search. Thus, no further information need to be transmitted.

2. `vec` means that Criminisi's search cannot identify a good quality block in source region Φ as replacement. Thus a transmitted similarity vector (SV) can help directly locate a best-matched block in Φ for copying.

3. `intra` means no similar block exists in source region Φ. Thus an intra coded block is transmitted for pixel completion.

For our AI implementation, we implemented these three types of AI with three additional modifications beyond the work in Chapter 2. First, `skip` can specify a search range θ, so that complexity for the Criminisi's search can be scalably reduced. Second, if the previous block is coded as `vec`, then the current block also coded as `vec` can have its SV differentially coded, reducing transmission overhead. Third, there is a one-bit flag in `intra` indicating the type of intra-prediction performed. Specifically, if the previous block is also coded as `intra`, then the one bit indicates whether the intra prediction should be performed using the previous block, or using the pixels across boundary $\delta\Omega$, with the isophote[3] as computed in the Criminisi's algorithm [21] as prediction direction. If the previous block is not coded as `intra`, then the one bit indicates whether intra prediction should not be performed, or performed using the pixels across $\delta\Omega$ as described earlier. The properties of the three types of AI are summarized in Table 6.1.

[2]Since there is a one-to-one correspondence between texture and depth, we will use these terms interchangeably.

[3]Isophote is basically the gradient at a pixel rotated by 90 degrees [21], which empirically we found to be a good intra-prediction direction.

Table 6.1: Rate-complexity of three types of AI. SV is short for similarity vector.

AI	Server Side	Client Side	Bit Rate	Decoder Complexity
`skip`(θ)	signal the search range θ	perform Criminisi's search inside specified range	low	moderate or high depending on θ
`vec`	template matching, encode, send SV	decode SV, block copy	moderate	low
`intra`	(intra-prediction), encode, send block	(intra-prediction), decode block	high	low

6.2.2 Choosing AI for RC Tradeoff

From Table 6.1, one can find the tradeoff between bit-rate and client-side complexity for different AIs. Suppose it is required that one must achieve a certain reconstruction quality for every block. `intra` is capable of reconstructing to any desired quality (specified by the quantization parameter (QP)) at low decoder complexity. However, the transmission cost of AI `intra` is the highest. If a good matching block does exist in the source region (one that satisfies the quality requirement), its location can be explicitly specified by AI `vec`, with a medium transmission cost, or search using the Criminisi's algorithm, which will incur a large computation cost at decoder. The RC-optimal selection of AI will be formulated in Section 6.3.2.

6.3 Problem Formulation

We divide the RC optimization of client-side virtual view synthesis into two sections. We first formulate the RC optimization for standard view synthesis paradigm, where each virtual view is synthesized using two nearby reference views. We then formulate RC optimization for a new view synthesis paradigm, where the second reference view is first synthesized from the first reference, and then disoccluded region is constructed using our proposed complexity-scalable inpainting algorithm.

6.3.1 Rate-Complexity Tradeoff for Standard View Synthesis

For decoders that adopt the standard view synthesis paradigm [97], reference view v and $v+1$, will be encoded as video frames, so that intermediate views between them can be synthesized using *two* references via DIBR. The RC tradeoff is how to select *additional* virtual views between them for rendering and encoding at server, so that complexity at client can be optimally reduced. More specifically, given network bandwidth can support M additional encoded frames for transmission, how to select M intermediate virtual views for encoding so that the complexity of synthesizing the remaining views is minimized.

For simplicity, we assume complexity of synthesizing a view in DIBR is a weighted sum of the

number of translated pixels[4] *plus* the number of inpainted pixels due to disocclusion. More precisely, let $\psi(u,w)$ be the complexity of synthesizing intermediate virtual views $u+1/K,\ldots,w-1/K$ between u and w, if each is synthesized via DIBR using encoded views u and w as left and right reference views, respectively. We write:

$$\psi(u,w) = \sum_{y=u+1/K}^{w-1/K} g_y(u,w) + \mu\, h_y(u,w) \tag{6.1}$$

where $g_y(u,w)$ and $h_y(u,w)$ are the numbers of translated and inpainted pixels in virtual view y respectively, given left and right reference views u and w are used during synthesis, and μ is a weighting parameter. If a computation-intensive inpainting method [23] is used for disocclusion hole-filling, then μ is assigned a value $\gg 1$.

The objective is to find M additional encoded views, $u_1,\ldots,u_M$, between reference views v and $v+1$, so that the overall complexity is minimized:

$$\min_{u_1,\ldots,u_M} C_L = \sum_{m=0}^{M} \psi(u_m,u_{m+1}) \quad \text{s.t.} \begin{cases} u_0 = v \\ u_{M+1} = v+1 \\ u_m < u_{m+1} \end{cases} \tag{6.2}$$

(6.2) can be solved recursively as follows. We first define $C_L(u,m)$ as the minimum complexity from intermediate view $u+1/K$ till $v+1-1/K$, given view u is an encoded view and m remaining intermediate views can be selected for encoding. $C_L(u,m)$ can be defined recursively as:

$$C_L(u,m) = \begin{cases} \min_w \psi(u,w) + C_L(w,m-1) & \text{if } m \geq 1 \\ \psi(u,v+1) & \text{o.w.} \end{cases} \tag{6.3}$$

Using (6.3), a recursive call to $C_L(v,M)$ will yield the optimal solution to (6.2). Further, the complexity of computing (6.3) can be reduced via *dynamic programming* (DP): each time $C_L(u,m)$ or $\psi(u,w)$ is computed, the solution is stored in entry $[u,m]$ or $[u,w]$ of DP tables, so that repeated calls to the same sub-problem can be looked up instead. This is particularly helpful if (6.2) needs to be solved multiple times for different M's.

