3D Multi-view Imaging: Object Contour Approximation For Depth Image Coding and Multi-view Image/Video Streaming

by

Yuan Yuan

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

Doctor of Philosophy

in

Signal and Image Processing

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Thanks to the rapidly dropping cost of digital cameras, multi-view imaging—using a series of cameras capturing images from the same 3D scene simultaneously but from different viewpoints—opens a wide variety of interesting research topics and applications. Among them, free viewpoint TV and light field camera are two of the most important applications, which enable users to observe a static 3D scene by freely changing their viewpoints. However, the amount of multi-view data that needs to be stored or transmitted is huge. Therefore, efficient image/video coding and streaming are crucial points for the success of such applications.

We first investigate object contours in depth image coding. A depth image provides partial geometric information of the captured 3D scene, which is important for synthesizing images corresponding to different virtual camera viewpoints via depth-image-based-rendering (DIBR). It has been shown that lossy compression of object contours will lead to bleeding artifacts in DIBR synthesized view, while losslessly coding of the exact object contours can be expensive at low rate. In this thesis, we propose to approximate object contours to save coding bits. Specifically, we first greedily approximate object contours based on an arithmetic edge coding (AEC) model to lower the edge coding cost. To control the induced synthesized view distortion due to contour approximation, we then introduce a rate-distortion (RD) optimal scheme. We show that the object contours themselves can be suitably approximated to save coding bits, while the synthesized objects remain sharp and natural for human perception.

We then study the problem of multi-view image/video streaming. In free viewpoint TV, a user can pull color and depth videos captured from two nearby reference viewpoints to synthesize his chosen intermediate virtual view for observation via DIBR. A user may pull the same reference views from the server with other users so that the streaming cost can be shared, while a reference view with further distance to his chosen virtual view may increase the synthesized view distortion. In this thesis, we divide users into groups, where a user simultaneously belongs to two groups and each group shares the streaming cost of a single reference view. We also aim to find a Nash Equilibrium (NE) solution of reference view selection for each user, so that the shared streaming cost and the synthesized view distortion are optimally traded off. Specifically, we first derive a lemma based on known property of synthesized view distortion functions. We then design a search algorithm to find a NE solution, leveraging on the derived lemma to reduce search complexity.

Interactively streaming light field multi-view images is another focus of this thesis. Interactive light field streaming (ILFS) means that a user periodically requests a viewpoint for observation, and in response the server transmits a pre-synthesized and encoded viewpoint image to the user. The challenge is how to design and pre-encode a storage-constraint frame structure to enable efficient view navigation. In this thesis, using a Lloyd's algorithm variant, we recursively insert into a frame structure a set of "landmarks" at locally optimal locations to improve ILFS performance, so as to trade off frame storage cost and the expected transmission cost. Experimental results show that our proposed structure has noticeably lower expected transmission cost for the same storage than other previous methods.

Preface

This thesis is an original work conducted by Yuan Yuan. It includes materials and results from the following publications:

[1] **Y. Yuan**, G. Cheung, P. Frossard, P. L. Callet and V. H. Zhao, "Contour Approximation & Depth Image Coding for Virtual View Synthesis", *IEEE International Workshop on Multimedia Signal Processing*, 2015.

[2] **Y. Yuan**, G. Cheung, P. L. Callet, P. Frossard, and V. H. Zhao, "Object Shape Approximation & Contour Adaptive Depth Image Coding for Virtual View Synthesis", accepted to *IEEE Transactions on Circuits and Systems for Video Technology*, August, 2017.

[3] **Y. Yuan**, B. Hu, G. Cheung and V. H. Zhao, "Optimizing Peer Grouping for Live Free Viewpoint Video Streaming", *IEEE International Conference on Image Processing*, 2013.

[4] **Y. Yuan**, G. Cheung and P. Frossard, "Optimizing Landmark Insertions for Interactive Light Field Streaming", accepted to *IEEE International Conference on Image Processing*, 2017.

Chapter 3 includes materials from Ref. [1] and Ref. [2]. Chapter 4 is based on Ref. [2]. In these two works, I was responsible for idea development, data collection and analysis as well as the manuscript composition. Professor G. Cheung was involved with the concept formation and manuscript composition. Professors P. L. Callet and P. Frossard contributed to manuscripts editing. Professor V. H. Zhao was the supervisory author.

Chapter 5 includes materials from Ref. [3]. I was responsible for experiment implementation and results analysis. B. Hu helped develop the concept. Professors G. Cheung and V. H. Zhao acted as supervisors.

Chapter 6 is based on Ref. [4]. In this work I was responsible for the idea development, data analysis as well as the manuscript composition. Professors G. Cheung and P. Frossard provided supervision.

The literature survey in Chapters 1 and 2, the concluding marks in Chapter 7 are my original work.

Acknowledgments

I would like to acknowledge my deepest thanks to my supervisor Dr. Vicky H. Zhao, for providing me this valuable opportunity to work with her. Her continuous support and patient guidance helped me a lot during my Ph.D program. I am also sincerely grateful to my co-supervisor, Dr. Hai Jiang, for what he did to help me complete my candidacy exam and this thesis when Dr. Vicky H. Zhao is on leave.

I would like to express my most grateful thanks to Dr. Gene Cheung, who supervised me when I was an intern student at National Institute of Informatics, Tokyo, Japan. My Ph.D program would not be completed without his patient guidance and constant encouragement. His devotion and expertise in research motivated me. His precision, his rigorous attitude, and desire for excellence not only inspired me in my academic, but also set up an role model for my professional and personal development with lifetime.

Many thanks to my colleagues and friends: Xiaohui Gong, Jing Yan, Chunyan Wu, Lu Qian, Bo Hu, Wei Hu, Jing Zeng, Jiahao Pang, Amin Zheng, Wei Dai, for their friendship, help and inspiration. I feel so lucky to be friends with them, and am also deeply grateful to everything I have learned from them. The friendship with them leaves me many pleasant and precious memories during my Ph.D study.

Most importantly, I would like to express my special thanks to Qintian Guo, who raised me up in all the frustrations during my program. Thanks to his support and accompany in the past years.

Last but not the least, I give my heartfelt gratitude to my family including my grandparents, parents, sister and all the relatives. Without their support and encouragement, I could never accomplish so much in my life.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Acronyms	Definition
3DSwIM	3D Synthesized View Image Quality Metric
3DTV	Three-dimensional TV
AC	Alternating Components
AEC	Arithmetic Edge Coding
AR	Augmented Reality
BD	Bjontegaard Delta
CDF	Cumulative Distribution Function
DCC	Differential Chain Code
DCT	Discrete Cosine Transform
DIBR	Depth Image Based Rendering
DP	Dynamic Programming
DSC	Distributed Source Coding
FTV	Free Viewpoint TV
FVV	Free Viewpoint Video
GFT	Graph Fourier Transform
HEVC	Hight Efficiency Video Coding
ILFS	Interactive Light Field Streaming
LF	Light Field
MMSE	Minimum Mean Square Error
MPEG	Motion Picture Experts Group
MSE	Mean Square Error
NE	Nash Equilibrium

P2P	Peer-to-Peer
PG	PSNR Gain
PSNR	Peak Signal-to-Noise Ratio
PWC	Piecewise Constant
PWS	Piecewise Smooth
QP	Quantization Parameter
RD	Rate-Distortion
RG	Rate Gain
RS	Rate-Score
SI	Side Information
TSVQ	Tree-Structure Vector Quantizer
VC	Video Conferencing
VR	Virtual Reality

List of Symbols

For Chapter 3:

Symbol	Definition
$\mathcal{A} = \{\mathtt{l}, \mathtt{s}, \mathtt{r}\}$	the three *relative* directions left, straight, and right
$\mathcal{A}^o = \{\rightarrow, \downarrow, \leftarrow, \uparrow\}$	the four *absolute* directions for the first edge
$d_{\mathbf{x}}(i, j)$	the MSE when constant pieces from i to j are approximated
$D_{\mathbf{x}}(i, k)$	the MMSE from i-th constant piece to the end of $\mathbf{x}$, given k more jumps to be eliminated
e_i	the i-th edge in $\mathcal{E}$
e_i^o	the i-th edge in $\mathcal{E}^o$
$\mathcal{E}^o$	the set of original contours in a 2D depth image
$\mathcal{E}$	the set of approximated contours in a 2D depth image
$H(\phi)$	the entropy of state random variable ϕ
$I_0(\cdot)$	the modified Bessel function of order 0
K	the size of previous edge window for AEC model
l	the best-fitting line for AEC
M	the number of jumps in approximated 1D PWC signal
N	the number of jumps in original 1D PWC signal
$Pr(\cdot)$	the probability
u_l	the direction of l
v_p	the predicted direction by AEC
$\theta \in \mathcal{A}$	the possible direction of edge e_{t+1}
γ_θ	the angle between θ and u_l

ϵ_θ	the minimum distance between the end point of θ and l

For Chapter 4:

Symbol	**Definition**
$\mathbf{c}_s$, $\mathbf{c}_o$	the AC coefficient matrices for synthesized, reference blocks
$d_{p^o}(q^o, q)$	the row distortion induced by shifting a vertical edge from (p^o, q^o) to (p^o, q)
D_b	the block distortion
D_b^i	the row distortion
$\hat{D}_b$	our distortion proxy of D_b
$D(\mathcal{E}, \mathcal{E}^o)$	the distortion for approximating $\mathcal{E}^o$ with $\mathcal{E}$
e_i^o, e_i	the i-th edge in $\mathcal{E}^o$, $\mathcal{E}$
$\mathcal{E}^o$, $\mathcal{E}$	the original and approximated segments
$\mathbf{f}_s^i$, $\mathbf{f}_o^i$	the CDFs for each row i for the synthesized and best-matched blocks
$f_{\sigma_o}(c)$, $f_{\sigma_s}(c)$	the Laplace distribution of AC coefficients
$\mathbf{F}_s$, $\mathbf{F}_o$	the CDF of the histograms $\mathbf{H}_s$, $\mathbf{H}_o$
$F_{\sigma_o}(c)$, $F_{\sigma_s}(c)$	the CDF of $f_{\sigma_o}(c)$ and $f_{\sigma_s}(c)$
$\langle f_{\sigma_o}, f_{\sigma_s} \rangle$	our computed row distortion
$g(c)$	the absolute difference between $F_{\sigma_o}(c)$ and $F_{\sigma_s}(c)$
$g_{\texttt{max}}$	the maximum value of $g(c)$
$\mathbf{H}_s$, $\mathbf{H}_o$	the histogram for each AC coefficient matrix $\mathbf{c}_s$, $\mathbf{c}_o$
$J(\mathbf{s}_t, p_t, q_t)$	the recursive function to find best $\mathcal{E}$
L	the number of bins for histogram
$N \times N$	the block size
(p_i^o, q_i)	the 2D coordinates of a vertical edge e_i's starting point in $\mathcal{E}$

(p_i^o, q_i^o)	the starting point of the original vertical edge in $\mathcal{E}^o$ crossing the p_i^o-th pixel row
QP_C	the quantization parameter for color images
$r(e_t\|\mathbf{s}_t)$	the estimated coding bits of edge e_t
$R(\mathcal{E})$	the coding rate of segment $\mathcal{E}$
T	the length of edges in a segment
V	the number of vertical edges in a segment
W	the searching window size
σ_s, σ_o	the coefficients distribution parameter

For Chapter 5:

Symbol	**Definition**
$d_u(v^l, v^r)$	the synthesized virtual view distortion
N	the number of users in a network
q_u	the peers distribution function
$s(v)$	the subscription fee for users requesting view v
u	a virtual view position
v^l, v^r	the left and right reference view
$\mathcal{V} = \{1, \ldots, V\}$	a set of captured reference views
$\mathcal{V}_u$	the selected reference view set for u
ε	the minimum reference view distance

For Chapter 6:

Symbol	**Definition**
$b(\theta)$	the storage cost of a structure θ
$c_i^{(t)}(l)$	the expected transmission cost
$g(t)$	the probability that there could be at least t view-switches in a session

$h_i^{(t)}()$, $\dot{h}_i^{(t)}()$, $\ddot{h}_i^{(t)}()$	the costs of 0-hop, 1-hop and 2-hop transmission
I_j	I-frame of view j
l_k	the landmark in partition Ψ_k
M_i	the merge frame (M-frame) of view i
N	a LF array of N views
$\mathcal{N}(i)$	the neighboring views that can be switched to, starting from view i
$p_{j\|i}$	the view switching probability
$P_i(j)$	a P-frame of view i, using I-frame I_j as predictor
q_i	the probability that a view i would be visited
r_j^I	transmission cost of transmitting an I-frame I_j
$r_j^P(i)$	transmission cost of transmitting a P-frame $P_j(i)$ plus an M-frame M_j
T	the lifetime of an ILFS session
T_o	the maximum lifetime of an ILFS session
$\delta(\cdot)$	the cost dealing with two partition's boundaries
$\psi(\cdot)$	the cost within a partition
Ψ_k	a partition of a 2D LF array
θ	a given frame structure
θ^*	the optimal frame structure

Chapter 1

Introduction

Thanks to the rapidly dropping cost of digital cameras, multi-view imaging—using a series of cameras capturing images of the same scene simultaneously but from different viewpoints—has attracted increasing attention recently. It opens a wide variety of interesting research topics and applications, such as virtual view synthesis, image/video segmentation, object tracking/recognition, industrial inspection, and virtual reality. While conventional single view images/videos are enough to handle some of these tasks, the availability of multiple views of a 3D scene can significantly broaden the field of applications and enhance user experience.

Three-dimensional TV (3DTV) [6] and free viewpoint television (FTV) [7] are two of the most important applications of multi-view imaging system, which expand user experience beyond what is offered by traditional media. Unlike traditional TV in which users are provided with only a single view of a 3D world and cannot control their viewpoint, 3DTV has more viewpoints and offers a 3D depth impression of the observed scene, which is closer to what we experience in the real world. Improving the drawback of 3DTV that the number of viewpoints is fixed, FTV enables users to view a 3D scene by freely changing their viewpoints, which could greatly contribute to improve immersive communication. The entire processing chain of FTV in real world applications is shown in Fig. 1.1, including multi-view image capturing, 3D scene representation, images coding and transmission, virtual view image rendering and 3D display.

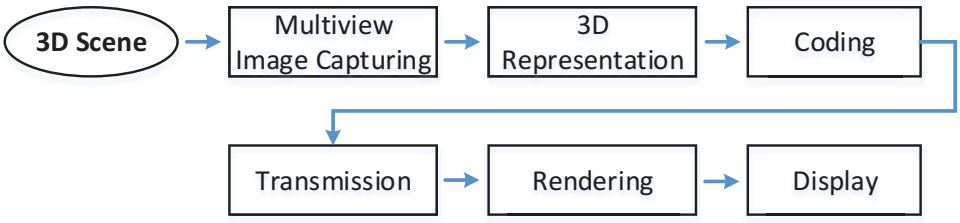


Fig. 1.1. Processing chain of a 3D multi-view images.

Another application of multi-view imaging is light filed (LF) images [8],which can be captured by a LF camera such as Raytrix[1] or Lytro[2]. Contrary to a conventional analog or digital camera that only records light intensity of a scene, a light field camera can capture not only the light intensity information of a scene but also the direction that the light rays are traveling in space. By inserting a 2D array of microlens in front of a regular, commercially available image sensor [3][8], a LF camera can caputer a static 3D scene with multiple images from different viewpoints. The capability of LF images that record the intensity and direction of incident light rays brings a lot of new possibilities for LF image processing and applications, such as refocusing [3], reducing glare [9], matting [10] and depth image estimation [11][12].

In the following, we first briefly introduce the processing chain of FTV and the LF images capturing system. Based on them, we then discuss the motivations and contributions of this thesis. The thesis outline will be shown at the last.

1.1 Background

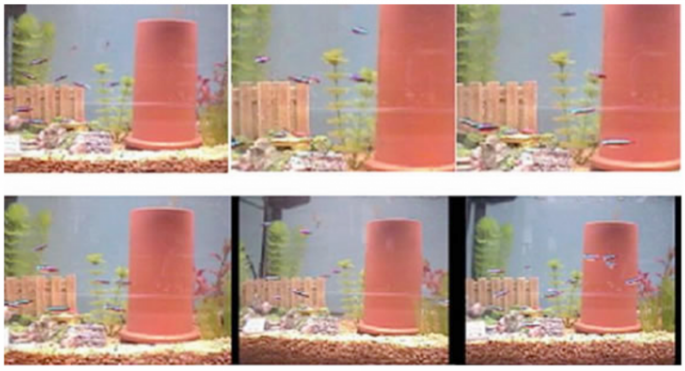
1.1.1 Multi-View Image Processing Chain of FTV

We first introduce how to capture the multi-view images in FTV. An example of FTV capturing system is shown in Fig. 1.2(a), where a series of digital cameras are set parallelly around a 3D scene, such that all the cameras capture the same scene simultaneously but from different viewpoints. This multi-view image capturing system allows users to freely control viewpoint of a real dynamic 3D scene in real-

[1]http://www.raytrix.de
[2]https://illum.lytro.com/illum

time and enables users to choose whichever viewpoint they want. This developed multi-view capturing system in Fig. 1.2(a) consists of one host-server PC and 100 client PCs, each equipped with a high-definition camera. This system is capable of capturing 100 synchronized video signals and the camera positions can be easily changed. Fig. 1.2(b) shows examples of generated free viewpoint images at various time instants and viewpoints. Complicated natural scenes including complex objects such as small moving fishes, bubbles from different viewpoint are reproduced with high quality. The different shape and height of the pillar can show the difference of viewpoints.