6.3.2 Rate-Complexity Tradeoff for New View Synthesis

We now formulate the RC optimization problem for the new view synthesis paradigm, where the virtual views between references v and $v+1$ are synthesized via DIBR using the two reference images, but the second reference $v+1$ is first synthesized using texture and depth maps of the first reference v only. The problem we pose is how to first construct the second reference image $v+1$ given the first reference v, using a complexity-scalable inpainting algorithm.

[4] A translated pixel means a texture pixel copied from left and/or right texture map(s) to target view, where the copied location is indicated by the depth map(s). See [97] for details of DIBR.

First, we assume that all translated pixels in reference view $v+1$ are mapped using texture and depth maps at reference v. The remaining disoccluded pixels in the target region Ω need to be filled with the help of transmitted AI. Specifically, each block b with center on the target / source region boundary $\delta\Omega$ needs to be completed using one of three AI: $\varphi_b \in \{\texttt{skip}(\theta), \texttt{vec}, \texttt{intra}\}$. To select AI φ_b for block b, we write our objective of overall bit-rate as:

$$\min_{\{\varphi_b\}} \sum_b R(\varphi_b, \varphi_{b-1}) \tag{6.4}$$

where $R(\varphi_b, \varphi_{b-1})$ is the encoding rate for block b if AI φ_b and φ_{b-1} are used for blocks b and $b-1$, respectively. There is a dependency on the AI chosen for previous block $b-1$, because: i) SV can be differentially coded if the consecutive blocks use `vec` as AI, and ii) available intra-prediction types depend on whether previous block is also coded as `intra`, as described in Section 6.2.1.

It is subject to a distortion constraint for each block b:

$$D(\varphi_b) \leq \bar{d}, \quad \forall b \tag{6.5}$$

where $D(\varphi_b)$ is the resulting distortion of block b if mode φ_b is used, and an overall complexity constraint for all blocks b's:

$$\sum_b C(\varphi_b) \leq \bar{C} \tag{6.6}$$

where $C(\varphi_b)$ is the processing time of block b if mode φ_b is used.

Instead of solving (6.4) directly, we solve the equivalent Lagrangian instead:

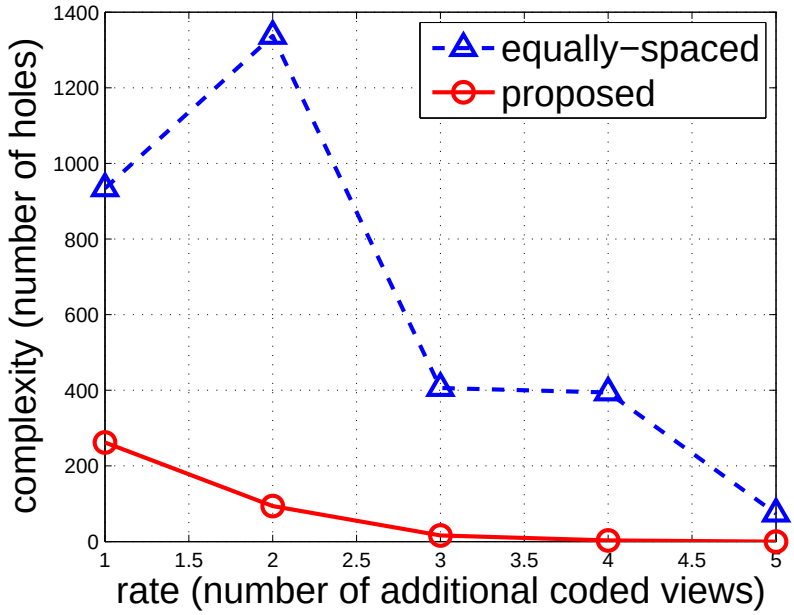
$$\min_{\varphi_b | D(\varphi_b) \leq \bar{d}} \sum_b R(\varphi_b, \varphi_{b-1}) + \lambda C(\varphi_b) \tag{6.7}$$

where λ is a Lagrangian multiplier, selected so that complexity constraint (6.6) is met for the entire frame. (6.7) can be solved by first constructing a trellis where each column b of three states correspond to the three types of AI that can be chosen for block b. The edge cost from state φ_{b-1} of column $b-1$ to state φ_b of column b is $R(\varphi_b, \varphi_{b-1}) + \lambda C(\varphi_b)$, except when $D(\varphi_b) > \bar{d}$, in which case the edge cost is infinity. Once the trellis is constructed, one can find find the shortest path in the trellis using the known Viterbi algorithm, which corresponds to the optimal set of AIs for all blocks b's.

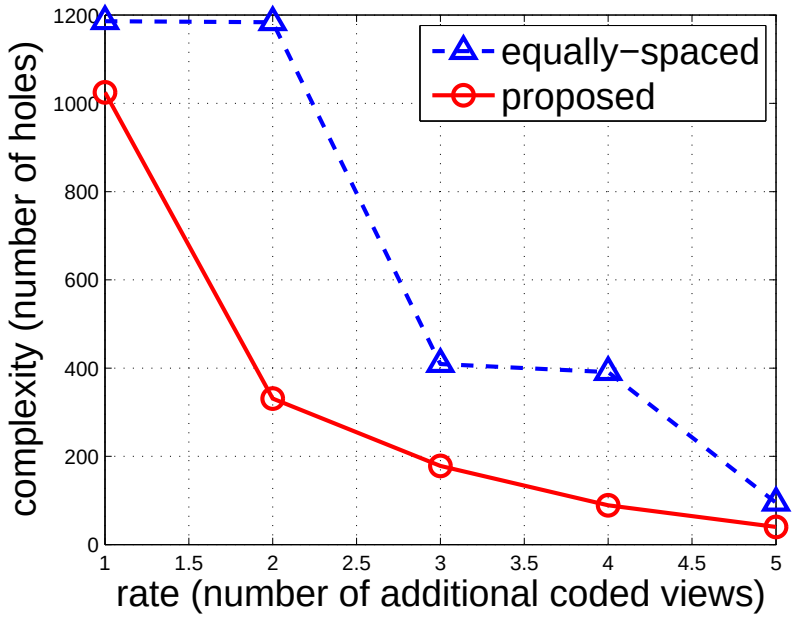
6.4 Experimentation

We now demonstrate the performance of our proposed RC optimizations for standard and new synthesis paradigms, respectively. The test sequences used are `Kendo` and `Balloons`[5].