(a)



(b)

Fig. 1.2. A multi-view image capturing system and generated free viewpoint images in FTV system [1]

To well represent a 3D scene, the depth of field information—distance from the surfaces of scene objects to a viewpoint—plays a very important role. The depth information can be provided through different methods. It can be estimated from a multi-camera setup using stereo correspondence algorithms [13] or be directly

recorded by special time-of-flight cameras [14]. The advent of depth sensing technologies like Microsoft Kinect[3] means that one can now easily acquire both color images and depth images of a 3D scene at the same time. While a color image provides the texture of a 3D scene, a depth image provides the geometric information, namely the shape of physical objects captured from a particular viewpoint. Hence, the *color-plus-depth* format, which consists of a pair of color image and the corresponding per-pixel depth image from the same viewpoint, has been widely used in 3D scene representation. As illustrated in Fig. 1.3, there are two color-plus-depth image pairs from left ((a) and (b)) and right ((d) and (e)) viewpoints, respectively, where (a) and (d) are two color images, (b) and (e) are their corresponding depth images from the same viewpoint for `teddy`[4].

When multiple images/videos are recorded simultaneously and the 3D scene is represented by both texture and depth information, the amount of data that needs to be transmitted or stored is huge. Therefore, efficient compression is a key condition for the success of such applications. Although classical 2D image/video coding technique is very mature [15] [16], the correlation between color and depth images and the specific characteristic of depth images lead to a lot of room for improvement and optimization. Hence, 3D multimedia data compression has received a lot of attention in research and development recently, especially for depth image compression [17].

In general, the denser capturing of multi-view images with a larger number of cameras provides a more precise 3D representation, resulting in higher quality views. However, when the number of views is pretty large, the cost in bit-rate to encode these views may be expensive. Moreover, even a large number of video views are transmitted, it is still not capable to capture any arbitrary viewpoint. To overcome these disadvantages, instead of capturing a large number of camera views, color and depth image pairs from sparse camera arrangements are utilized to capture 3D scenes, while intermediate virtual views could be rendered. This kind of depth

[3] http://www.xbox.com/en-CA/xbox-one/accessories/kinect

[4] http://en.wikipedia.org/wiki/Von Mises distribution

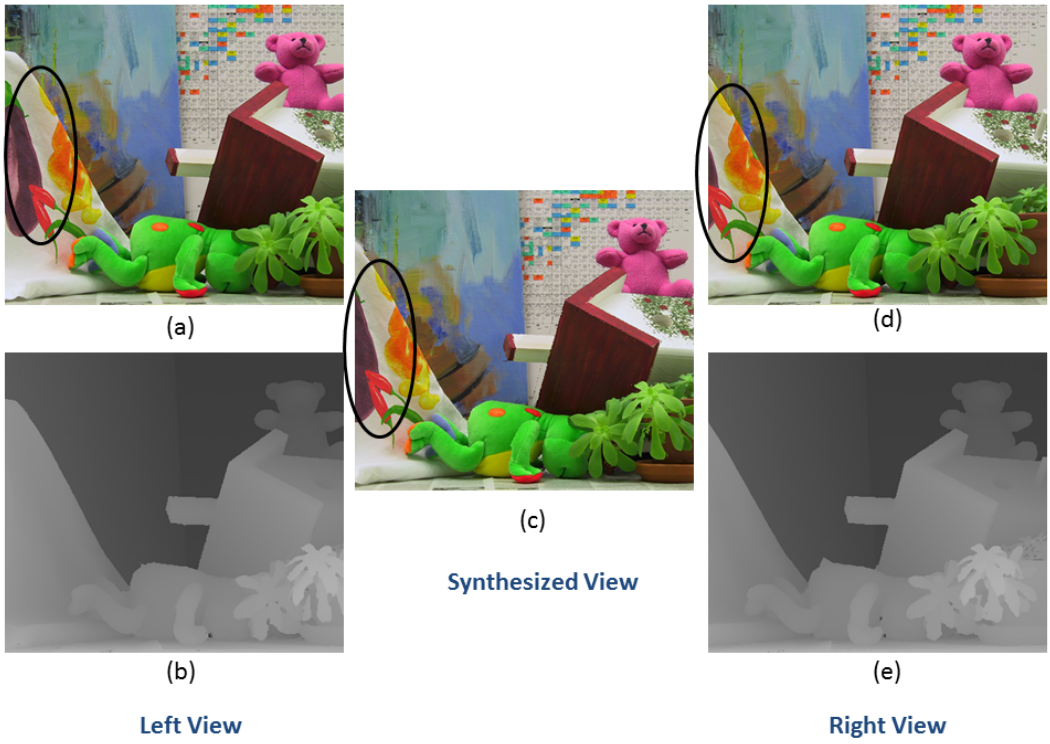


Fig. 1.3. Color and depth image pairs from left viewpoint ((a) and (b)) and right viewpoint ((d) and (e)). Image (c) is the corresponding synthesized view from the virtual middle viewpoint. The circled eggplant shows the differences of capturing viewpoints for the same 3D scene.

related virtual view synthesis has been explored, for example in Motion Picture Experts Group (MPEG). The work in [2] proposed a *depth-image-based-rendering* (DIBR) method to generate intermediate virtual views, where color and depth image pair(s) captured from cameras are taken as input. DIBR essentially maps color pixels in the camera views to appropriate pixel locations in a virtual view; such locations are derived from the corresponding depth pixels in the reference views. Fig. 1.4 shows an example of sparse cameras' multi-view color-plus-depth system, in which only three views (V_1, V_5, V_9) plus their corresponding depth (D_1, D_5, D_9) are transmitted to the decoder. Additional intermediate views V_2—V_4 and V_6—V_8 then can be rendered using DIBR techniques after the three color-plus-depth image pairs are reconstructed. As in Fig. 1.3, with the left and right reference color and

depth image pairs, a synthesized virtual view by DIBR is shown in Fig. 1.3(c), which is corresponding to a virtual camera viewpoint in the middle of the two original viewpoints. The major differences among these three different viewpoints are highlighted in Fig. 1.3 by circles.

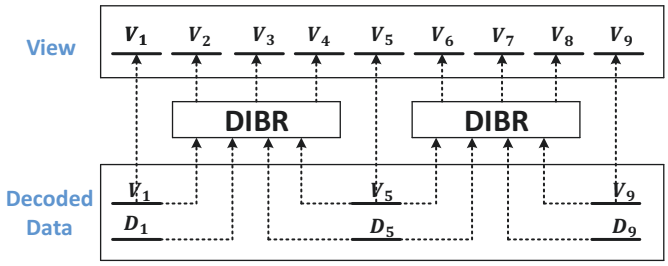


Fig. 1.4. Example of multi-view color-plus-depth system (9 output views out of 3 input views plus depth) [2]

1.1.2 Light Field Multi-View Imaging

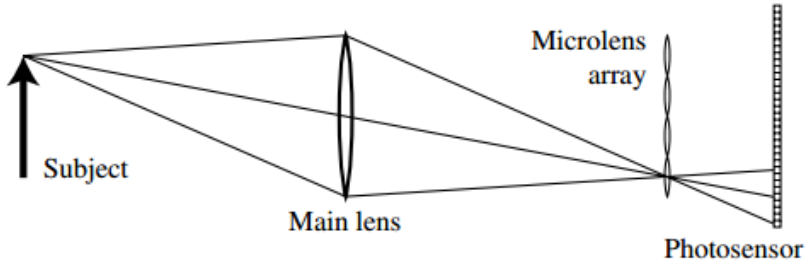


Fig. 1.5. A conceptual schematic of LF camera [3].

A light field is a 2D array of closely spaced images of a static 3D scene. We now briefly introduce the capturing system of LF images. LF acquisition process is based on capturing not only the light intensity of a 3D scene but also the direction information of light rays. One type of LF camera inserts a 2D microlens array in front of a conventional 2D photo sensor [3][8]. A conceptual schematic of the LF camera proposed in [3] is shown in Fig. 1.5, where a microlens array is inserted between the main lens and the photosensor. Rays of lights from a single point on the subject first translate to the main lens along its optical axis, to focus on a subject of interest, which is exactly the same as in a conventional camera. Then, from the main lens, rays of light are brought to a single convergence point on the focal plane

of the microlens array. Based on the direction of these rays of light, the microlens separates rays of light to create a focused image of the aperture of the main lens on the photo sensor. Since the 2D array of microlens split up light rays passing through the main lens into multiple individual light ray, each microlens forms a tiny sharp image to measure the directional distribution of light at that microlens. The resulting raw image is then a composition of as many tiny images as there are microlens.

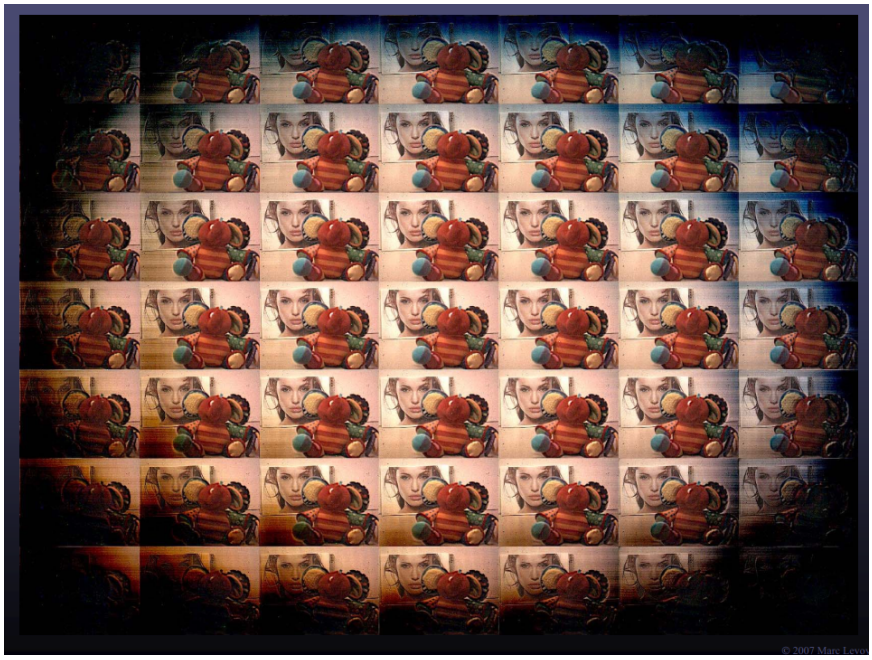


Fig. 1.6. An example of captured multi-view LF images [4].

The acquired data by a LF camera is characterized by considering the two-plane light filed L, where $L(u, v, s, t)$ denotes the light traveling along the ray that interacts the main lens at position (u, v) and the microlens plane at (s, t). By holding (u, v) fixed and considering all (s, t), the LF camera can generate an image formed by extracting the same pixel under each microlens. Hence, a LF camera can provide information of a 3D scene viewed from a continuum of possible viewpoints bounded by the main lens aperture. An example of multi-view images captured by an LF camera is shown in Fig. 1.6, where the images are simulated as been seen by slightly shifting the view position up, down, left or right. These 2D array of densely

spaced viewpoint images can be navigated freely by a user to create a sense of depth in the captured static 3D scene via motion parallex[5]. With these captured multi-view images, one can also estimate the depth-of-field information and synthesize images from virtual viewpoint [8], [11], [12], or refocus sharp photographs at different depths [3]. An example of refocusing images is shown in Fig. 1.7, where (a) is focused at the front objects and (b) is focused at the background [6].

Fig. 1.7. One example application of LF images: refocusing.

1.2 Thesis Motivations and Contributions

This thesis focuses on improving depth image coding efficiency, optimizing peer grouping for live free viewpoint video streaming, and optimally inserting landmarks to improve interactive light filed streaming (ILFS) performance. We explain this three works in the following.

[5]http://lightfield.stanford.edu/
[6]Screenshots from https://pictures.lytro.com/

1.2.1 Object Contour Approximation and Depth Image Coding

A depth image provides important geometric information of a 3D scene, namely the shapes of physical objects as observed from a particular viewpoint. To enable decoder-side virtual view synthesis, depth and color image pairs from the same viewpoints must be compressed for network transmission (usually, depth and color images are transmitted in two separate bitstreams). Traditional image codecs like JPEG, H.264 and HEVC [17] employ fixed block transforms like discrete cosine transform (DCT), where coarser quantization of transform coefficients at low bit rates will result in blurring of sharp edges. It has been demonstrated that blurring of object contours[7] in a depth image leads to unpleasant bleeding artefacts in images synthesized via DIBR [19]. Thus state-of-the-art depth image coding algorithms employ contour-adaptive transforms [18], [20]–[22] and wavelets [23] to preserve sharp object contours, which are losslessly coded as side information (SI) separately. The bitstreams of this SI and the smooth depth regions are considered together for depth image decoding. However, the SI coding cost can be expensive at low bit rate, amounting to 50% of the total depth bit budget in some cases [18].

In response, we pursue a new paradigm in depth image coding for color-plus-depth representation of a 3D scene: *in a pre-processing step, we pro-actively simplify complex object shapes in a depth and color image pair to lower depth coding cost*. In Chapter 3, we propose a greedy algorithm to approximate object contours based on a contour coding complexity model. The drawback of the greedy algorithm is that it does not take the induced distortion into consideration. To overcome this drawback, we then propose a rate-distortion (RD) optimal method in Chapter 4, where we simplify object contours to lower depth coding cost at a controlled increase in synthesized view distortion. This means that as the bit budget becomes stringent, actual shapes of physical objects in the scene are simplified, but rendering of the objects remains sharp and natural for human perception.

[7] Object contours can be detected via a gradient-based edge detection method [18]

1.2.2 Optimize Peer Grouping for Free Viewpoint Video Streaming

In FTV, a user can pull color and depth videos captured from two nearby reference viewpoints to synthesize his chosen intermediate virtual view for observation via DIBR. For users who are observing the same free viewpoint video (FVV) synchronized in time—e.g., during a live video broadcast of a public event like a piano recital—but not necessarily from the same viewpoint, they have incentive to pull color and depth video streams from the same reference views, so that the streaming cost can be shared. On the other hand, it has been shown [24], [25] that in general distortion of the synthesized view increases with its distance to the reference views. Thus, a user has incentive to select videos of reference views that tightly “sandwich" his chosen virtual view, in order to minimize visual distortion. This poses an interesting dilemma for users: how to best select and share video streams of different reference views, so that the streaming cost and the resulting collective synthesized view distortion are optimally traded off?

In Chapter 5, we investigate the reference view sharing strategies—ones that optimally trade off shared streaming costs with synthesized view distortion. We first divide users into groups. A user can simultaneously belong to two groups, and each group shares the streaming cost of a single view. Given the number of users and the virtual view that each user is watching, we aim to find a Nash Equilibrium (NE) [26] solution of reference view selection for all the users. A NE solution is stable, from which no user has incentive to unilaterally deviate. Specifically, we first derive a lemma based on known properties of synthesized view distortion functions. We then design a search algorithm to find a NE solution, leveraging on the derived lemma to reduce search complexity.

1.2.3 Optimal Landmark Insertion for Interactive Light Field Streaming

Light field imaging enables a user to navigate and observe a static 3D scene from different viewpoints. However, the volume of captured LF data is large, and downloading the entire data prior to user’s viewpoint navigation would incur a large start up delay. Instead, previous works proposed an interactive light field streaming

(IFLS) framework [27]–[30], where a user periodically requests a desired view for observation, and in response a server transmits a pre-synthesized and encoded viewpoint image. The technical challenge is to design and pre-encode a storage-constrained frame structure to facilitate user-requested view-switches during an ILFS session. Pre-encoding only I-frames[8] for all views would lead to a large transmission cost, while pre-encoding P-frames[9] for all possible user view-switch requests from any view i to j for a LF array of N views would be $O(N^2)$ and thus expensive in storage cost.

To lower transmission cost while reducing storage requirement, in Chapter 6, we design a new frame structure to facilitate ILFS via optimal selection of *landmarks*. A landmark is a designated view with P-frames to/from each neighborhood view, so that any viewpoint image can transition to any other viewpoint image by first visiting a landmark, and then from the landmark to the destination view. This results in a transmission cost of only two P-frames, and the number of stored P-frames is only $O(2N)$. The crux is how to select the optimal number and locations of landmark views, and P-frame connections to/from landmarks for the remaining views. In Chapter 6, using a Lloyd's algorithm variant, we first recursively insert into a frame structure a set of "landmarks" at locally optimal locations. We then employ a greedy algorithm to add/subtract P-frames based on a rate-storage criteria.

1.2.4 Summary of Major Contributions

In summary, we investigate three main problems in this thesis: object contour approximation for depth image coding, optimize user grouping for FVV streaming and optimal landmark insertion for ILFS. Fig. 1.8 highlights the roles of these three works in a generalized multiview image/video streaming system (applicable to both FTV and LF systems). After capturing the source multiview images/videos, a server encodes and transmits them to users and the users then display the captured or

[8] An I-frame is an intra-coded picture, which does not need other video frames for decoding. Due to the independent coding, the coding bit-count for an I-frame is large.