[5] http://www.tanimoto.nuee.nagoya-u.ac.jp/ fukushima/mpegftv/



(a) Kendo



(b) Balloons

Figure 6.3: Tradeoff between the number of encoded virtual views and total number of holes after DIBR for standard synthesis.

For standard synthesis paradigm, we select a subset of virtual views between two reference views for rendering and encoding at server. A naïve method to select M intermediate virtual views is to pick views that are equally spaced, *i.e.*, insert M encoded views at locations $v + 1/(M + 1)$, $v + 2/(M + 1)$, etc. This method is denoted by "equally-spaced" in Fig. 6.3. Also shown is our "proposed" scheme. Note that M is a simple proxy for rate. One can alternatively consider actual rates of the M encoded views in the RC optimization, at the cost of a more complex optimization algorithm. This is left for future work.

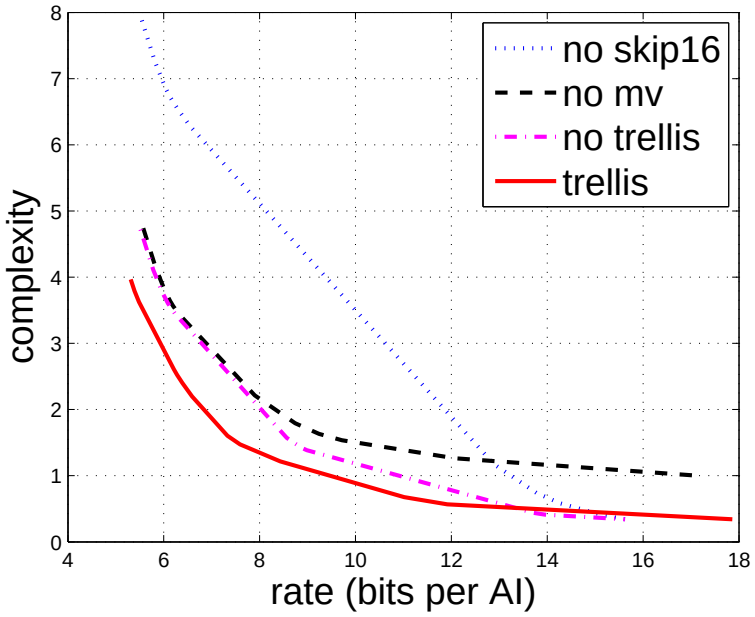
We see in Fig. 6.3 that for the same number of encoded views, "proposed" can achieve much lower complexity (measured in number of disocclusion holes). The reason can be explained as follows. In practice, the non-stationary geometrical information of the 3D scene means that disoccluded pixels are not evenly distributed across views, but rather skewed towards views with cameras that are closer to objects in the scene. Our proposed recursive optimization is robust to this non-stationarity and can smartly select the views with more disoccluded pixels, resulting in a dramatic decrease in client-side complexity.

In the second experiment for new synthesis paradigm, we verify the performance of our complexity-scalable DIBR-based view synthesis algorithm by smartly selecting AI to assist inpainting of the second reference view. In Fig. 6.4, trellis-based optimization ("trellis") can achieve noticeable gain over separate optimization that selects AI for each block individually ("no trellis"), by exploiting rate dependency between two consecutive AIs. Note that since the shortest path in trellis is searched at encoder, there is no client-side complexity increase when using "trellis" instead of "no trellis".

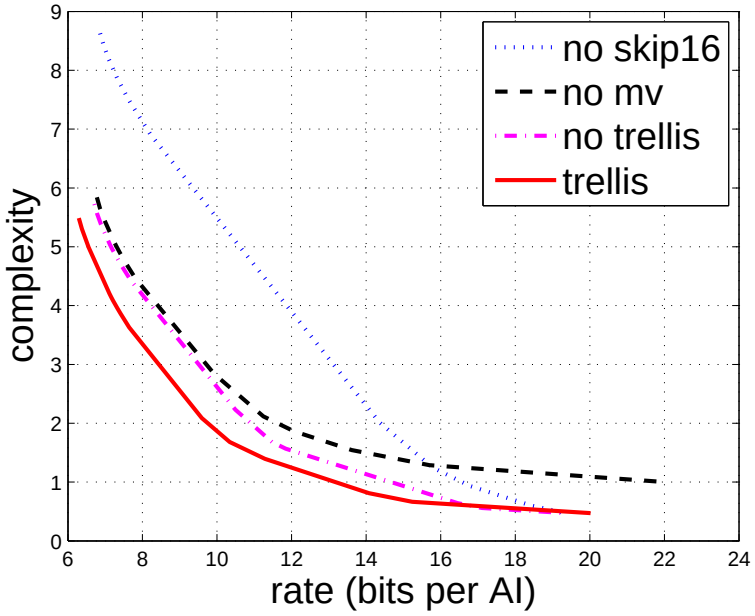
Fig. 6.4 also shows the performance of trellis-based optimization without `skip`(16)—only `skip`(32) and `skip`(64) are used for AI `skip`—and optimization without `vec` ("no mv"). Without `vec`, we can have comparable performance in low-bitrate region, but there is a larger performance gap in high-bitrate region, which implies that when complexity is more a concern, `vec` is an important AI to effectively reduce complexity. Without `skip`(16), the AI-aided method cannot perform well at low bitrate region, where the complexity is dramatically increased. This observation means that a well-defined search range is critical to scalably reduce the complexity at low bitrate.

6.5 Summary

DIBR-based view synthesis at client entails a sizable computation overhead, which may be too costly for computation-constrained devices. In this chapter, we propose two techniques to scalably reduce the complexity of view rendering at client, at the expense of a controlled increase of transmission rate from server. For standard view synthesis paradigm, we propose a dynamic programming algorithm to identify subset of virtual views for rendering and encoding at server, so that the client is only required to decode the encoded rendered images with no view synthesis overhead. For a new view synthesis paradigm, we propose to tune the selection of auxiliary information (AI) used to aid



(a) Kendo



(b) Balloons

Figure 6.4: RC tradeoff of view synthesis with the AI-aided hole-filling for new synthesis paradigm.

inpainting of large disoccluded regions, to optimally trade off transmission rate and inpainting complexity. Experimental results show that the proposed schemes can scalably and gracefully reduce client-side complexity, and the proposed optimizations achieve better rate-complexity tradeoff than competing schemes.

Chapter 7

Conclusions

In this thesis, we investigate the design of different MVC approaches for different multiview/3D video applications. We first review the previous works, and then various efficient coding techniques are proposed to achieve significant improvements over existing methods. In this chapter, the conclusions are drawn and the possible future directions are discussed.

7.1 Conclusions

The compression of texture-plus-depth video from multiple closely-spaced camera viewpoints is important for 3D imaging applications and new free viewpoint video communication. In the first part of this thesis, we propose an encoder-driven inpainting strategy to complete disocclusion holes in the DIBR-synthesized image in an RD optimal manner. Missing pixel regions that are difficult-to-inpaint are first completed following instructions from the encoder in the form of auxiliary information (AI). The remaining easy-to-fill holes are then completed without encoder's help via nonlocal template matching, which is effective due to the self-similarity characteristics in natural images. Finally, we propose two patch-based transform coding techniques (graph Fourier transform and DCT sparsification), so that only missing pixels in a target patch are encoded, avoiding representation redundancy. In doing so, our coding strategy successfully exploits the three kinds of redundancy inherent in the texture-plus-depth representation for coding gain: i) inter-view redundancy via DIBR-based 3D warping; ii) inter-pixel redundancy via patch-based transform coding; and iii) inter-patch redundancy via nonlocal template matching. Experimental results show noticeable coding gain over a comparable HEVC implementation.

In the second part of this thesis, two fast methods are proposed to generate the JND maps of multiview videos. In the JND synthesis method, for some reference views, the JND maps are obtained by exploiting the properties of luminance adaptation and texture masking of frames. The

JND maps of other views are synthesized by utilizing existing neighboring JND maps. In the JND copying method, the motion and disparity information yielded in JMVC encoder is utilized to predict JND block from reference JND maps. The error propagation of the method is studied and a JND block refreshing approach is proposed to alleviate the influence of the error propagation. A perceptual MVC scheme is developed based on the synthesized and predicted JND maps, where the residuals after intra or inter prediction are adjusted within the range of the corresponding JND thresholds. Experimental results show that the JND prediction method can be much faster than the JND synthesis method and has a comparable performance with the expensive direct JND method.

Thirdly, we presented an image tiling framework for representation of an object's dynamic 3D geometry, where the object's 3D mesh is first projected as 2D images to a set of carefully placed image tiles, before a multiview image codec tuned for piecewise smooth signals is deployed for coding of tile images. The key to a compact yet accurate representation is the selection and placement of image tiles. We showed that if only planar and cylindrical tiles are considered, the optimal tile placement problem (a finite tile set that best matches the object's representative 2D cross section) can be mapped to a piecewise linear approximation problem. The approximation problem can be subsequently solved efficiently using a dynamic programming algorithm. Experimental results show that optimal tile placements can outperform naïve tile placements by up to 35% in rate reduction.

In the fourth part, unlike previous work on IMVS that studied the tradeoff between expected streaming rate and storage cost when optimizing frame structures, we proposed to redesign frame structures with the help of available real-time computation, where frequently used view-switches are handled by pre-computed frames in storage, and infrequently used view-switches are handled by real-time computed frames. In contrast to multi-merge frames previously proposed that offer good tradeoff between transmission rate and storage cost, we proposed a uni-merge frame that is computed in real-time and offers good tradeoff between transmission rate and real-time computation cost. Experimental results show that with real-time computation, the expected streaming rate can be further decreased by 50% compared to pre-encoded structures without real-time computation.

DIBR-based view synthesis at client entails a sizable computation overhead, which may be too costly for computation-constrained devices. In the fifth part of this thesis, we propose two techniques to scalably reduce the complexity of view rendering at client, at the expense of a controlled increase of transmission rate from server. For standard view synthesis paradigm, we propose a dynamic programming algorithm to identify subset of virtual views for rendering and encoding at server, so that the client is only required to decode the encoded rendered images with no view synthesis overhead. For a new view synthesis paradigm, we propose to tune the selection of auxiliary information (AI) used to aid inpainting of large disoccluded regions, to optimally trade off transmission rate and inpainting complexity. Experimental results show that the proposed schemes can scalably and gracefully reduce client-side complexity, and the proposed optimizations achieve better rate-complexity tradeoff than competing schemes.