[9] A P-frame is differentially coded using another video frame as a predictor. Hence, a P-frame can be decoded only when the predictor is reconstructed. Compared to an I-frame, a differentially coded P-frame has a much smaller coding bit-count.

synthesized images/videos for observation. Our proposed contour approximation work is a pre-processing step before image coding, whose purpose is to improve depth image coding efficiency. The optimal user grouping in FVV or the optimal landmark insertion for ILFS is an important step to facilitate the server transmitting multiview images/videos to users.

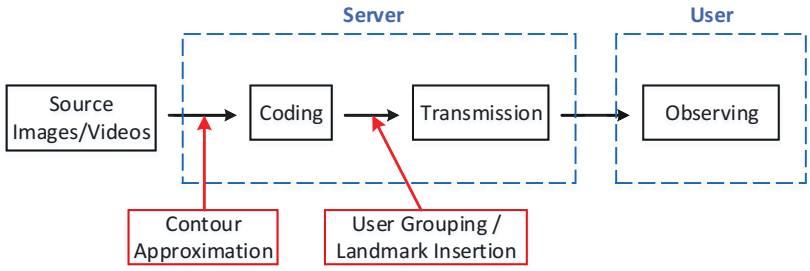


Fig. 1.8. The roles of proposed works in multiview image/video streaming system.

1.3 Thesis Outline

The thesis is organized as follows. In Chapter 2, some related works are surveyed. To improve depth image coding efficiency, we propose to first approximate object contours and then edge-adaptively code depth images. A greedy contour approximation method is first discussed in Chapter 3 and an RD optimization based contour approximation method for virtual view synthesis is proposed in Chapter 4. The optimization of peer grouping for live free viewpoint video streaming is presented in Chapter 5. In Chapter 6, we improve the interactive streaming of light field multi-view images by optimally inserting landmarks. Chapter 7 concludes the thesis, and gives possible future research directions as well.

Chapter 2

Literature Review

2.1 Depth Image

A depth image provides the geometric information in 3D representation, which plays an important role in 3D imaging applications, such as 3D reconstruction [31] and DIBR [2]. A depth image is a gray-scale image that represents the per-pixel distances between physical objects' surfaces and capturing cameras. A commonly used depth representation format is as follows [2], [19], [32]:

$$Z = \frac{1}{\frac{Y}{255} \cdot (\frac{1}{Z_{near}} - \frac{1}{Z_{far}}) + \frac{1}{Z_{far}}} \tag{2.1}$$

where Y is the pixel value of depth image with a range bewteen 0 and 255; Z_{near} and Z_{far} are the nearest and the farthest depth distance values in the physical scene and Z is the physical depth distance value for pixel value Y. With the knowledge of the per-pixel distance between the object shapes and the capturing camera, a depth image can be built. In such a depth image, pixel value 0 represents the farthest 3D point with depth being Z_{far}, and pixel value 255 represents the nearest 3D point with depth value Z_{near}. In a 3D scene, the distance Z from 3D scene to the camera gradually radiates from Z_{near} to Z_{far}. Hence depth images contain areas with slowly varying sample values. Furthermore, depth images show step function at the object boundaries. Thus, depth images show piece-wise smooth (PWS) properties, *i.e.*, smooth regions separated by sharp edges, which can be seen

from Fig. 1.3 (b) and (e).

2.2 Virtual View Synthesis

Virtual view synthesis is one of the most important step in FTV, where users' freely choosing virtual viewpoint can be synthesized by captured camera viewpoints. Here, we briefly introduce the procedures for DIBR [2], as shown in Fig. 2.1.

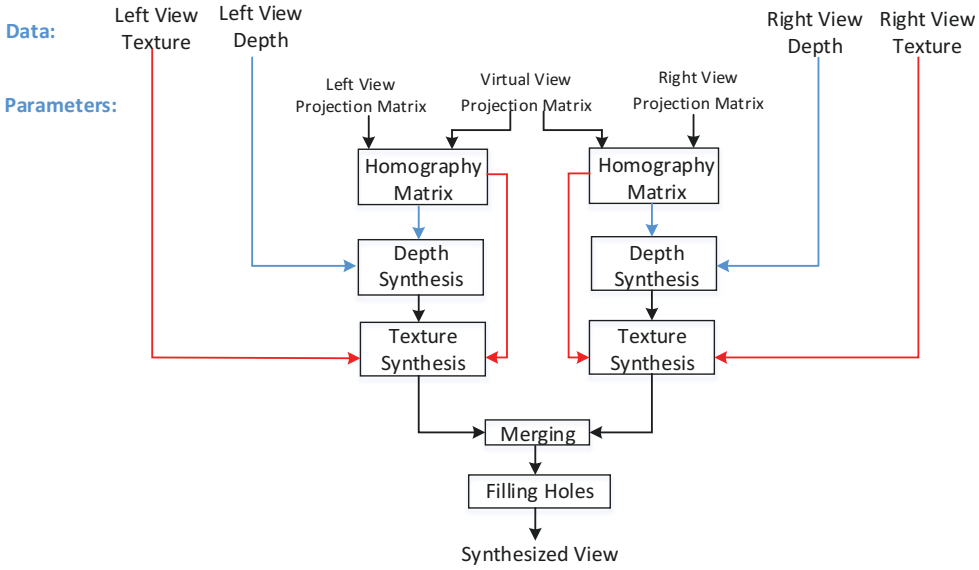


Fig. 2.1. Block diagram of DIBR [2] algorithm

The main idea of DIBR is to synthesize a new virtual view from each of reference views separately and then merging them all in one. In order to depict the problem of view synthesis, a pinhole camera geometric model is utilized, which is defined based on the world coordinate system, the camera coordinate system, the image coordinate system and camera parameters. The camera parameters, such as focal length, produce an intrinsic matrix and an extrinsic matrix. The intrinsic matrix represents the transformation from a camera coordinate to its image coordinate, while the extrinsic matrix transforms world coordinates to camera coordinates. We mark them as projection matrices. The relationship among these

three coordinates are as follows:

$$z_C \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A_{3\times3} \cdot \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} \tag{2.2}$$

$$\begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} = R_{3\times3} \cdot \begin{pmatrix} x_W \\ y_W \\ z_W \end{pmatrix} + t_{3\times1} \tag{2.3}$$

where $A_{3\times3}$ is the intrinsic matrix, $R_{3\times3}$ and $t_{3\times1}$ are rotation matrix and a translation vector, respectively, which compose the extrinsic matrix $E = [R|t]$. The intrinsic matrix and extrinsic matrix can be computed given camera parameters. The (u, v), (x_C, y_C, z_C) and (x_W, y_W, z_W) are image, camera and world coordinates, respectively.

We now look at how to map a pixel located at (u_r, v_r) in the image coordinate system of the reference view to its correspondent location (u_v, v_v) in the virtual view, with the provided reference depth image information. Given (u_r, v_r), from (2.2) and (2.3), we first obtain the world coordinate of a 3D point corresponding to (u_r, v_r) as

$$\begin{pmatrix} x_W \\ y_W \\ z_W \end{pmatrix} = R^{-1}_{3\times3,r} \cdot \left(z_{C,r} \cdot A^{-1}_{3\times3,r} \begin{pmatrix} u_r \\ v_r \\ 1 \end{pmatrix} - t_{3\times1,r} \right) \tag{2.4}$$

where the subscription r indicates an item (matrix or coordinate) related to the given reference view, and $z_{C,r}$ is the depth value calculated from the reference depth image using (2.1) given the depth pixel value located at (u_r, v_r). Again from (2.2) and (2.3), since the world coordinate for reference and virtual views is the same, we

can map this 3D point to the image coordinate system of the virtual view as

$$z_{C,v} \cdot \begin{pmatrix} u_v \\ v_v \\ 1 \end{pmatrix} = A_{3\times3,v} \cdot \left(R_{3\times3,v} \begin{pmatrix} x_W \\ y_W \\ z_W \end{pmatrix} + t_{3\times1,v} \right) \tag{2.5}$$

where a subscription v indicates an item related to the virtual view. When known the distance $z_{C,v}$ between objects to the virtual camera, we can compute the correspondent location (u_v, v_v) in the virtual view. We can then map the color image pixel value located at (u_r, v_r) in the reference view to location (u_v, v_v) in the virtual view. Above shows the general 3D rendering procedure.

Combining (2.4) and (2.5) together we get our homography matrix, which maps depth pixel position from reference view to virtual view. With this homography matrix and the color and depth information in referencing view, we do depth and color synthesis for a single path of the virtual view. At the end of each path, the synthesized images from all reference views (paths) are merged together. Due to the fact that there are some pixel locations that never get mapped from reference views—pixel locations are occluded in both reference views, black holes are resulted in the merged view. Then depth-based inpainting techniques [33][34] are utilized to complete the hole fillings.

In summary, DIBR essentially maps color pixels in the reference views to appropriate pixel locations in a virtual view; such locations are derived from the corresponding depth pixels in the reference views. Disoccluded pixels in the synthesized view—pixel locations that are occluded in the two reference views—can be recovered using depth-based inpainting techniques [33], [34]. Because inpainting offers only a best-guess solution, the larger the disoccluded regions are, the lower the synthesized view image quality will be in general.

2.3 Contour Approximation and Depth Image coding for Virtual View Synthesis

We divide the review of related work on object contour approximation and depth image coding into three sections. We first discuss existing literature on depth image coding. We then overview previous work on image contour coding. Finally, we discuss existing work on shape approximation.

2.3.1 Depth Image Coding

Because of the popularity of color-plus-depth representation format of a 3D scene [35], depth image compression has drawn a growing interest. Typical image coding algorithms such as JPEG [15] and H.264/AVC standard [36] employ fixed block-based transforms like 2D DCT on images. Then quantization and entropy coding are applied to the transform coefficients. In practice, this conventional DCT is implemented separately through two 1D transforms, one along the vertical direction and the other along horizontal direction. As a result, DCT based image coding algorithms can only well represent image patches with horizontal and vertical edges. Considering that there exist some other directions of edges in a image block that are perhaps as equally important as the vertical/horizontal direction, the work in [37] proposes a directional DCT transform for image coding. They develop their first transform following a main direction in the block other than the vertical or horizontal one, while the second transform is arranged to be a horizontal one. This directional DCT framework can be adapted to diagonal edges, but it cannot deal with more arbitrarily shaped edges such as “L" and “V", as a number of unnecessary non-zero alternating components (AC) coefficients will be generated. Practically, this means that coarse quantization of the high-frequency components in these transforms at low bit rates budget will lead to blurring of arbitrarily shaped edges in the reconstructed depth image.

Observing that depth image contours play an important role in the quality of the DIBR-synthesized view [19], contour-preserving image coding algorithms have been proposed. Many of these works exploit depth images’ PWS characteristics:

smooth interior surfaces separated by sharp contours. The work in [38] models depth images with piecewise linear functions (platelet) in each image block. A quadtree decomposition is adopted to sub-divide the image into variable sized blocks until each block could be approximated with one platelet model depending on its edge information. However, the representation inherently has a non-zero approximation error even at high rates, since depth images are not strictly piecewise linear but PWS. Works in [39] and [23] propose contour-adaptive wavelets using lifting to avoid filtering across edges. Recently, the work in [20] proposes block-based unweighted graph Fourier transform (GFT) for depth image coding. They divide images into sub-blocks and in each block they build a 4-connected graph in which each pixel is connected to its immediate neighbors only when they are not separated by an edge. They then construct a GFT based on this graph. The main advantage of GFT is that it only filters pixels that are connected in the graph, which automatically avoids filtering across edges. Extending [20], the work in [18] searches for an optimal weighted graph for GFT-based image coding in a multi-resolution framework. In all these works, detected edges are encoded losslessly as SI, which can cost up to 50% of the total budget at low rates.

Since depth images are usually used for synthesizing virtual views at the decoder side and are not directly viewed, it is necessary to consider the resulting synthesized view distortion, instead of depth distortion itself, for RD optimization during depth image coding [19], [32]. How depth distortion is related to synthesized view distortion has been investigated [40]–[42]. For example, the work in [40] proposes a synthesized view distortion function as the multiplication of depth distortion and local color information. In 3D-HEVC [43]–[45], the advanced 3D video extension of HEVC, depth images are coded along with color images, where a linear plane fitting model is used to describe a smooth region in depth images, and a wedgelet partition or contour partition is adopted to code a depth block with edges in order to preserve edge sharpness. Synthesized view distortion is considered for optimal depth image coding mode selection. However, these methods assume peak signal-to-noise ratio (PSNR) is the quality metric for the synthesized view when

compressing depth images, which has been demonstrated to not correlate well with actual human perception [5].

In [5], the authors show that while objects in a synthesized view may be slightly shifted due to the DIBR-rendering process, the overall visual quality of the image may still be acceptable. However, this type of artifacts is penalized by pixel-by-pixel based quality metric such as PSNR. They thus propose a 3D synthesized view image quality metric (3DSwIM) tailored specifically for artifacts in DIBR synthesized views. They conclude that this metric has a higher correlation with subjective quality assessment scores than PSNR. Using 3DSwIM as the chosen metric, in Chapter 4 we approximate object contours and augment depth/color image pairs for more efficient coding of 3D image content.

There are other pre-processing methods that attempt to improve video/image coding efficiency. Works in [46]–[49] apply filters on input videos/images to remove noise. The work in [50] corrects the color value of multi-view videos to make them consistent, and the work in [51] evaluates the coding performance of applying a color image guided filter on the estimated low quality depth image. Different from these works, we propose to modify the geometry structure of a 3D scene to improve color-plus-depth image coding efficiency.

2.3.2 Object Contour Coding

Freeman chain code [52] is widely used to efficiently encode object contours [53]–[55]. Usually, object contours are first converted into a sequence of symbols, where each symbol is from a finite set with four or eight possible absolute directions. Alternatively, the relative directions between two neighbouring directions, which is also known as *differential chain code* (DCC) [56], can also be used. Given the probability of each symbol, the contour chain code is entropy-coded losslessly using either Huffman [54] or arithmetic coding [57].

The works in [58] [59] introduce an *arithmetic edge coding* (AEC) method employing a linear *geometric model* to estimate the probability of next edge. Given a small window of previous edges, they first construct a best-fitting line that

minimizes the sum of squared distances to the end point of each edge. Then the probability for the next edge direction is assigned based on the angle difference between the edge direction and the best fitted line, which is subsequently used as context for arithmetic coding. The assigned probability for current edge only depends on previous encoded edges. We will adopt AEC model to code object contours during our contour approximation.

2.3.3 Image Contour Approximation

There exist some contour approximation methods in the compression literature. The work in [60] proposes a polygon / spline-based representation to approximate a contour. Vertices are encoded differentially and polygon curves between neighbouring vertices are considered to approximate the original contours. Later, the authors in [61]–[64] improve this vertex-based shape coding method. All of these works take the maximum or mean distance between original and approximated contours as distortion measure, which is not appropriate for assessment of visual quality of synthesized views.

Some other works approximate contours from the chain code representation. The method in [65] saves coding bits by omitting two neighboring turning points if the slope between them is nearly vertical or horizontal. The algorithms in [59] and [66] replace some pre-defined irregular edge patterns with a smooth straight line. In Chapter 3, we propose to approximate object contours based on a contour complexity model to lower contour coding rate. Later in Chapter 4, we take the synthesized view distortion induced by contour approximation into consideration. We propose to approximate contours using the 3DSwIM metric to control the induced synthesized view distortion, which leads to more pleasant results in view synthesis.

2.4 Peer Grouping in Live Video Streaming

We divide the overview for peer grouping of live video streaming into two areas. We first discuss the literature on multi-view video streaming. We then discuss the related work on collaborative streaming of single view video.

2.4.1 Multi-View Video Streaming

As technologies for compression of color and depth images for free viewpoint video become more mature [67][68], research focus has shifted to the streaming and distribution of this new media type. The work in [69] designs a multi-view video compression algorithm in combination with an observer's head position prediction scheme, so that the likely captured video views to be observed by client in the near future are automatically pulled from server. Works in [70][71] study the problem of how smartly encoded multi-view video that facilitates view-switching can be replicated in storage-constrained distributed servers across a network to minimize view-switching delay. Works in [72][73] investigate how color and depth videos can be unequally protected to minimize the synthesized view distortion when streaming over a network is prone to packet losses. None of these prior streaming works study the problem of how video streams of different views can be optimally selected and shared among users observing different virtual views, which is one focus of our work in this thesis.

2.4.2 Collaborative Video Streaming

On the other hand, video sharing for single-view video, mostly for Peer-to-Peer (P2P) video streaming, has been studied extensively in the literature. For example, the work in [74] derives a stochastic fluid model to analytically reveal the characteristics of P2P streaming systems and exposes the key designing features to achieve a satisfactory system performance. The work in [75] studies a real world large-scale P2P streaming system to gain insights for successful deployment of such systems. The work in [76] reviews different overlay network structures for both P2P live streaming and video-on-demand. However, all these works studies single-view video streaming, which is passive in nature (*i.e.,* no view-switching), and the results cannot be directly applied to the live free viewpoint scenario (*i.e.,* each viewer can freely select a virtual view for observation). How to select and share the reference views to address the tradeoff between the streaming cost and the synthesized view distortion is a key issue for live free viewpoint video distribution, which we study

in our work.