7.2 Future Work

7.2.1 Inpainting Assisted Multiview Video Coding

In the future work of the inpainting assisted multiview video coding as presented in Chapter 2, the performance can be further improved by incorporating with quadtree coding structure, because different sizes of coding block can adapt to local image characteristics. Secondly, since finding the optimal patch order in the proposed strategy is NP-hard, another heuristic to rank patches leading to better RD performance than the proposed one could exist.

7.2.2 Perceptual Multiview video Coding

Future work of the JND-based perceptual multiview video coding includes the design of more advanced JND-based residual filters to take full advantage of perceptual coding, the derivation of JND specific to stereoscopic and autostereoscopic display, and the application of subband-based JND model or structure-based perceptual assessment metrics [100] to MVC. There is also much room for improvement in the DIBR approach used in this thesis.

7.2.3 Geometry Compression

We are currently considering the following for future work. First, there is an implicit assumption in our current framework that the object's many cross sections along the principal axis are similar and well represented by the largest cross section. For objects where the cross sections vary drastically along the principal axis, one can divide the object into N parts before performing our tile optimization for each part, resulting in N times as many tiles. Second, after identifying the principal axis, one can identify a secondary axis, orthogonal to the principal axis, where the range of the projected 3D points onto the axis is maximized. We can then perform tile selection for a representative cross section orthogonal to the secondary axis. Third, it is possible to project the geometry of 3D object to alternative 2D manifolds, *e.g.*, a sphere, other than the proposed planar and cylindrical tiles. However, some pre-processing prior to depth coding may be required accordingly.

Bibliography

[1] JMVC (Joint Multiview Video Coding) software. version 8.0, 2009. [51]

[2] A. D. Abreu, P. Frossard, and F. Pereira. Optimized MVC prediction structures for interactive multiview video streaming. *IEEE Signal Processing Letters*, 20(6):604–606, 2013. [11]

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *9th ACM SIGCOMM Conference on Internet Measurement Conference*, Chicago, IL, November 2009. [12]

[4] M. Bertalmio, A. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–355. IEEE, 2001. [5, 20]

[5] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of SIGGRAPH*, pages 417–424, New Orleans, US, 2000. [5, 20, 23]

[6] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. [70]

[7] G. Bjontegaard. *Calculation of average PSNR differences between RD-curves*. VCEG-M33, Mar. 2001. [xi, xii, 36, 37, 38, 39, 61]

[8] A. Buades, B. Coll, and J. Morel. A non-local algorithm for image denoising. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, San Diego, CA, June 2005. [17]

[9] T.F. Chan and J. Shen. Nontexture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001. [20]

[10] Y. Chen, Y. Wang, K. Ugur, M. Hannuksela, J. Lainema, and M. Gabbouj. The emerging MVC standards for 3D video services. *EURASIP Journal on Advances in Signal Processing*, 2009:1–13, Jan. 2009. [9]

[11] J. Cheng, F. Ye, J. Di, C. Liu, and A. Men. Depth map compression via edge-based inpainting. In *Proceedings of Picture Coding Symposium*, Krakow, Poland, May 2012. [10]

[12] G. Cheung, J. Ishida, A. Kubota, and A. Ortega. Transform domain sparsification of depth maps using iterative quadratic programming. In *Proceedings of IEEE International Conference on Image Processing*, Brussels, Belgium, September 2011. [9, 18, 29]

[13] G. Cheung, A. Kubota, and A. Ortega. Sparse representation of depth maps for efficient transform coding. In *Proceedings of Picture Coding Symposium*, Nagoya, Japan, December 2010. [9, 18]

[14] G. Cheung and S. McCanne. A Framework for Computation-Memory Algorithmic Optimization for Signal Processing. *IEEE Transactions on Multimedia*, 5(2):174–185, Jun. 2003. [76]

[15] G. Cheung, A. Ortega, and N.-M. Cheung. Interactive streaming of stored multiview video using redundant frame structures. In *IEEE Transactions on Image Processing*, volume 20, no.3, pages 744–761, March 2011. [4, 11, 12, 75, 76, 78, 85]

[16] G. Cheung, A. Ortega, N.-M. Cheung, and B. Girod. On media data structures for interactive streaming in immersive applications. In *SPIE Visual Communications and Image Processing*, Huang Shan, China, July 2010. [11]

[17] G. Cheung, W. s. Kim, A. Ortega, J. Ishida, and A. Kubota. Depth map coding using graph based transform and transform domain sparsification. In *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, Hangzhou, China, October 2011. [9]

[18] N.-M. Cheung, A. Ortega, and G. Cheung. Distributed source coding techniques for interactive multiview video streaming. In *27th Picture Coding Symposium*, Chicago, IL, May 2009. [11, 75, 76, 77]

[19] C.-H. Chou and C.-W. Chen. A perceptually optimized 3-d subband codec for video communication over wireless channels. *IEEE Transactions on Circuits and Systems for Video Technology*, 6 (2):143 – 156, Apr. 1996. [8, 56]

[20] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174, 1998. [71]

[21] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. In *IEEE Transactions on Image Processing*, volume 13, no.9, pages 1–13, September 2004. [xi, 6, 7, 10, 17, 18, 20, 22, 23, 30, 84, 86]

[22] I. Daribo, G. Cheung, T. Maugey, and P. Frossard. RD optimized auxiliary information for inpainting-based view synthesis. In *Proceedings of 3DTV-Conference*, Zurich, Switzerland, October 2012. [10, 66]

[23] I. Daribo and B. Pesquet-Popescu. Depth-aided image inpainting for novel view synthesis. In *IEEE Multimedia Signal Processing Workshop*, Saint-Malo, France, October 2010. [88]

[24] I. Daubechies, R. Devore, M. Fornasier, and S. Gunturk. Iteratively re-weighted least squares minimization for sparse recovery. In *Commun. Pure Appl. Math*, 2009. [29]

[25] M. Deering. Geometry compression. In *Proceedings of the ACM conference on computer graphics and interactive techniques*, 1995. [4, 11]

[26] M. Droese, T. Fujii, and M. Tanimoto. Ray-space interpolation based on filtering in disparity domain. *Proceedings of 3D Image Conference*, pages 213–216, 2004. [9]

[27] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the IEEE Internation Conference on Computer Vision*, volume 2, pages 1033–1038, 1999. [6]

[28] D. Farin, Y. Morvan, and P. H. N. de With. View interpolation along a chain of weakly calibrated cameras. *Proceedings of IEEE Workshop Content Generation and Coding for 3D TV*, Eindhoven, Netherlands, Jun. 2006. [9]

[29] C. Fehn. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. In *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI*, pages 93–104, San Jose, US, 2004. [2, 5]

[30] M. Flierl and B. Girod. Multiview video compression. *IEEE Signal Processing Magazine*, 24 (6):66 – 76, Nov. 2007. [3, 49]

[31] M. Flierl, A. Mavlankar, and B. Girod. Motion and disparity compensated coding for multiview video. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 17, no.11, pages 1474–1484, November 2007. [17]

[32] T. Fujii and M. Tanimoto. Free viewpoint TV system based on ray-space representation. In *Proceedings of SPIE*, volume 4864, page 175, 2002. [1]

[33] Y. Gao, D. Chan, and J. Liang. JPEG XR optimization with graph-based soft decision quantization. In *Proc. 2011 IEEE International Conference on Image Processing (ICIP)*, pages 321–324, Brussels, Belgium, September 2011. [13]

[34] Y. Gao, G. Cheung, and J. Liang. Rate-complexity tradeoff for client-side free viewpoint image rendering. In *Proc. IEEE International Conference on Image Processing*, Melbourne, Australia, September 2013. [13]

[35] Y. Gao, G. Cheung, J. Liang, and A. Kaup. Optimizing frame structure with real-time computation for interactive multiview video streaming. In *Proceedings of 3DTV-Conference*, pages 1–4, Zurich, Switzerland, October 2012. [13]

[36] Y. Gao, G. Cheung, T. Maugey, P. Frossard, and J. Liang. Sender-driven inpainting for multiview video compression. *submitted to IEEE Transactions on Image Processing*, July 2014. [13]

[37] Y. Gao, G. Cheung, T. Maugey, P. Frossard, and J. Liang. 3D geometry representation using multiview coding of image tiles. In *Proc. 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, May 2043. [13]

[38] Y. Gao and J. Liang. Noncausal directional intra prediction: Theoretical analysis and simulation. In *Proc. 2012 IEEE International Conference on Image Processing (ICIP)*, pages 2917–2920, Orlando, FL, USA, September 2012. [13]

[39] Y. Gao, J. Wang, and J. Liang. System distortion exponents of two-way relay networks. In *Proc. 2011 IEEE Global Communications Conference (GLOBECOM)*, pages 1–5, Houston, TX, USA, December 2011. [13]

[40] Y. Gao, X. Xiu, J. Liang, and W. Lin. Perceptual multiview video coding using synthesized just noticeable distortion maps. In *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 2153–2156, 2011. [13]

[41] Y. Gao, X. Xiu, J. Liang, and W. Lin. Just noticeable distortion map prediction for perceptual multiview video coding. In *Proc. 2012 IEEE International Conference on Image Processing (ICIP)*, pages 1045–1048, Orlando, FL, USA, September 2012. [13]

[42] Y. Gao, X. Xiu, J. Liang, and W. Lin. Fast synthesized and predicted just noticeable distortion maps for perceptual multiview video coding. In *Journal of Visual Communication and Image Representation*, volume 24, no.6, pages 700–707, August 2013. [13]

[43] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1999. [31]

[44] S. Gokturk, H. Yalcin, and C. Bamji. A time-of-flight depth sensor—system description, issues and solutions. In *Proceedings of Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, Washington, DC, June 2004. [17]

[45] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *ACM SIGGRAPH*, New Orleans, LA, August 1996. [4]

[46] R. Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, 1999. [19]

[47] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. [49]

[48] Z. He, Y.K. Kim, and S.K. Mitra. Low-delay rate control for DCT video coding via rho-domain source modeling. In *IEEE Transactions on Circuits and Systems for Video Technology*, August 2002. [29]

[49] K. Hideaki, K. Masaki, K. Kazuto, and Y. Yoshiyuki. Multiview video coding using reference picture selection for free-viewpoint video communication. *Proceedings of Picture Coding Symposium*, pages 15–17, 2004. [9]

[50] W. Hu, G. Cheung, X. Li, and O. Au. Depth map compression using multi-resolution graph-based transform for depth-image-based rendering. In *IEEE International Conference on Image Processing*, Orlando, FL, September 2012. [9, 19, 26, 27, 66]

[51] J. Huang, F. Qian, A. Gerber, Z.-M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, June 2012. [12]

[52] H. S. Hussein, M. El-Khamy, and M. El-Sharkawy. Blind configuration of multi-view video coder prediction structure. *IEEE Transactions on Consumer Electronics*, 59(1):191–199, Feb. 2013. [9]

[53] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. In *Proceedings of the IEEE*, volume 81, no.10, pages 1385–1422, October 1993. [4, 7, 8, 49]

[54] Y. Jia, W. Lin, and A. Kassim. Estimating just-noticeable distortion for video. *IEEE Transactions on Circuits and Systems for Video Technology*, 16 (7):820 – 829, Jul. 2006. [49]