2.5 Interactive Light Field Streaming

A light field is a 2D array of spatially correlated images of a static 3D scene. ILFS is the application where users continuously request successive light field images along a view trajectory, and in response the server transmits appropriate data to users so that they can correctly reconstruct desired images.

ILFS is first studied in [28][27]. In [28], new switching mechanisms to adjacent views based on Wyner-Ziv coding are proposed. However, the navigation model (which only permits switches to horizontal and vertical adjacent views) is limited. In our work, we discuss a more general view navigation model. The work in [27] employs SP-frame [77] to improve compression efficiency during random accessing light filed images. But they do not consider the storage cost at the server for storing multiple representations of images. In contrast, we optimally trade off the storage cost of the frame structure with expected transmission cost in our work. Works in [78][79] propose rate-distortion optimized ILFS, where they study bit-stream packetization and packets scheduling during light field image streaming. The work in [80] focuses on streaming the dynamic real-time captured light filed video to users in school network environment. All these three works are orthogonal to our problem, where we investigate how to design a reasonable sized frame structure of light field images a priori (without knowing eventual clients' view trajectories) storing at the server.

More recent studies on ILFS are proposed in [29][81], where they have employed a more general view navigation model for ILFS. But the focuses are on the use of new *distributed source coding* (DSC) frames [82] and merge frames [83] for view-switching without coding drift. The work in [30] discusses the notion of landmark to be used in ILFS but they did not provide an explicit landmark insertion strategy. Orthogonally, the work in [84] partitions the 3D scene into segments for efficient coding and streaming. In contrast, we focus on the optimal selection of landmarks to facilitate ILFS. A landmark operates like an airline hub in commercial

aviation[1]: by creating direct flights to / from a designated hub for all cities—$O(2N)$ flights for N cities—a passenger can travel from any city to any other city via only two flights (one connecting flight). In our work, with the proper insertion of landmarks, we can not only lower transmission cost, but also reduce storage requirement.

[1] As an example, United Airline has its largest domestic hub in Chicago O'Hare International Airport.

Chapter 3

Contour Approximation and Depth Image Coding: A Greedy Algorithm

In this chapter, object contour approximation for depth image coding is investigated. A greedy algorithm is proposed to approximate object contours to save depth coding rate. [1]

3.1 Introduction

A depth image provides geometric information of a 3D scene, namely the shapes of physical objects captured from a particular viewpoint. This information can be used, together with texture images (color images like RGB) from the same viewpoint(s), to synthesize novel images as observed from virtual viewpoints via DIBR.

To enable decoder-side virtual view synthesis, depth and color image pairs from the same viewpoints must be compressed together for network transmission. As introduced before, depth contours play an important role in the quality of synthesized virtual view and losslessly contour coding is expensive at low rates. In this chapter, we argue that while it is important to maintain a depth image's PWS characteristic even during lossy compression, the object contours themselves can be suitably approximated to reduce SI coding costs. Specifically, we first define a notion of complexity that estimates the SI coding costs of object contours in a

[1] A version of this chapter has been published in *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*.

given depth image based on edge statistics. Given an initial rate estimation, we then pro-actively approximate object contours in such a manner that guarantees rate reduction when the simplified contours are coded using AEC [58]. Given the sharp but approximated contours, finally we encode the depth image using an edge-adaptive image codec with GFT for edge preservation [18], [21].

The outline of this chapter is as follows. We first discuss how the PWS signal can be modeled and approximated for the 1D case. We then extend the complexity model to 2D case and approximate 2D contours. Experimental results and concluding remarks are presented at the end.

3.2 Geometry Approximation: 1D Case

We start our discussion on approximation of PWS signals with the simpler 1D case first (depth pixel row). We first define a notion of "complexity" for 1D PWS signals, which we use as a proxy for a signal's coding cost during actual compression. We then present an approximation algorithm that can reduce a signal's complexity while maintaining its PWS characteristic.

3.2.1 PWS Signal Complexity: 1D Case

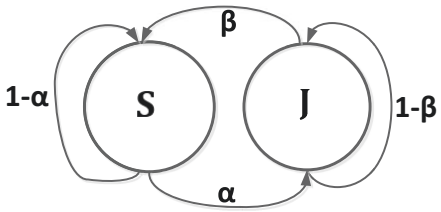


Fig. 3.1. A two-state Markov model for PWS signal

It is observed that depth images exhibit the PWS characteristic [20], [21], *i.e.*, smooth surfaces divided by sharp edges. For a 1D discrete signal (a pixel row), PWS means a smooth 1D signal interrupted occasionally by discontinuities. Given that the smooth portions can be efficiently coded using edge-adaptive wavelets [23] and transforms [18], [20], we are interested here only in the complexity of representing the set of discontinuities. To model this set, we build a 2-state Markov model as shown in Fig. 3.1: **S** and **J** are the "smooth" and "jump" (discontinuity) states,

respectively, with state transition probabilities $Pr(\mathbf{J}|\mathbf{S}) = \alpha$ and $Pr(\mathbf{S}|\mathbf{J}) = \beta$. We now define the complexity of the 1D PWS signal as the entropy $H(\phi)$ of state random variable $\phi \in \{\mathbf{S}, \mathbf{J}\}$, where

$$H(\phi) = \log(\alpha + \beta) - \frac{\alpha}{\alpha + \beta} \log \alpha - \frac{\beta}{\alpha + \beta} \log \beta. \tag{3.1}$$

Fig. 3.2 shows complexity $H(\phi)$ as a function of α ($\alpha \leq 0.5$) for different values of β.

We assume that there cannot be two back-to-back discontinuities for a 1D signal, hence $\beta = 1$ and the complexity becomes: $H(\phi) = \log(1 + \alpha) - \frac{\alpha}{1+\alpha} \log \alpha$. The more discontinuities a PWS signal has, the larger $Pr(\mathbf{J}|\mathbf{S}) = \alpha$ is, and the larger the complexity $H(\phi)$ is. We argue that complexity $H(\phi)$ of a PWS signal is positively correlated with the bit count required to encode the signal's discontinuities. An empirical verification will be provided in section 3.3.2.

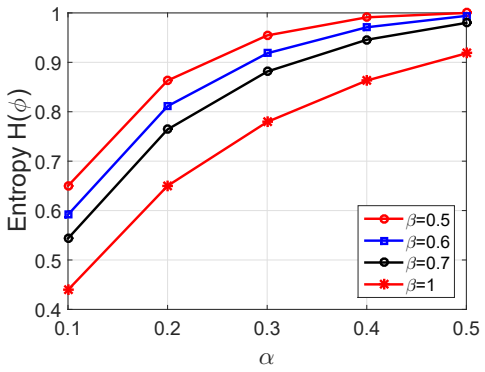


Fig. 3.2. The entropy $H(\phi)$ in (3.1) as a function of α with different values of β

3.2.2 Approximation of 1D PWS Signal

If the complexity $H(\phi)$ of a PWS signal is large, a large coding cost is needed. One can reduce the number of discontinuities in the signal, resulting in a smaller α and $H(\phi)$. For simplicity, we further assume that the smooth segments in a 1D PWS signal are constant values, *i.e.*, we assume 1D PWS signal as *piecewise constant* (PWC) (see $F(x)$ in Fig. 3.3 as an example). We then have the following lemma for 1D signal approximation.

Lemma 1: If an original PWC signal containing N jumps is best approximated (in mean squared error (MSE)) by a signal with only M jumps, $M < N$, then the M jump locations in the approximated signal are a subset of the original N jump locations, and the value of each new longer constant signal segment spanning multiple original pieces is simply the average of those pieces. (Refer to Appendix 3.A for a proof.)

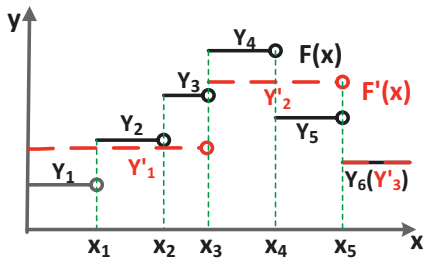


Fig. 3.3. An example of 1D PWC signal $F(x)$ in black, and an approximated signal $F'(x)$ in red.

Fig. 3.3 shows one example of approximating original PWC signal $F(x)$ (with 5 jumps and 6 constant pieces) to signal $F'(x)$ (with 2 jumps and 3 constant pieces). The jump locations in $F'(x)$ are coincident with original jumps locations x_3 and x_5 in $F(x)$. The constant value of each piece in $F'(x)$ is the mean value of the original $F(x)$ pieces which are spanned by this new piece, *e.g.* the value Y_1' is given as $\frac{x_1 Y_1 + (x_2 - x_1) Y_2 + (x_3 - x_2) Y_3}{x_3}$.

Given the lemma, we now look at how to compute the best approximated signal via *dynamic programming* (DP) to identify the best M jump locations among N possible choices as follows.

Denote by $\mathbf{x}$ a PWC signal with $N+1$ constant pieces and N jumps. A PWC signal $\mathbf{x}'$ with M jumps that best approximates $\mathbf{x}$ must eliminate $(N-M)$ original jumps. We define $D_{\mathbf{x}}(i,k)$ as a recursive function that returns the minimum MSE of approximating from the i-th constant piece of $\mathbf{x}$ to the $(N+1)$-th piece by eliminating k jumps. We first define $d_{\mathbf{x}}(i,j)$ as the MSE when a constant signal segment approximates $j-i+1$ pieces with indices from i to j, where $j \in [i, N+1]$ and $d_{\mathbf{x}}(i,i) = 0$. $d_{\mathbf{x}}(i,j)$ eliminates $j-i$ jumps between i-th and j-th pieces and leaves $k-(j-i)$ more jumps to be eliminated starting from the $(j+1)$-th constant

piece. $D_{\mathbf{x}}(i,k)$ selects the j-th piece that results in the minimum total distortion:

$$D_{\mathbf{x}}(i,k) = \min_{j\in[i,N+1]} \big\{ d_{\mathbf{x}}(i,j) + D_{\mathbf{x}}(j+1, k-(j-i)) \big\}. \tag{3.2}$$

Some base cases are defined as

$$D_{\mathbf{x}}(i,k) = \begin{cases} 0, & \text{if } k = 0,\ \forall i \in [1, N+2] \\ +\infty, & \text{if } k > 0 \ \&\ (N+1) - i < k \\ +\infty, & \text{if } k < 0. \end{cases} \tag{3.3}$$

The first case, $k = 0$, means no more jumps need to be eliminated and thus $D_{\mathbf{x}}(i,0) = 0$. The second case means that from the i-th constant piece to the end of the signal, the number of existing jumps is less than the required k. The last case, $k < 0$, happens when $(j-i) > k$ in Eq. (3.2). We define $D_{\mathbf{x}}(i,k) = +\infty$ for the last two cases to imply a violation.

According to Eq. (3.2), a recursive call of $D_{\mathbf{x}}(1, N-M)$ would yield the minimum total distortion for approximating signal $\mathbf{x}$ with M jumps. The best M jump locations in $\mathbf{x}'$ can be identified accordingly. This DP algorithm could be implemented either by top-down with memorization or bottom-up method with complexity $O(N^2)$ [85].

3.3 Geometry Approximation: 2D Case

We now generalize our previous approximation of 1D PWS signal to 2D. We will first review one previous contour coding scheme—*arithmetic edge coding (AEC)*. Based on that, we then introduce a similar notion of complexity for contours in a depth image—an estimate of the overhead for contour coding in the image. Finally, to reduce the complexity we describe a procedure to approximate the contours to simpler ones.

3.3.1 Arithmetic Edge Coding

Given a depth image, we first detect edges via a gradient-based method [18], where the edges exist *between* pixels and outline contours of physical objects. Fig. 3.4

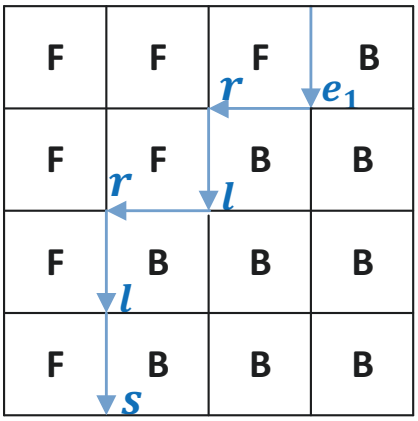


Fig. 3.4. Edges in a 4×4 block that separate foreground (F) from background (B).

shows an example of a 4×4 block with *between-pixel edges*—each edge exists between a pair of vertically or horizontally adjacent pixels—separating foreground and background depth values. The set of contiguous edges composing a contour is described by a sequence of symbols $\mathcal{E} = \{e_T, \ldots, e_1\}$ of some length T. Each e_t, $t > 1$, is chosen from a size-three alphabet—$e_t \in \mathcal{A} = \{\mathtt{l}, \mathtt{s}, \mathtt{r}\}$—representing *relative* directions left, straight and right respectively, with respect to the previous edge e_{t-1}. In contrast, the first edge e_1 is chosen from a size-four alphabet—$e_1 \in \mathcal{A}^o = \{\rightarrow, \downarrow, \leftarrow, \uparrow\}$—denoting *absolute* directions east, south, west and north respectively. This contour representation is also known as DCC [52].

To efficiently encode a symbol e_{t+1} in sequence $\mathcal{E}$, using a small window of K previous edges $\{e_t, \ldots, e_{t-K+1}\}$ as context, AEC [58], [59] uses a linear regression model to estimate probabilities $Pr(e_{t+1} = \theta)$ of the three possible directions $\theta \in \mathcal{A}$, which are subsequently used for arithmetic coding [57] of e_{t+1}. Specifically, using $\{e_t, \ldots, e_{t-K+1}\}$, a *best-fitting* line l with direction u_l is constructed via linear regression. "best-fitting" means that the line l minimizes the sum of squared distances from the end point of each edge in $\{e_t, \ldots, e_{t-K+1}\}$ to line l. See Fig. 3.5 for an illustration, where the line l is the best fitting line given three edges $\{e_3, e_2, e_1\}$.

In [58], [59], the angle $\gamma_\theta \in [0, \pi]$ between a relative direction $\theta \in \mathcal{A}$ and u_l is first computed, then the probability $P(e_{t+1} = \theta)$ for edge e_{t+1} is defined such that

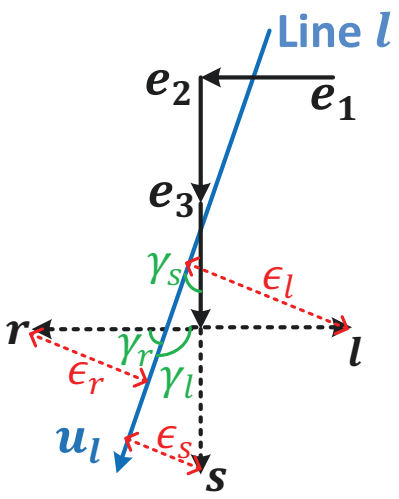


Fig. 3.5. Given edges $\{e_3, e_2, e_1\}$ to estimate the probability of edge e_4. With the best-fitting line l and its direction u_l, the angel difference γ_θ and distance ϵ_θ for $e_4 = \theta, \theta \in \mathcal{A}$ is illustrated. The predicted direction $v_p = \mathtt{s}$.

smaller γ_θ leads to larger $P(e_{t+1} = \theta)$. In our work, we consider also the minimum distance ϵ_θ from the end point of e_{t+1} to line l when determining $P(e_{t+1} = \theta)$. Specifically, We define $P(e_{t+1} = \theta)$ as:

$$P(e_{t+1} = \theta) = \frac{1}{2\pi I_0(\kappa)} \cdot \exp\{\kappa \cos \gamma_\theta\} \cdot \exp\left\{-\frac{\epsilon_\theta^2}{2\omega^2}\right\} \tag{3.4}$$

where $I_0(\cdot)$ is the modified Bessel function of order 0. The parameter $1/\kappa$ is the variance in the circular normal distribution. ω is a chosen parameter to adjust the relative contribution of the distance term $\exp\{-\frac{\epsilon_\theta^2}{2\omega^2}\}$. The distance term is added to the Von Mises probability distribution model [2] to differentiate the case where there exist two directions with the same γ_θ (*e.g.*, a diagonal straight line). The computed probabilities $P(e_{t+1})$, synchronously computed at both the encoder and the decoder, are then used for arithmetic coding of the actual edge e_{t+1}, where a larger probability leads to a smaller number of coding bits.

In words, (3.4) states that the direction probability increases as the angle γ_θ and distance ϵ_θ decrease. In the previous example in Fig. 3.5, with the best-fitting line l and line direction u_l, the angle difference γ_θ and distance ϵ_θ for $\theta \in \mathcal{A}$ are

[2] http://en.wikipedia.org/wiki/Von_Mises_distribution

illustrated. The relative direction θ with the smallest angle and distance with respect to u_l is $\mathtt{s}$ in this example.