[55] M. Karczewicz and R. Kurceren. The SP- and SI-frames design for H.264/AVC. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 13, no.7, pages 637–644, July 2003. [78]

[56] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila. Depth map coding with distortion estimation of rendered view. In *Proceedings of SPIE Visual Information Processing and Communication*, San Jose, CA, January 2010. [9]

[57] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. *International Journal of Computer Vision*, 2:508–515, 2001. [9]

[58] M. Levoy and P. hanrahan. light field rendering. In *Proceedings of ACM SIGGRAPH*, New Orleans, LA, August 1996. [4]

[59] Y.-C. Li, I-.J. Liao, H.-P. Cheng, and W.-T. Lee. A cloud computing framework of free view point real-time monitor system working on mobile devices. In *Inter. Symp. Intelligent Sig. Procs. Comm. Sys.*, Chengdu, China, Dec. 2010. [75]

[60] D. Liu, X. Sun, and F. Wu. Intra prediction via edge-based inpainting. In *Proceedings of Data Compression Conference*, Snowbird, UT, March 2008. [10]

[61] D. Liu, X. Sun, F. Wu, S. Li, and Y.-Q. Zhang. Image compression with edge-based inpainting. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 17, no.10, pages 1273–1287, October 2007. [10]

[62] A.D. Liveris, Z. Xiong, and C.N. Georghiades. Compression of binary sources with side information at the decoder using ldpc codes. In *IEEE Communications Letters*, volume 6, no.10, pages 440–442, October 2002. [78]

[63] J.-G. Lou, H. Cai, and J. Li. A real-time interactive multi-view video system. In *Proceedings of ACM International Conference on Multimedia*, Singapore, November 2005. [11, 85]

[64] Y. Luo, Z. Zhang, and P. An. Stereo video coding based on frame estimation and interpolation. *IEEE Trans. Broadcast*, 49(1):14–21, Mar. 2003. [9]

[65] E. Martinian, A. Behrens, J. Xin, and A. Vetro. View synthesis for multiview video compression. *Proceedings of Picture Coding Symposium*, 2006. [9, 50]

[66] S. Masnou and J.M. Morel. Level lines based disocclusion. In *Proceedings of the IEEE International Conference on Image Processing*, Chicago, Illinois, 1998. [5]

[67] T. Maugey and P. Frossard. Interactive multiview video system with low decoding complexity. In *IEEE International Conference on Image Processing*, Brussels, Belgium, September 2011. [12]

[68] P. Merkle, K. Muller, A. Smolic, and T. Wiegand. Efficient compression of multi-view video exploiting inter-view dependencies based on H.264/MPEG4-AVC. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 1717 – 1720, 2006. [51]

[69] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand. Multi-view video plus depth representation and coding. In *IEEE International Conference on Image Processing*, San Antonio, TX, October 2007. [2, 9, 17]

[70] P. Merkle, A. Smolic, K. Muller, and T. Wiegand. Efficient prediction structures for multiview video coding. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 17, no.11, pages 1461–1473, November 2007. [9, 17, 75, 86]

[71] K. Mller and A. Vetro. Common Test Conditions of 3DV Core Experiments. In *JCT3V-G1100*, San Jose, US, 2014. [33, 35]

[72] Y. Morvan, D. Farin, and P. H. N. de With. Multiview depth-image compression using an extended H.264 encoder. In *Proceedings of Advanced Concepts for Intelligent Vision Systems, Lecture Notes in Computer Sciences*, volume 4678, pages 675–686, 2007. [9]

[73] K. Muller, P. Merkle, H. Schwarz, T. Hinz, A. Smolic, T. Oelbaum, and T. Wiegand. Multiview video coding based on H.264/AVC using hierarchical B-frames. *Proceedings of Picture Coding Symposium*, Beijing, China, Apr. 2006. [9]

[74] K. Muller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F. H. Rhee, G. Tech, M. Winken, and T. Wiegand. 3D high-effciency video coding for multi-view video and depth data. *IEEE Transactions on Image Processing*, 22(9):3366–3378, Sept. 2013. [9]

[75] S. Nakagawa, T. Yamasaki, and K. Aizawa. Deformation-based data reduction of time-varying meshes for displaying on mobile terminals. In *3DTV-Conference*, Tampere, Finland, June 2010. [11, 66]

[76] J.M. Ogden, E.H. Adelson, J.R. Bergen, and P.J. Burt. Pyramid-based computer graphics. In *RCA Engineer*, volume 30, no.5, pages 4–15, 1985. [5]

[77] Kedar A Patwardhan, Guillermo Sapiro, and Marcelo Bertalmío. Video inpainting under constrained camera motion. *IEEE Transactions on Image Processing*, 16(2):545–553, 2007. [20]

[78] X. Qi, T. Zhang, F. Ye, A. Men, and B. Yang. Intra prediction with enhanced inpainting method and vector predictor for HEVC. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Kyoto, Japan, March 2012. [10]

[79] K. Ramchandran, A. Ortega, and M. Vetterli. Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders. In *IEEE Transactions on Image Processing*, volume 3, no.5, September 1994. [31]

[80] S. Seitz and C. Dyer. view morphing. In *ACM SIGGRAPH*, New Orleans, LA, August 1996. [4]

[81] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proceedings of ACM SIGGRAPH*, New York, NY, September 1998. [86]

[82] G. Shen, W.-S. Kim, S.K. Narang, A. Ortega, J. Lee, and H. Wey. Edge-adaptive transforms for efficient depth map coding. In *Proceedings of Picture Coding Symposium*, Nagoya, Japan, December 2010. [9, 18, 19, 27, 66]

[83] G. Shen, W.-S. Kim, A. Ortega, J. Lee, and H. Wey. Edge-aware intra prediction for depth-map coding. In *IEEE International Conference on Image Processing*, Hong Kong, September 2010. [26]

[84] S. Shimizu, M. Kitahara, H. Kimata, K. Kamikura, and Y. Yashima. View scalable multiview coding using 3-D warping with depth map. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 17, no.11, pages 1485–1495, November 2007. [17]

[85] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *Proceedings of ACM SIGGRAPH*, Los Angeles, LA, August 1999. [4]

[86] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. In *IEEE Signal Processing Magazine*, volume 30, no.3, pages 83–98, May 2013. [27]

[87] T. Sikora and B. Makai. Shape-adaptive DCT for generic coding of video. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 5, no.1, February 1995. [25]

[88] V. Srinivasan and G. Varghese. Fast address lookups using controlled prefix expansion. In *ACM Transactions on Computer Systems*, volume 17, no.1, pages 1–40, February 1999. [76]

[89] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (HEVC) standard. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 22, no.12, December 2012. [3, 17]

[90] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (HEVC) standard. In *IEEE Transactions on Circuits and Systems for Video Technology*, December 2012. [24]

[91] G.J. Sullivan, J.M. Boyce, Y. Chen, J.-R. Ohm, C.A. Segall, and A. Vetro. Standardized extensions of high efficient video coding (HEVC). In *IEEE Journal of Selected Topics in Signal processing*, pages 1001–1016, December 2013. [9, 33]

[92] J. Sun, H.-Y. Shum, and N.-N. Zheng. Stereo matching using belief propagation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 25, no.7, pages 787–800, July 2003. [17]

[93] L. Sun, J. Wen, F. Zhang, W. Hu, W. Feng, and S. Yang. A framework for heuristic scheduling for parallel processing on multicore architecture: A case study with multiview video coding. In *IEEE Trans. Circuits Systems Video Technology*, volume 19, No. 11, pages 1658–1666, Nov. 2009. [75]

[94] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo. Free-viewpoint TV. In *IEEE Signal Processing Magazine*, volume 28, no.1, January 2011. [17, 74, 83, 84]

[95] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *Proceedings of the ACM conference on Computer graphics and interactive techniques*, pages 123–132, 1998. [11]

[96] G. Taubin and J. Rossingnac. Geometric compression through topological surgery. In *ACM Transactions on Graphics*, volume 17, pages 84–115, 1998. [11]

[97] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila. View synthesis techniques for 3D video. In *Proceedings of the SPIE, Applications of Digital Image Processing XXXII*, volume 7443 (2009), pages 74430T–74430T–11, 2009. [9, 17, 83, 84, 87, 88]

[98] I. Tosic and P. Frossard. Omnidirectional views selection for scene representation. In *IEEE International Conference on Image Processing*, Atlanta, GA, October 2006. [11]

[99] I. Tosic, P. Frossard, and P. Vandergheynst. Progressive coding of 3-D objects based on overcomplete decomposition. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 16, no.11, pages 1338–1349, November 2006. [11]

[100] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. In *IEEE Transactions on Image Processing*, volume 13, no.4, pages 600–612, August 2005. [96]

[101] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 13, no.7, pages 560–576, July 2003. [3, 10, 17, 66]

[102] Y. Wu, C. Wu, B. Li, X. Qiu, and F. Lau. CloudMedia: When cloud on demand meets video on demand. In *Inter. Conf. Distributed Computing Systems*, pages 268–277, Minnesota, USA, Jun. 2011. [75]

[103] Z. Xiong, X. Sun, and F. Wu. Block-based image compression with parameter-assistant inpainting. In *IEEE Transactions on Image Processing*, volume 19, no.6, pages 1651–1657, June 2010. [10]

[104] X. Xiu, G. Cheung, and J. Liang. Delay-cognizant interactive multiview video with free viewpoint synthesis. In *IEEE Transactions on Multimedia*, volume 14, no.4, pages 1109–1126, August 2012. [11, 12, 75, 85]

[105] X. Xiu, D. Pang, and J. Liang. Rectification-based view interpolation and extrapolation for multiview video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 21 (6):693–707, Mar. 2011. [9]

[106] Y. Xu and H. Xiong. Advanced inpainting-based macroblock prediction with regularized structure propagation in video compression. In *Proceedings of Picture Coding Symposium*, Nagoya, Japan, December 2010. [10]

[107] 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 Transactions on Circuits and Systems for Video Technology*, 17(11):1436–1449, Nov. 2007. [9]

[108] T. Yamasaki and K. Aizawa. Bit allocation of vertices and colors for patch-based coding in time-varying meshes. In *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010. [11, 66]

[109] T. Yamasaki and K. Aizawa. Patch-based compression for time-varying meshes. In *IEEE International Conference on Image Processing*, Hong Kong, September 2010. [11, 66]

[110] X. Yang, W. Lin, Z. Lu, E. Ong, and S. Yao. Motion-compensated residue preprocessing in video coding based on just-noticeable-distortion profile. *IEEE Transactions on Circuits and Systems for Video Technology*, 15 (6):742 – 752, Jun. 2005. [7, 8, 56]

[111] S. Yea and A. Vetro. View synthesis prediction for multiview video codings. In *Signal Processing: Image Communication*, volume 24, no.1-2, pages 89–100, January 2009. [9]

[112] C. Zhang, Z. Yin, and D. Florencio. Improving depth perception with motion parallax and its application in teleconferencing. In *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, Rio de Janeiro, Brazil, October 2009. [83]

[113] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *Proceedings of ACM SIGGRAPH*, pages 600–608, 2004. [57, 59]