3.3.2 PWS Signal Complexity: 2D Case

Using the linear regression model to estimate edge direction probabilities, we can define a notion of "smoothness" for contours in 2D: *a smooth contour is one with edges that are mostly predictable by the geometry-driven probabilistic model*. We can thus reuse our 2-state Markov model for 1D discontinuities to compute a complexity measure for a given contour. Specifically, among the three possible directions $\theta \in \mathcal{A}$ for next edge e_{t+1}, we term the direction θ with the largest estimated probability the *predicted direction* v_p (*e.g.*, the predicted direction is $\mathtt{s}$ in Fig. 3.5). If $e_{t+1} = v_p$, then e_{t+1} conforms to the prediction model and we transition to the state $\mathbf{S}$. Otherwise, e_{t+1} does not match with the predicted v_p, and we transition to the state $\mathbf{J}$. Thus we can trace state transitions in the same 2-state Markov model as in Fig. 3.1 for a given sequence $\mathcal{E}$ and a given geometry-driven prediction model.

In practice, the model parameters α and β can be computed using a training dataset as follows. We first tabulate the number of visits $Q(\phi)$ to state ϕ, $\phi \in \{\mathbf{S}, \mathbf{J}\}$, in the training dataset. We then tabulate the number of occurrences of the four possible state transition pairs, $Q([a, b])$, $a, b \in \{\mathbf{S}, \mathbf{J}\}$, for state transition from a to b. We see that $Q(\phi) = Q([\phi, \mathbf{S}]) + Q([\phi, \mathbf{J}])$. Finally, we calculate the model parameters as: $\alpha = Q([\mathbf{S}, \mathbf{J}])/Q(\mathbf{S})$ and $\beta = Q([\mathbf{J}, \mathbf{S}])/Q(\mathbf{J})$.

Hence, we can also define the complexity of contours in a 2D depth image as the entropy $H(\phi)$ in Eq. (3.1). In Fig. 3.6, we plot the bit count of encoded contours in a 2D depth image using AEC, versus our complexity metric of the contours multiplied by the contour lengths. We observe a positive correlation between bits per symbol and the complexity, which verifies that the complexity $H(\phi)$ of contours is positively correlated with the bit count required to encode the contours.

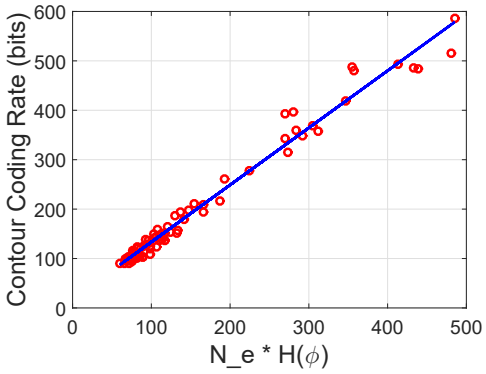


Fig. 3.6. The contour coding bits versus our complexity metric $H(\phi)$ for contours in a 2D depth image multiplied by the length of contours N_e.

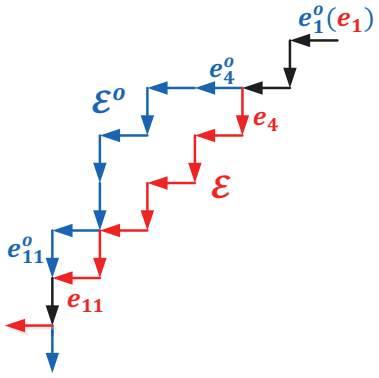


Fig. 3.7. Greedy Contour Approximation: the original contour $\mathcal{E}^o$ (in blue) is approximated by contour $\mathcal{E}$ (in red). Here $K = 3$. Edges in black means original and approximated edges are coincident.

3.3.3 Greedily Approximation of 2D PWS Signal

We now describe a procedure of contour approximation to reduce the complexity $H(\phi)$ if it is too large (and hence bit count for coding the contours is too large). Recall from Fig. 3.2 that a larger α results in a higher complexity. Hence to lower complexity, we prefer smaller occurrences of $Q([\mathbf{S}, \mathbf{J}])$ and larger occurrences of $Q([\mathbf{S}, \mathbf{S}])$. Specifically, We first define a *horizon* h, which is the size of a window of candidate edges in which we will consider contour approximation. A larger h will naturally lead to a larger search space, resulting in more approximation. At a given edge e_{t+1}^o, with a sequence of K previous edges $\{e_t^o, \ldots, e_{t-K+1}^o\}$ of an original contour $\mathcal{E}^o$, we compute the predicted direction v_p for edge e_{t+1}^o using the

previously described AEC model. We then consider an approximated contour $\mathcal{E}$ that replaces the original edge e^o_{t+1} with v_p, and continue to replace future edges with next predicted v_p, until either: i) a predicted edge v_p is actually part of the original contour $\mathcal{E}^o$, or ii) the approximation has reached length h. If it is the former termination condition, then we have successfully replaced a section of the original $\mathcal{E}^o$ with a new segment that is perfectly predicted by the linear regression model, thus lowering α and the complexity $H(\phi)$. If it is the latter termination condition, then no approximation is executed, and the procedure repeats at edge e^o_{t+h}.

Fig. 3.7 shows an example of approximating an original contour $\mathcal{E}^o$ with $K = 3$. Given the first 3 edges and selecting the predicted direction based on AEC for every next edge, a more predictable contour is shown in red as $\mathcal{E}$. According to our proposed scheme, if the horizon $h \geq 12$, then we can replace edge segment $\{e^o_4, \cdots, e^o_{11}\}$ on the original contour $\mathcal{E}^o$ with the segment $\{e_4, \cdots, e_{11}\}$ on approximated contour $\mathcal{E}$ to lower coding cost. Otherwise, if $h < 12$, then no approximation happens. Some subjective approximation results for the `teddy` image are illustrated in Fig. 3.8, where a larger h leads to smoother contours.

(a) `teddy`

(b) (c) (d) (e)

Fig. 3.8. (a) `teddy`. (b) original interception of (a). (c) $\sim$ (e) are the approximation results with increasing values of h. The contour becomes smoother with h increasing.

3.3.4 Edge-adaptive Coding of Depth Blocks

Given the approximated contours, coded losselessly using AEC, we now describe how actual pixel values of a depth image can be coded edge-adaptively block-by-block via GFT [21], preserving the image's PWS characteristic. First, some pixel values on a depth image are modified according to the approximated contours, so that foreground depth values fall on one side of a contour and background depth values fall on the other side. Then, for each pixel block a 4-connected graph is constructed, where each node corresponds to a pixel. An edge weight is assigned 1 if the connected pixels are on the same side of a contour, and 0 otherwise. Given the constructed graph, the transform is the eigen-matrix of the graph Laplacian. This weight assignment (deducible from the encoded contours) means that filtering across sharp boundaries is avoided, preventing blurring of edges. The computed transform is multiplied with the vectorized depth block signal for the transform coefficients, which are quantized and entropy-coded into a bitstream. See [18] for details.

3.4 Experimentation

3.4.1 Experimental Setup

Using color-plus-depth image sequences from the Middlebury dataset[3], we perform extensive experiments to validate the effectiveness of our proposed depth image contour approximation method . The resolution of both color and depth images are fixed at 368×416. For `teddy` and `cones`, the second and sixth views are used as the left and right reference views, and the fourth view is used as the target virtual view for DIBR view synthesis. For the other sequences, the first and fifth views are the reference views, and the third view is the target virtual view. We first approximate contours in each original depth image with different horizon scale α, where the horizon value h is equal to the length of contours times the scale. We then deploy GFT [20] to code pixel blocks in depth images (termed approximated GFT or `AGFT`

[3] http://vision.middlebury.edu/stereo/data/

TABLE 3.1
BG GAIN IN PSNR FOR MIDDLEBURY SEQUENCES

Sequences	GFT	JPEG
teddy	1.29	1.45
cones	0.92	3.34
Dolls	2.17	2.41
Rocks	2.30	7.14
Lampshade	1.72	9.01
Average	1.68	4.67

for short). Finally combining with original or JPEG compressed color images, we synthesize middle views via DIBR [2], which are compared against original middle view images for distortion computation.

3.4.2 Results: Comparing with GFT Compressed Original Depth Image

We first demonstrate the benefit of effectively approximating contours before edge-adaptive image compression. Virtual images which are synthesized using depth images compressed by GFT with losslessly coded original edges (termed GFT) are adopted for comparison. We compare the RD curves of synthesized virtual view by our proposed AGFT with GFT. The same color images compressed by JPEG are used for synthesizing. Fig. 3.9 shows the RD curves for cones, Dolls, Rocks and Lampshade sequences, respectively. The x-axis is the depth image coding rate in bits per pixel (bpp), and the y-axis is the synthesized view quality in PSNR. The computed Bjontegaard delta (BD) PSNR gains [86], [87] are shown in Table 3.1, where we see that the average BG gain in PSNR achieves 1.68dB over GFT compressed original depth image and 4.67dB over JPEG compressed original depth image.

Our approximated depth images results in better RD performance at low bit rates, since by approximating contours we can save significant coding bits while the synthesized view distortion due to approximation is negligible. Subjective results will be shown in the next subsection.

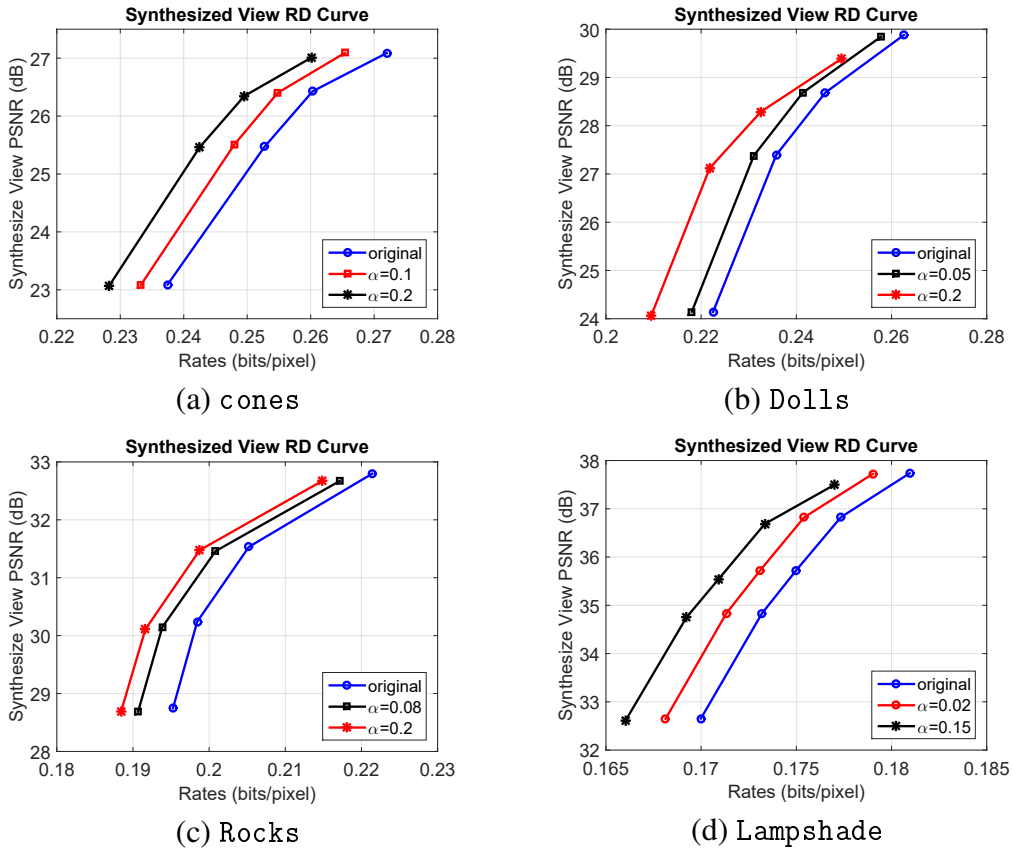


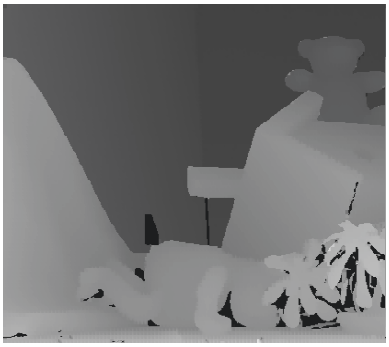
Fig. 3.9. Synthesized virtual view RD curves for `cones`, `Dolls`, `Rocks` and `Lampshade`, respectively. Here 'original' means virtual view synthesized using depth images compressed adaptive to original edges. "α" referred curves meaning virtual view synthesized with correspondingly approximated depth image based on our proposed method.

3.4.3 Results: Comparing with JPEG Compressed Depth Image

We now demonstrate the importance of edge preservation in compressed depth images. We compare synthesized images using depth images compressed by our `AGFT`, which preserves sharp edges, with depth images compressed by JPEG. Fixed DCT-transform coding in JPEG compression leads to blurred edges at low bitrates. Since section 3.4.2 has shown our `AGFT` has better RD performance than `GFT`, and it has already been demonstrated in [21] that `GFT` outperforms JPEG in depth image compression in terms of RD performance, here we do not compare our `AGFT` with JPEG in RD performance. Rather, we show here that even for similar depth image PSNR, the resulting synthesized images using `AGFT` compressed depth images outperform images synthesized using JPEG compressed depth images, since `AGFT`

maintains the PWS characteristic.

Fig. 3.10 shows the AGFT and JPEG compressed depth images and the corresponding synthesized virtual views. We observe that AGFT compressed depth image (a) has sharper edges comparing with JPEG compressed result (c), even though they have nearly equal PSNR values (the difference is less then 0.1dB for both left and right depth images). (b) and (d) are the corresponding synthesized views. We enlarge the right-bottom corner for each image to enhance the visual quality, where there are corrupted edges in (d) while (b) looks clearer with sharp edges. This is also reflected in the corresponding PSNR values. The results illustrate that even for the similar PSNR quality of depth images, an edge-preserved depth image can result in a higher quality of synthesized view, which supports the importance and effectiveness of our edge adaptive approximation method.



(a) By AGFT, 34.93dB

(b) Virtual View, 29.54dB

(c) By JPEG, 34.87dB

(d) Virtual View, 27.60dB

Fig. 3.10. For teddy, depth images with nearly equal PSNR, compressed by AGFT (a) (coding bits: 0.20 bpp) and JPEG (c) (coding bits: 0.72 bpp), respectively. Together with original color images, the corresponding synthesized views are shown in (b) and (d). Visually, (d) has a lot of noise around edges while (b) is very clean. The right-bottom corner is enlarged for better visual quality.

Sub-regions of the synthesized views for the previous mentioned four sequences are shown in Fig. 3.11. The first row views are synthesized by AGFT compressed depth images and second row views are by JPEG compressed depth images, where the depth image quality are of nearly the same PSNR values. There are a lot of corrupted edges in the JPEG compressed results, which badly affected visual quality. For AGFT compressed results, although there is also some distortion around edges due to approximation, the edges in synthesized views still look very sharp.

(a) cones (b) Dolls (c) Rocks

Fig. 3.11. Sub-regions of the synthesized views by original color images and AGFT / JPEG compressed depth images, respectively, where the PSNR quality of depth images compressed by AGFT and JPEG are almost the same. The first row views are by AGFT compressed depth images, while the second row views are by JPEG compressed depth images. Rocks is enlarged to be more distinct. (We strongly recommend readers to read an electronic version of this thesis to distinguish the differences of the color images.)

3.5 Conclusion

Efficient coding of depth image is essential for decoder-side virtual view synthesis via DIBR. Existing works either employ fixed transforms like DCT that blur depth image's sharp edges at low rates, or use edge-adaptive transforms that require

lossless coding of detected edges as side information, which accounts for a large share of the bit budget at low rates. In this chapter, we propose to first approximate object contours to lower the edge coding cost, then use edge-adaptive GFT for block-based coding so that sharp edges are preserved and the PWS characteristic is maintained. Experiments show noticable performance gain over previous coding schemes using either fixed transform (*e.g.*, JPEG codec) or edge-adaptive transform (*e.g.*, GFT) with lossless coding of detected contours.

3.6 Appendix 3.A: Proof of Lemma 1

It is straightforward that in an MSE manner, the value of one new longer constant signal segment is the average of original constant pieces spanned by this new segment. Here we mainly want to prove that the M jump locations in approximated signal are coincident with original jump locations. Based on this property, the best M jump locations can be identified by a DP algorithm with Eq. (3.2).

Denote by $\mathbf{x}$ a PWC signal with N jump locations $\{x_1, \cdots, x_N\}$ and $N+1$ constant pieces $\{c_1, \cdots, c_{N+1}\}$. For notational convenience, we will include the starting and ending points of signal $\mathbf{x}$ into the set of jump locations as x_0 and x_{N+1}. We also define the length of each constant piece as Δx_i, where $\Delta x_i = x_i - x_{i-1}$, for $i \in \{1, N+1\}$. Considering a PWC signal $\mathbf{x}'$ with M jumps that best approximates $\mathbf{x}$, where the jump locations are labeled as $\{y_1, \cdots, y_M\}$ and the $M+1$ constant pieces are labeled as $\{f_1, \cdots, f_{M+1}\}$.

Given the i-th jump location y_i of $\mathbf{x}'$, suppose on signal $\mathbf{x}$, the closest jump location to the right of y_i is x_{t_i}, where $t_i \in \{1, N+1\}$. We define $k_i = x_{t_i} - y_i$ as an integer showing how far y_i is away from x_{t_i}, and $k_i \in (0, \Delta x_{t_i}]$. See Fig. 3.12 for an illustration. In the following, we want to find the best value of k_i that minimizes the total approximation MSE.

Considering two neighboring jump locations of $\mathbf{x}'$: $y_i = x_{t_i} - k_i$ and $y_{i-1} = x_{t_{i-1}} - k_{i-1}$, where $y_{i-1} < y_i$. In an MSE manner, the new constant piece f_i is the

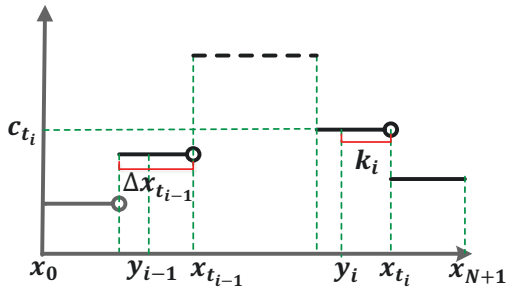


Fig. 3.12. An illustration of original and approximated jump locations.

average of the original signal $\mathbf{x}$ from y_{i-1} to y_i. Thus, f_i can be computed as

$$\begin{aligned} f_i &= \frac{\sum_{j=t_{i-1}+1}^{t_i} c_j \Delta x_j - k_i c_{t_i} + k_{i-1} c_{t_{i-1}}}{\sum_{j=t_{i-1}+1}^{t_i} \Delta x_j - k_i + k_{i-1}} \\ &\triangleq \frac{q_i - k_i c_{t_i} + k_{i-1} c_{t_{i-1}}}{p_i - k_i + k_{i-1}} \end{aligned} \tag{3.5}$$

where $q_i = \sum_j c_j \Delta x_j$ is the sum of $\mathbf{x}$ from jump location $x_{t_{i-1}}$ to x_{t_i}, and $p_i = \sum_j \Delta x_j = x_{t_i} - x_{t_{i-1}}$ is the length. Here, we use $\sum_j$ to stand for $\sum_{j=t_{i-1}+1}^{t_i}$ for simplicity. By using f_i to approximate the original signal $\mathbf{x}$ from y_{i-1} to y_i, the induced MSE d_i of this new segment can be computed as

$$d_i = \sum_j (f_i - c_j)^2 \Delta x_j - k_i (f_i - c_{t_i})^2 + k_{i-1} (f_i - c_{t_{i-1}})^2. \tag{3.6}$$

And the total MSE D of approximating $\mathbf{x}$ with $\mathbf{x}'$ becomes

$$D = \sum_{i=1}^{M+1} d_i. \tag{3.7}$$

To check which k_i gives the minimum value of D, we do a partial derivative of D with respect to k_i. It is easy to see that k_i is only related to MSE d_i and d_{i+1}. Hence, we have

$$\frac{\partial D}{\partial k_i} = \frac{\partial d_i}{\partial k_i} + \frac{\partial d_{i+1}}{\partial k_i}. \tag{3.8}$$

We calculate $\frac{\partial d_i}{\partial k_i}$ first, as follows:

$$\frac{\partial d_i}{\partial k_i} = 2\sum_j (f_i - c_j)\Delta x_j \frac{\partial f_i}{\partial k_i} - (f_i - c_{t_i})^2 - [2k_i(f_i - c_{t_i}) - 2k_{i-1}(f_i - c_{t_{i-1}})]\frac{\partial f_i}{\partial k_i} \tag{3.9}$$

$$= 2\left[(p_i - k_i + k_{i-1})f_i - (q_i + k_i c_{t_i} - k_{i-1}c_{t_{i-1}})\right]\frac{\partial f_i}{\partial k_i} - (f_i - c_{t_i})^2$$

$$= -(f_i - c_{t_i})^2,$$

where the last equality is due to the fact that $(p_i - k_i + k_{i-1})f_i - (q_i + k_i c_{t_i} - k_{i-1}c_{t_{i-1}}) = 0$, which is from Eq. (3.5).

Similarly, we can derive

$$\frac{\partial d_i}{\partial k_{i-1}} = (f_i - c_{t_{i-1}})^2, \tag{3.10}$$

and by substituting i with $i + 1$, we have

$$\frac{\partial d_{i+1}}{\partial k_i} = (f_{i+1} - c_{t_i})^2. \tag{3.11}$$

Thus the first-order derivative of D is expressed as

$$\frac{\partial D}{\partial k_i} = (f_{i+1} - c_{t_i})^2 - (f_i - c_{t_i})^2. \tag{3.12}$$

We now look at the second-order derivative of D with respected to k_i. We first check the partial derivative of $P \triangleq (f_i - c_{t_i})^2$ with respect to k_i as

$$\frac{\partial P}{\partial k_i} = 2(f_i - c_{t_i})\frac{\partial f_i}{\partial k_i}. \tag{3.13}$$

Since

$$f_i - c_{t_i} = \frac{q_i - p_i c_{t_i} + k_{i-1}(c_{t_{i-1}} - c_{t_i})}{p_i - k_i + k_{i-1}}, \tag{3.14}$$

and

$$\frac{\partial f_i}{\partial k_i} = \frac{q_i - p_i c_{t_i} + k_{i-1}(c_{t_{i-1}} - c_{t_i})}{(p_i - k_i + k_{i-1})^2}, \tag{3.15}$$

and $p_i - k_i + k_{i-1}$ is the segment length between y_{i-1} and y_i, which is always positive, we can get that $\frac{\partial P}{\partial k_i} \geq 0$. With a similar derivation process, we have $\frac{\partial Q}{\partial k_i} \leq 0$, where $Q \triangleq (f_{i+1} - c_{t_i})^2$. Hence, we can conclude that $\frac{\partial^2 D}{\partial^2 k_i} \leq 0$, which means that given $k_i \in (0, \Delta x_{t_i}]$, the best k_i that results in the minimum D is when $k_i = \Delta x_{t_i}$ (which means $y_i = x_{t_i - 1}$) or when $k_i = 0$ (which means $y_i = x_{t_i}$).

Above discussion proves that when considering to minimize the approximation MSE, each jump location on the approximated signal $\mathbf{x}'$ should be coincident with one of the original jump locations on $\mathbf{x}$. Thus, the M jump locations of $\mathbf{x}'$ should be a subset of the original N jump locations of $\mathbf{x}$. Hence Lemma 1 is proved. □

Chapter 4

Contour Approximation and Depth Image Coding: A RD Optimal Algorithm

In previous chapter, a greedy algorithm is proposed to approximate object contours to reduce depth coding cost, where the edge direction with the largest estimated probability computed by AEC is selected every time. However, since depth images are usually used for synthesizing virtual viewpoint at the decoder side, it is necessary to consider the resulting synthesized view distortion caused by depth contour approximation. Hence, in this chapter, we pursue a new paradigm in depth image coding for color-plus-depth representation of a 3D scene: we pro-actively simplify complex object shapes in a depth and color image pair to lower depth coding cost, at a controlled increase in synthesized view distortion. This means that as the bit budget becomes stringent, actual shapes of physical objects in the scene are simplified, but rendering of the objects remains sharp and natural for human perception.

Specifically, we execute our proposed object shape approximation in our color-plus-depth coding system as follows (See Fig. 4.1 for an overview). Given an input color-plus-depth image pair from the same viewpoint, we first approximate object contours via a DP algorithm to optimally trade off the cost of contours coded using AEC and the synthesized view distortion induced due to contour approximation. The color and depth images are then modified according to the approximated object contours to ensure inter-view consistency. Finally, the modified depth

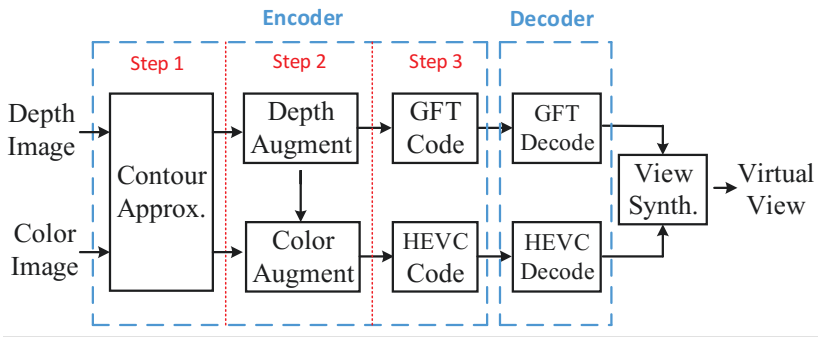


Fig. 4.1. Overview of our proposed color-plus-depth image coding system.

image is coded using a contour-adaptive image codec based on GFT for edge preservation [18], and the modified color image is coded by HEVC intra. At the decoder, the decoded depth and color images are then used for free-view synthesis via DIBR.

To measure the induced synthesized view distortion, we propose to use 3DSwIM [5]—a quality metric tailored specifically for DIBR-synthesized images. However, the complex definition of 3DSwIM makes it too expensive to directly quantify distortion due to contour approximation. In response, we mathematically derive a local distortion proxy that serves as an upper bound of 3DSwIM with reduced inter-dependencies across pixel rows to ease optimization.

Given the contour coding rate by AEC introduced in Section 3.3.1, in this Chapter, we first discuss synthesized view distortion induced by contour approximation. Based on the contour coding rate and distortion, an RD optimal method for depth contour approximation is proposed. We then introduce depth and color image coding using the approximated object contours. Finally, experiments and conclusion are presented. [1]

4.1 Distortion: A Proxy of 3DSwIM

Given that contour coding rate can be estimated by AEC, we now consider how to quantify distortion due to contour approximation. We first review 3DSwIM [5],

[1] A version of this chapter has been accepted to *IEEE Transactions on Circuits and Systems for Video Technology, August, 2017*.An arxiv version can be find on: https://arxiv.org/abs/1612.07872.

a new metric designed specifically to measure visual quality distortion of virtual view images synthesized via DIBR [2]. To control the extent of errors introduced during contour approximation, we need a notion of distortion stemming from the resulting geometric errors. Unfortunately, the 3DSwIM metric is difficult to optimize directly. We thus propose a simpler proxy that serves as an upper bound of 3DSwIM.

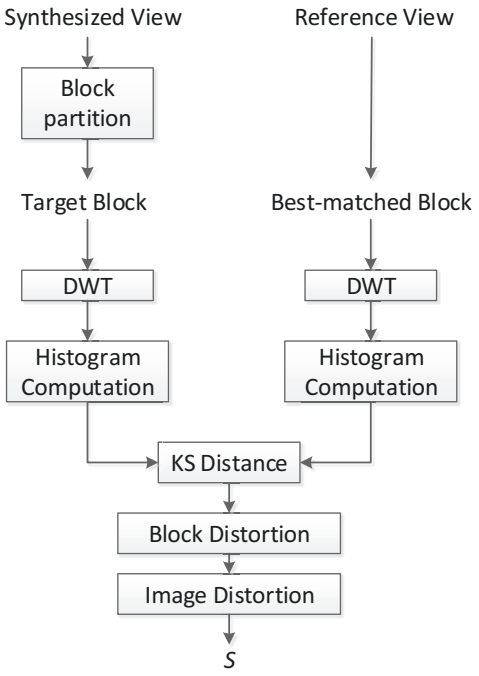


Fig. 4.2. Block diagram of 3DSwIM, from [5].

4.1.1 3D Synthesized View Image Quality Metric

The operations to compute 3DSwIM between a reference image and a DIBR-synthesized image is shown in the flow diagram in Fig. 4.2. First, the synthesized image is divided into B non-overlapping blocks, each of size $N \times N$. For a target block in the synthesized image, the best-matched block (using MSE as the matching criteria) is searched in the reference image within a search window of size $2W$ pixels in the horizontal direction, via a search algorithm like [88]. This horizon search procedure is based on the observation that objects may be horizontally shifted during DIBR's 3D warping of pixels in rectified images.

According to [89]'s analysis of DIBR distortions, the DIBR rendering errors are typically located near vertical edges of objects, and they appear as extraneous horizontal details. Hence in [5], the authors apply wavelet transform on each image block and measure the synthesized image degradation by analyzing the wavelet coefficients that describe the horizontal details. In our implementation of 3DSwIM, a multi-resolution 1D Haar wavelet transform (DWT) [90] is applied on each row of a target block to extract the horizontal details.

Since 3DSwIM is only concerned with horizontal image details, here we analyze the AC coefficients (details) of 1D DWT to compute a synthesized image quality measure. Performing a Haar DWT to each row of an $N \times N$ target block and its corresponding best-matched block in the reference image, each row generates an AC coefficient vector with length $(N-1)$. Stacking the coefficient vectors for different rows into a matrix, we construct two AC coefficient matrices $\mathbf{c}_s$ and $\mathbf{c}_o$ with size $N \times (N-1)$ corresponding to synthesized and reference blocks respectively.

We then construct the histograms $\mathbf{H}_s$ and $\mathbf{H}_o$ for each AC coefficient matrices: we divide the coefficients range into L bins of size τ and count how many coefficients fall into each bin, where τ is:

$$\tau = \frac{c_{max} - c_{min}}{L}, \tag{4.1}$$

$c_{max} = \max\{\mathbf{c}_o, \mathbf{c}_s\}$ and $c_{min} = \min\{\mathbf{c}_o, \mathbf{c}_s\}$ are the maximum and minimum values of the two AC coefficient matrices.

Finally, with the cumulative distribution functions (CDF) $\mathbf{F}_s$ and $\mathbf{F}_o$ of the histograms $\mathbf{H}_s$ and $\mathbf{H}_o$ for the synthesized and the best-matched reference blocks respectively, the block distortion D_b for block $b \in \{1, \cdots, B\}$ is defined as the *Kolmogorov-Smirnov* (KS)[91] distance between the two CDFs, *i.e.*

$$D_b = \max_{j \in \{1, \cdots, L\}} \left| \mathbf{F}_o(j) - \mathbf{F}_s(j) \right|. \tag{4.2}$$

The overall normalized image distortion d and the final image quality score S are

then computed as:

$$d = \frac{1}{D_0} \sum_{b=1}^{B} D_b, \text{ and } S = \frac{1}{1+d}, \tag{4.3}$$

where D_0 is a normalization constant. The score S ranges in the interval $[0, 1]$, where a lower distortion corresponds0 to a higher score and vice versa.

4.1.2 A Proxy of 3DSwIM

When approximating a contour in a pixel block, given a window of K previous edges $\{e_t, \ldots, e_{t-K+1}\}$, it is desirable to evaluate the effects on aggregate rate and distortion of choosing an edge e_{t+1} *locally* without considering edges *globally* outside the window. The complex definition of 3DSwIM, however, makes this difficult. When computing distortion between a target and a reference block, 3DSwIM identifies the maximum difference between the CDFs of the two corresponding histograms of wavelet coefficients. Hence the effects on distortion due to changes in one edge e_{t+1} are not known until all other edges are decided, which together determine the bin populations of wavelet coefficients in the histogram. This inter-dependency among edges in a block makes it difficult to minimize distortion systematically without exhaustively searching through all possible edge sequences.

In response, we propose a simple proxy to mimic 3DSwIM during contour approximation. Specifically, we first prove that the sum of local distortions (for individual pixel rows) is an upper bound of global 3DSwIM (for the entire pixel block). Thus, we can minimize the sum of local distortions to minimize the upper bound of the global distortion. Second, by assuming that the CDF of wavelet coefficients for a pixel row follows a certain model, we compute the local distortion as the maximum difference between two CDFs, which reduces to a simple function of respective model parameters. The sum of local distortions in a block is then used as a distortion proxy for contour approximation.

4.1.2.1 Local Distortion Upper Bound

In 3DSwIM, the 1D wavelet detail coefficients for different pixel rows in an $N \times N$ block are collected and sorted into L bins to construct a histogram. $\mathbf{F}_s$ and $\mathbf{F}_o$ are the CDFs of the histograms for the synthesized and the best-matched reference blocks respectively. Suppose that instead we divide coefficients in each row into the same L bins, and define the cumulative distribution of the L bins for each row i as $\mathbf{f}_s^i$ and $\mathbf{f}_o^i$, for the synthesized and best-matched blocks respectively. We see easily that $\mathbf{F}_o = \sum_{i=1}^{N} \mathbf{f}_o^i$ and $\mathbf{F}_s = \sum_{i=1}^{N} \mathbf{f}_s^i$. Thus the block distortion in (4.2) can be rewritten as:

$$D_b = \max_{j \in \{1,\cdots,L\}} \left| \sum_{i=1}^{N} \mathbf{f}_o^i(j) - \sum_{i=1}^{N} \mathbf{f}_s^i(j) \right|. \tag{4.4}$$

To derive an upper bound for D_b in (4.4), we see that

$$\begin{aligned} D_b &\leq \max_{j \in \{1,\cdots,L\}} \sum_{i=1}^{N} \left| \mathbf{f}_o^i(j) - \mathbf{f}_s^i(j) \right| \\ &\leq \sum_{i=1}^{N} \max_{j \in \{1,\cdots,L\}} \left| \mathbf{f}_o^i(j) - \mathbf{f}_s^i(j) \right| \\ &\triangleq \sum_{i=1}^{N} D_b^i \end{aligned} \tag{4.5}$$

where D_b^i denotes the maximum difference between the CDFs of the coefficient histograms for pixel row i; we call this the *row distortion*. The equality can be reached when there is one bin j^* that has the largest difference for all the N rows and $\mathbf{f}_o^i(j^*)$ is always no smaller or no larger than $\mathbf{f}_s^i(j^*)$, *i.e.*,

$$\begin{aligned} &j^* = \arg \max_{j \in \{1,\cdots,L\}} \left| \mathbf{f}_o^i(j) - \mathbf{f}_s^i(j) \right|, \quad \forall\, i \in \{1, \cdots, N\}, \text{ and} \\ &\mathbf{f}_o^i(j^*) \leq \mathbf{f}_s^i(j^*) \text{ or } \mathbf{f}_o^i(j^*) \geq \mathbf{f}_s^i(j^*), \quad \forall\, i \in \{1, \cdots, N\}. \end{aligned} \tag{4.6}$$

Hence, we argue that $\sum_{i=1}^{N} D_b^i$ is the supremum of the block distortion D_b. We can now write our distortion proxy $\hat{D}_b$ that mimics 3DSwIM as the sum of individual

row distortions, *i.e.*

$$\hat{D}_b = \sum_{i=1}^{N} D_b^i. \tag{4.7}$$

4.1.2.2 A Model for Row Distortion

We now investigate how to compute row distortion D_b^i efficiently. We first model the wavelet coefficients of a color pixel row using a *Laplace distribution* similar to [92]. We then compute the row distortion as a function of the model parameters.

In [92], Wouwer *et al.* show that for a color image block with size 64×64, the histogram of the AC wavelet transform coefficients on each subband can be modelled by a generalized Gaussian density function:

$$\begin{aligned} f_{\sigma,\rho}(c) &= \frac{\rho}{2\sigma \cdot \Gamma(\frac{1}{\rho})} \cdot e^{-(\frac{|c|}{\sigma})^{\rho}},\ c \in \mathbb{R} \\ \Gamma(x) &= \int_0^{\infty} e^{-t} t^{x-1} dt,\ x > 0 \end{aligned} \tag{4.8}$$

where σ is the variance, and ρ is a shape parameter ($\rho = 2$ or 1 for a Gaussian or Laplace distribution). Histogram $f_{\sigma,\rho}(c)$ can also be interpreted as the probability of a coefficient taking on value c.

Inspired by [92], in our work we assume also that the AC wavelet coefficients on a color pixel row follow the same distribution (4.8). For simplicity, we assume further that $\rho = 1$, *i.e.*, the AC wavelet coefficients follow a Laplace distribution with parameter σ, where

$$f_{\sigma}(c) = \frac{1}{2\sigma} \cdot e^{-\frac{|c|}{\sigma}},\ c \in \mathbb{R} \tag{4.9}$$

since $\Gamma(1) = 1$. In practice, given M observed AC wavelet coefficients $\{c_1, \cdots, c_M\}$, the best estimate of model parameter σ using the *maximum likelihood estimation* (MLE) [93] criteria is

$$\sigma = \frac{1}{M} \sum_{i=1}^{M} |c_i|. \tag{4.10}$$

The derivation for (4.10) is in Appendix 4.A.

Assuming that the AC wavelet coefficients of a pixel row on the synthesized and its best-matched reference blocks are both Laplace distributions with respective parameters σ_s and σ_o, we now compute the row distortion D_b^i as a function of σ_s and σ_o. Since D_b^i computes the maximum difference between the CDFs of two histograms, we first write the CDF of $f_\sigma(c)$ as:

$$\begin{aligned} F_\sigma(c) &= \int_{-\infty}^{c} f_\sigma(x)\mathrm{d}x \\ &= \begin{cases} \frac{1}{2}\exp\left\{\frac{c}{\sigma}\right\}, & \text{if } c < 0 \\ 1 - \frac{1}{2}\exp\left\{-\frac{c}{\sigma}\right\}, & \text{if } c \geq 0\ . \end{cases} \end{aligned} \tag{4.11}$$

Then we define $g(c) = |F_{\sigma_o}(c) - F_{\sigma_s}(c)|$, and D_b^i is equivalent to finding the maximum value of $g(c)$, $c \in \mathbb{R}$.

For simplicity, we define $\sigma_{\texttt{max}} = \max\{\sigma_o, \sigma_s\}$ and $\sigma_{\texttt{min}} = \min\{\sigma_o, \sigma_s\}$. When $\sigma_{\texttt{max}} = \sigma_{\texttt{min}}$, $g(c) = 0$. When $\sigma_{\texttt{max}} > \sigma_{\texttt{min}}$, ignoring the constant weight $1/2$, we get

$$g(c) = \begin{cases} \exp\left\{\frac{c}{\sigma_{\texttt{max}}}\right\} - \exp\left\{\frac{c}{\sigma_{\texttt{min}}}\right\}, & \text{if } c < 0 \\ \exp\left\{-\frac{c}{\sigma_{\texttt{max}}}\right\} - \exp\left\{-\frac{c}{\sigma_{\texttt{min}}}\right\}, & \text{if } c \geq 0\ . \end{cases} \tag{4.12}$$

The first-order derivative function of $g(c)$ becomes

$$g'(c) = \begin{cases} \frac{1}{\sigma_{\texttt{max}}}\exp\left\{\frac{c}{\sigma_{\texttt{max}}}\right\} - \frac{1}{\sigma_{\texttt{min}}}\exp\left\{\frac{c}{\sigma_{\texttt{min}}}\right\}, & \text{if } c < 0 \\ \frac{1}{\sigma_{\texttt{min}}}\exp\left\{\frac{-c}{\sigma_{\texttt{min}}}\right\} - \frac{1}{\sigma_{\texttt{max}}}\exp\left\{\frac{-c}{\sigma_{\texttt{max}}}\right\}, & \text{if } c \geq 0. \end{cases} \tag{4.13}$$

When we set $g'(c) = 0$, we get the optimal c as:

$$c = \begin{cases} \frac{\sigma_{\texttt{max}}\sigma_{\texttt{min}}}{\sigma_{\texttt{max}} - \sigma_{\texttt{min}}} \ln \frac{\sigma_{\texttt{min}}}{\sigma_{\texttt{max}}} \triangleq c^*, & \text{if } c < 0 \\ -c^*, & \text{if } c \geq 0\ . \end{cases} \tag{4.14}$$

We thus conclude that $g(c^*)$ is a maximum for $c < 0$ and $g(-c^*)$ is a maximum for

$c \geq 0$. Since $g(c^*) = g(-c^*)$, the maximum value of $g(c)$ becomes

$$g_{\mathtt{max}} = g(c^*) = \left(\frac{\sigma_{\mathtt{min}}}{\sigma_{\mathtt{max}}}\right)^{\frac{\sigma_{\mathtt{min}}}{\sigma_{\mathtt{max}}-\sigma_{\mathtt{min}}}} - \left(\frac{\sigma_{\mathtt{min}}}{\sigma_{\mathtt{max}}}\right)^{\frac{\sigma_{\mathtt{max}}}{\sigma_{\mathtt{max}}-\sigma_{\mathtt{min}}}}. \tag{4.15}$$

In summary, for a pixel row in the synthesized and best-matched blocks with respective histograms f_{σ_s} and f_{σ_o}, the row distortion D_b^i can be computed as:

$$\begin{aligned} D_b^i &= \langle f_{\sigma_o}, f_{\sigma_s} \rangle \\ &\triangleq \begin{cases} \left(\frac{\sigma_{\mathtt{min}}}{\sigma_{\mathtt{max}}}\right)^{\frac{\sigma_{\mathtt{min}}}{\sigma_{\mathtt{max}}-\sigma_{\mathtt{min}}}} - \left(\frac{\sigma_{\mathtt{min}}}{\sigma_{\mathtt{max}}}\right)^{\frac{\sigma_{\mathtt{max}}}{\sigma_{\mathtt{max}}-\sigma_{\mathtt{min}}}}, & \text{if } \sigma_{\mathtt{max}} > \sigma_{\mathtt{min}} \\ 0, & \text{if } \sigma_{\mathtt{max}} = \sigma_{\mathtt{min}} \end{cases} \end{aligned} \tag{4.16}$$

where $\sigma_{\mathtt{max}} = \max\{\sigma_o, \sigma_s\}$ and $\sigma_{\mathtt{min}} = \min\{\sigma_o, \sigma_s\}$.

Together with (4.7), our proposed distortion proxy that mimics 3DSwIM becomes

$$\hat{D}_b = \sum_{i=1}^{N} \langle f_{\sigma_o^i}, f_{\sigma_s^i} \rangle, \tag{4.17}$$

where σ_s^i and σ_o^i are the respective Laplace distribution parameters for coefficients on the pixel row i for the synthesized and best-matched blocks.

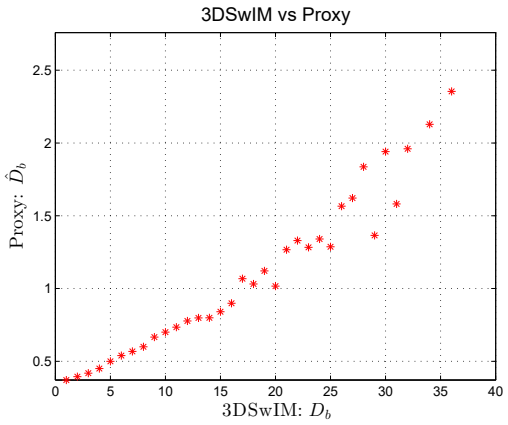


Fig. 4.3. x-axis: block distortion calculated by 3DSwIM; y-axis: mean value of distortion calculated by our proposed proxy

To verify the accuracy of our distortion proxy, we tested 17 image sequences

from the Middleburry dataset[2]. Virtual views are synthesized by a simple implementation of 3D warping [2] and each image is divided into around 600 blocks with size 16×16 for distortion computation. The result is shown in Fig. 4.3, in which the x-axis is D_b by 3DSwIM and the y-axis is the mean value of our proxy $\hat{D}_b$ in the corresponding blocks. This result clearly shows a positive linear trend between D_b and $\hat{D}_b$, demonstrating the effective approximation of our proposed distortion proxy.

4.2 RD Optimal Object Contour Approximation

Given the AEC-based contour coding and our proposed distortion proxy discussed in Sections 3.3.1 and 4.1 respectively, we now discuss how to approximate a detected contour.

4.2.1 Dividing Contour into Segments

When approximating a contour detected in a depth image, we first divide the contour into *segments*, where each segment is composed of edges with only two *non-opposite* directions: $\{\downarrow, \rightarrow\}$, $\{\downarrow, \leftarrow\}$, $\{\uparrow, \leftarrow\}$ or $\{\uparrow, \rightarrow\}$. We then approximate each segment separately. We require each approximated segment to start and end at the same locations as the original segment. Given that edges in a segment can only take non-opposite directions, it implies that the length T of the approximated segment must be the same as the original. Thus the search space of candidates is only $\binom{T}{V}$, where V ($\leq T$), is the number of vertical edges ($\downarrow$ or $\uparrow$) in the original segment. To efficiently search for an RD-optimal approximated segment within this space, we propose to use a DP algorithm. Finally, the approximated segments are combined to form an approximated contour.

4.2.2 DP Algorithm to Approximate One Segment

4.2.2.1 Efficient Computation of the Distortion Proxy

We first discuss how to efficiently compute our distortion proxy during segment approximation. Our distortion proxy Eq. (4.17) is computed by comparing

[2]http://vision.middlebury.edu/stereo/data/

a pixel block in a synthesized view image with a best-matched block in the reference image. To avoid laboriously augmenting the input color/depth image pair and synthesizing a virtual view image to compute distortion for each segment approximation, we propose to use *only the input color image* and a simple *shifting window* procedure to compute our distortion proxy. We describe this next.

We first divide the input color image into non-overlapping $N \times N$ pixel blocks. The block distortion—the sum of row distortions according to our proxy of Eq. (4.17)—is computed for blocks that contain parts of the segment. See Fig. 4.4 for an illustration of blocks of a color image containing contour segments (in green). When a segment is approximated and altered, *vertical edges* in the original segment are horizontally shifted. As an example, in Fig. 4.5, the vertical edge e_3^o in the original blue segment is shifted left by one pixel to e_4 in the approximated red segment.

At a given pixel row, assuming that the contained vertical edge has horizontally shifted, we compute the resulting row distortion using a shifting window procedure as follows. Suppose that an original vertical edge starting from 2D coordinate (p^o, q^o) (specifying respective row and column indices) is horizontally shifted by k pixels (positive/negative k means shifting to the right/left) to a new vertical edge with starting point (p^o, q), where $k = q - q^o$. Delimiting the pixel row in the original block with an N-pixel window, we shift the window by $-k$ pixels to identify a new set of N pixels that represent the pixels in the corresponding block of the synthesized view after segment modification.

An example of shifting window is illustrated in Fig. 4.4. One pixel row (black) in an original 8×8 block contains the pixel set $\mathbf{u} = \{I_1, \cdots, I_8\}$. During segment approximation, the original vertical edge between pixels I_4 and I_5 shifts *left* by 2 pixels ($k = -2$) to a new edge. As a result, we shift the delimiting window *right* by 2 pixels, and identify a different set of eight pixels, $\mathbf{v} = \{I_3, \cdots, I_{10}\}$. We see that the original edge in the shifted window is located two pixels from the left, and the new edge in the original window is also located two pixels from the left. Because of this alignment, we can simply use window $\mathbf{v}$ as a representation of the

pixel row after segment approximation, instead of the original pixels $\mathbf{u}$ plus image augmentation due to the vertical edge shift.

We can now compute the row distortion using (4.16) using two sets of N pixels delimited by the window before and after the shift operation. Because 3DSwIM only searches for a best-matched reference block within a window of size $2W$ pixels, when $|k| > W$, pixels in the shifted window no longer well represent the best-matched pixel row. We thus set the distortion to infinity in this case to signal a violation. We can now write the row distortion $d_{p^o}(q^o, q)$ induced by a horizontally shifted vertical edge from start point (p^o, q^o) to (p^o, q) for the p^o-th pixel row as

$$d_{p^o}(q^o, q) = \begin{cases} \langle f_{\sigma_\mathbf{u}}, f_{\sigma_\mathbf{v}} \rangle, & \text{if } |q - q^o| \leq W \\ \infty, & \text{otherwise} \end{cases} \tag{4.18}$$

where $\sigma_\mathbf{u}$ and $\sigma_\mathbf{v}$ are the respective Laplace distribution parameters computed using pixels in the original and the shifted windows $\mathbf{u}$ and $\mathbf{v}$. Continuing with the example in Fig. 4.5, $d_2(3, 2)$ is the distortion of edge e_3^o (starting from $(2, 3)$) being shifted to edge e_4 (starting from $(2, 2)$).

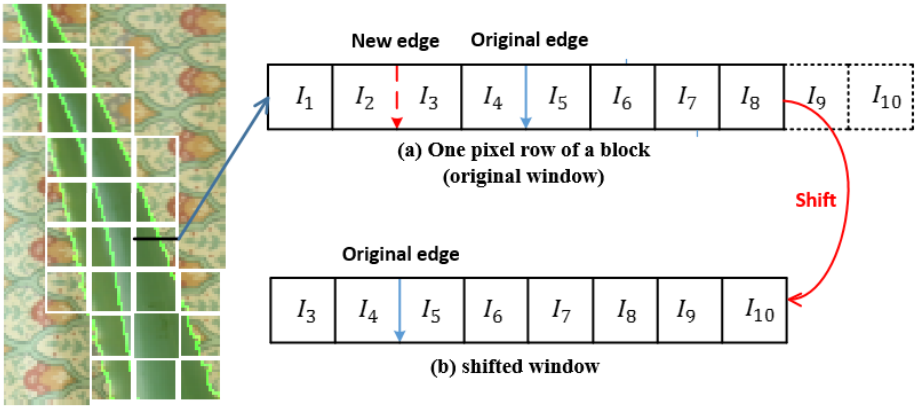


Fig. 4.4. An example of shifting window. The detected contours are shown in green in the color image. The white blocks contain part of contour segments. The original edge crossing one pixel row (black) in a block is shifted *left* by 2 pixels to a new edge. We shift the original window *right* by 2 pixels and identify the shifted window.

4.2.2.2 DP Algorithm

We next design a DP algorithm to approximate a contour segment composed of edges with directions $\{\downarrow, \leftarrow\}$. The algorithms for segments composed of edges with other different non-opposite direction pairs are similar, and thus are left out for brevity. We first define distortion and rate terms. Then we define a DP recursive equation based on Lagrangian relaxation.

4.2.2.2.1 Problem Formulation Denote by $\mathcal{E}^o$ and $\mathcal{E}$ the original and approximated segments respectively, and by T the length of $\mathcal{E}^o$. Further, we define (p_1, q_1) as the 2D coordinate of the first edge's start point. Given V vertical edges with direction $\downarrow$ in $\mathcal{E}^o$, the coordinate of the last edge's end point can be computed as $(p_{T+1}, q_{T+1}) = (p_1 + V, q_1 - (T - V))$. Given our constraint that $\mathcal{E}$ must start and end at the same locations as there in $\mathcal{E}^o$, the search space $\mathcal{S}$ for $\mathcal{E}$ is restricted to the rectangle region with opposite corners at (p_1, q_1) and (p_{T+1}, q_{T+1}). As an example, the original blue segment in Fig. 4.5 has length $T = 6$ and $V = 4$, with start point $(p_1, q_1) = (1, 4)$ and end point $(p_7, q_7) = (5, 2)$. The search space is the green rectangle region.

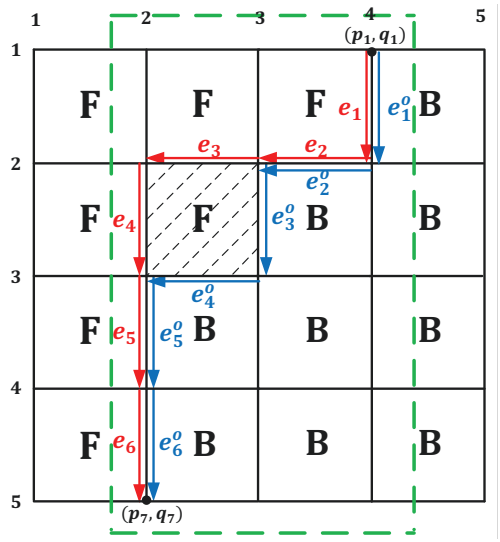


Fig. 4.5. An example of segment approximating. The original segment $\mathcal{E}^o = \{e_1^o, \cdots, e_6^o\}$ separating foreground (F) and background (B), where the approximated segment $\mathcal{E} = \{e_1, \cdots, e_6\}$. The vertical edge e_4 is shifted by edge e_3^o. The pixel with the top left corner $(2, 2)$ need to be altered after approximation.

Given that the row distortion is induced by a horizontally shifted vertical edge, the total distortion induced by an approximated segment is the sum of row distortions induced by all shifted vertical edges. Using the defined row distortion $d_{p^o}(q^o, q)$ for the p^o-th row, given an approximated segment $\mathcal{E} = \{e_T, \cdots, e_1\}$ with two non-opposite directions $\{\downarrow, \leftarrow\}$, the total distortion for approximating $\mathcal{E}^o$ with $\mathcal{E}$ is:

$$D(\mathcal{E}, \mathcal{E}^o) = \sum_{e_i \in \mathcal{E} | e_i = \downarrow} d_{p_i^o}(q_i^o, q_i) \tag{4.19}$$

where (p_i^o, q_i) is the 2D coordinate of edge e_i's starting point in $\mathcal{E}$, and q_i^o is the column index of a vertical edge on $\mathcal{E}^o$ crossing the p_i^o-th pixel row.

To approximate a segment $\mathcal{E}^o$, we solve the following RD cost function:

$$\min_{\mathcal{E}} \; D(\mathcal{E}, \mathcal{E}^o) + \lambda R(\mathcal{E}) \tag{4.20}$$

where $R(\mathcal{E})$ is the coding rate of segment $\mathcal{E}$ using AEC edge coding (Section 3.3.1). Given a window of K previous edges $\mathbf{s}_t = \{e_{t-1}, \cdots, e_{t-K}\}$ and the computed probability $P(e_t|\mathbf{s}_t)$ based on AEC, we use the entropy to estimate the coding bits of edge e_t, *i.e.*, $r(e_t|\mathbf{s}_t) = -\log_2 P(e_t|\mathbf{s}_t)$ [3]. Thus the total segment coding rate becomes

$$R(\mathcal{E}) = \sum_{t=1}^{T} r(e_t|\mathbf{s}_t). \tag{4.21}$$

4.2.2.2.2 DP Algorithm Development We propose a DP algorithm to solve (4.20). First, we define $J(\mathbf{s}_t, p_t, q_t)$ as the recursive RD cost of a partial segment from the t-th edge to the last edge, given that the K previous edges are $\mathbf{s}_t \in \{\downarrow, \leftarrow\}^K$, and (p_t, q_t) is the coordinate of the t-th edge e_t's starting point. $J(\mathbf{s}_t, p_t, q_t)$ can be recursively solved by considering each candidate e_t inside the search space

[3] Given a segment, when $t \leq K$, we define $\mathbf{s}_t$ as the combination of the first $t-1$ edges in the current segment and the last $K - t + 1$ edges in the previous segment. As to the the first segment in a contour, when $t \leq K$, we define $\mathbf{s}_t = e_{t-1} \cup \mathbf{s}_{t-1}$ with $\mathbf{s}_1 = \emptyset$, and $P(e_1|\mathbf{s}_1) = 1/4$ since $e_1 \in \mathcal{A}^o$ with $|\mathcal{A}^o| = 4$, and $P(e_t|\mathbf{s}_t) = 1/3$ for $t > 1$, since $e_t \in \mathcal{A}$ with $|\mathcal{A}| = 3$.

$\mathcal{S}$ (the rectangle box):

$$J\left(\mathbf{s}_t, p_t, q_t\right) = \min_{e_t \in \left\{ \{\downarrow, \leftarrow\} | (p_{t+1}, q_{t+1}) \in \mathcal{S} \right\}} \Big\{ \lambda \cdot r(e_t | \mathbf{s}_t) + d_{p_t^o}(q_t^o, q_t) \cdot \delta_{\downarrow} + J\left(\mathbf{s}_{t+1}, p_{t+1}, q_{t+1}\right) \cdot \delta(p_{t+1}, q_{t+1}) \Big\} \tag{4.22}$$

where $p_t^o = p_t$ and $\delta_{\downarrow}$ is a binary indicator that equals 1 if $e_t = \downarrow$ and 0 otherwise. The value $\mathbf{s}_{t+1}$ is updated by combining e_t with the first $K-1$ elements in $\mathbf{s}_t$. The pixel position (p_{t+1}, q_{t+1}) equals $(p_t + 1, q_t)$ or $(p_t, q_t - 1)$ if $e_t = \downarrow$ or $e_t = \leftarrow$, respectively. Finally $\delta(p, q)$ indicates the termination condition, which is

$$\delta(p, q) = \begin{cases} 0, & \text{if } (p, q) = (p_{T+1}, q_{T+1}) \\ 1, & \text{otherwise} \end{cases} \tag{4.23}$$

Eq. (4.22) is considered a DP algorithm because there are overlapping sub-problems during recursion. Each time a sub-problem with the same argument is solved using (4.22), and the solution is stored as an entry into a DP table. Next time the same sub-problem is called, the previously computed entry is simply retrieved from the DP table and returned.

Starting from the first edge, the recursive call results in the minimum cost of the segment approximation. The corresponding segment $\mathcal{E}^* = \arg\min J\big(\mathbf{s}_1, p_1, q_1\big)$ is the new approximated segment.

4.2.2.3 Complexity of DP Algorithm

The complexity of a DP algorithm is upper-bounded by the size of DP table times the complexity to compute each table entry. The size of the DP table can be bounded as follows. The argument $\mathbf{s}_t$ in $J(\mathbf{s}_t, p_t, q_t)$ can take on $O(2^K)$ values. On the other hand, the argument (p_t, q_t)—the starting point of e_t—can take on $O(V \times (T - V))$ locations in search space $\mathcal{S}$. Thus the DP table size is $O(2^K V(T-V))$. To compute each DP table entry using Eq. (4.22), a maximum of two choices for e_t need to be tried, and each choice requires a sum of three terms. The second term involves row distortion $d_{p_t^o}(q_t^o, q_t)$, which itself can be computed and stored into a DP table of

size $O(VW)$, each entry computed in $O(N^2)$. Thus, the complexity of computing $d_{p_t^o}(q_t^o, q_t)$ is $O(VWN^2)$. Hence, the overall complexity of (4.22) is $O(2^K V(T - V) + VWN^2)$.

In our experiments, we set $K = 3$ as done in [58], $W = 10$ and $N = 16$. The number $T - V$ of horizontal edges in a segment is usually smaller than N^2. Thus the overall complexity of the algorithm can be simplified to $O(VWN^2)$.

4.2.3 Greedy Segment Merging

We next discuss how two consecutive segments can be merged into one to further reduce edge coding cost. Given the start point l_0 of the first segment and the end point l_2 of the second, we first delimit the feasible space of edges by the rectangle with opposing corners at l_0 and l_2. The original edges which are outside this feasible space are first projected to the closest side of the rectangle, so that edges in the feasible space become a new segment. This edge projection leads to edge shifting and thus induce distortion; we call this the *merge distortion*. We then execute the aforementioned DP algorithm on this new simplified segment to get a new minimum cost. If this cost plus the merge distortion is smaller than the cost sum of the original two segments, we merge the two original segments to this new simplified segment.

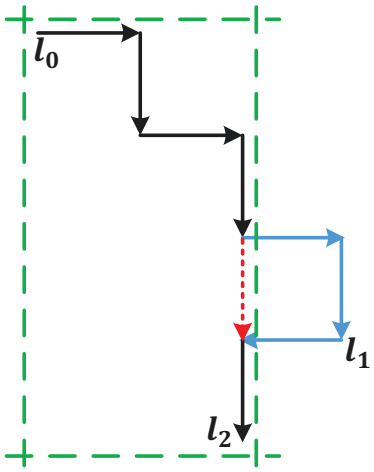


Fig. 4.6. An example of segment merging.

As illustrated in Fig. 4.6, a segment with directions $\{\rightarrow, \downarrow\}$ starts from point l_0 and ends at l_1. Another segment with directions $\{\leftarrow, \downarrow\}$ exists between l_1 to l_2.

The feasible space is marked as the green rectangle region. The original blue edges outside the feasible space are projected to the dash red edge, such that the new segment from l_0 to l_2 only contains two directions $\{\rightarrow, \downarrow\}$.

The procedure of approximating one contour using our proposed DP algorithm and greedy segment merging method is summarized in Algorithm 1.

Algorithm 1: Depth Contour Approximation

1: Divide a contour into M segments;
2: Given a λ, execute Eq. (4.22) on each segment to get the RD cost;
3: Greedily merge two neighbouring segments until there is no cost decrease;
4: Finally get M' segments, where $M' \leq M$.

4.3 Image Coding With Approximated Contours

Given a set of approximated contours in an image that are coded losslessly using AEC [58], we now describe how to code depth and color images at the encoder. We also discuss inter-view consistency issues.

4.3.1 Depth and Color Images Modification

Approximating object contours means modifying the geometry of a 3D scene. Accordingly, both depth and color images from the same viewpoint need to be modified to be consistent with the new geometry. Specifically, for the depth image, we exchange foreground and background depth pixel values across a modified edge, so that all foreground (background) depth values fall on the same side of an edge. In Fig. 4.5, the pixel with the top left corner index $(2, 2)$ needs to be replaced by a background pixel.

For the color image, pixels corresponding to the altered pixels in the depth image are removed and labeled as holes. An image inpainting algorithm [94] is then used to inpaint the identified holes, with a constraint that only existing background (foreground) pixels can be used to complete holes in the background (foreground).

Fig. 4.7 shows an example of contour approximation and image value modification for one depth and color image pair. The two images (a) and (e) are part of the original depth and color image pair for `Aloe`. Depth contours are approximated

by our proposed method with an increasing value of λ for images from (b) to (d). We alter depth pixel values, and the PWS characteristic is still preserved in the approximated depth images. Images (f) to (h) are the corresponding inpainted color images, which look natural along its neighboring foreground / background regions. The geometry structure in the altered depth and color image pairs are consistent with the approximated contours.

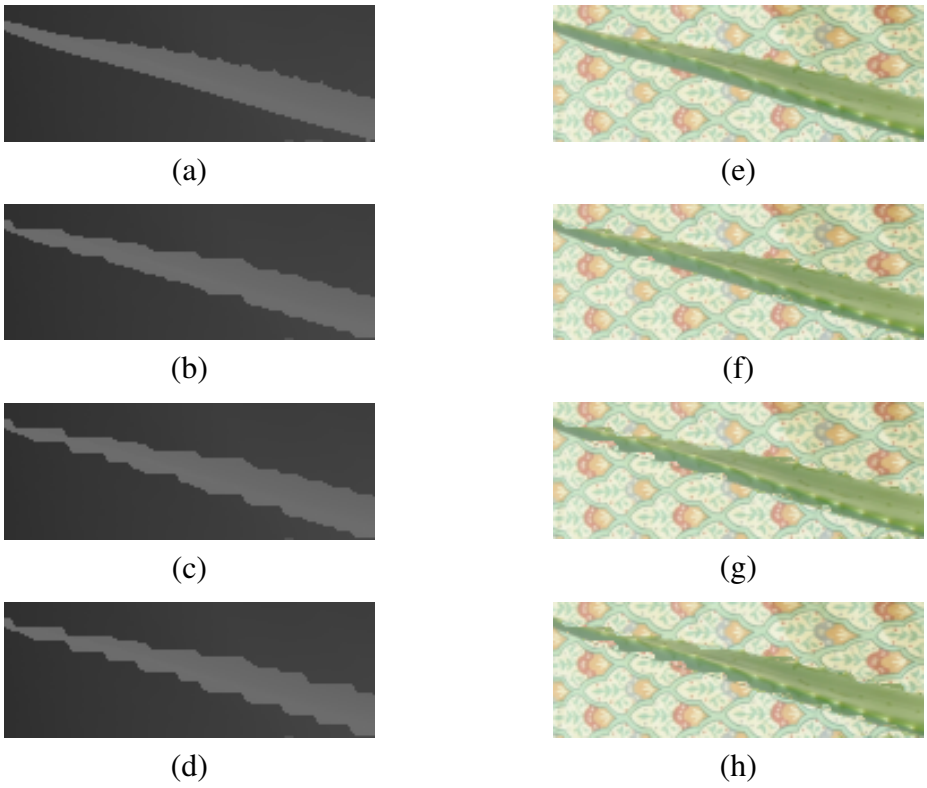


Fig. 4.7. Example for contour approximation and image value alteration.

4.3.2 Inter-View Consistency

Given that the encoder needs to encode two color-plus-depth image pairs of the same 3D scene (e.g., left and right views) for synthesis of an intermediate virtual view at the decoder, we can perform contour approximation in an inter-view consistent manner as follows.

First, we approximate object contours of the left view and augment the corresponding depth and color images. We then project the approximated and

modified left depth image to the right viewpoint via DIBR. With the projected depth image, we augment the original right depth and color image pair accordingly, such that the approximation on the left view is mapped to the right view. Finally, we approximate the augmented right depth and color images. To penalize the inter-view inconsistency when approximation happens on the right view, we add a penalty term to the row distortion (4.18), *i.e.* we define the row distortion for right view approximation as

$$d'_{p^o}(q^o, q) = d_{p^o}(q^o, q) + \rho|q - q^o|^2, \tag{4.24}$$

where $|q - q^o|$ takes an additional meaning that the shifted edge on the augmented right view is now inconsistent with the approximated left view. The weight parameter ρ is set very large to penalize the inter-view inconsistency. By doing so, the depth and color image pair on the right view is very likely consistent with that of the left view after contour approximation.

4.3.3 Edge-Adaptive Image Coding

The remaining task is to code the depth and color images, which have been modified accordingly after our contour approximation.

We first describe how edge-adaptive GFT [18] can be used to code the altered depth image. For each pixel block in the altered depth image, a 4-connected graph is constructed, where each node corresponds to a pixel. The edge weight between two neighbouring pixels is defined based on their weak or strong correlations. Given the constructed graph, the transform is the eigen-matrix of the graph Laplacian (see [18] for details). The edge weight assignment (deducible from the encoded contours) means filtering across sharp boundaries are avoided, preventing blurring of edges.

Since our work is mainly about approximating object contours, here we focus on how contour approximation influences the depth coding rate. For color images, we code them with a state-of-the-art image coding method—HEVC HM 15.0 [17].

4.4 Experimentation

4.4.1 Experimental Setup

We perform extensive experiments to validate the effectiveness of our proposed contour approximation method using color-plus-depth image sequences from the Middlebury dataset, `Undo Dancer` and `GFTly` [95]. View synthesis for `Undo Dancer` and `GTFly` is performed using VSRS software[4]. Due to the lack of camera information, DIBR for the Middlebury sequences is performed using a simple implementation of 3D warping [2].

For each image sequence, we first approximate the depth and color image pair with different values of λ based on our proposed approximation method. We then deploy the GFT based edge-adaptive depth coding [18] to code the approximated and altered depth image using different quantization parameters (QP), where the approximated contours are losslessly coded as SI. The approximated and inpainted color image from the same viewpoint is compressed by HEVC intra HM 15.0 [17]. Depth and color image pairs from different viewpoints but approximated by the same λ are then transmitted to the decoder for virtual view synthesis. The ground truth (reference image) is synthesized by the original depth and color images (no approximation and compression) for distortion computation. 3DSwIM with block size 16×16 is used to evaluate the quality of the synthesized views, where a higher score means a better quality. As an additional metric, we also evaluate synthesized view quality with PSNR. For the Middleburry sequences, we encode view 1 and view 5, and synthesize the intermediate virtual views 2 to 4. For `Undo Dancer` and `GFTly` sequences, view 1 and view 9 are encoded and the intermediate views 2 to 8 are synthesized. The average distortion of all the intermediate views is taken to evaluate the contour approximation performance. For each image sequence, the convex hull of all operational points represents the rate-score (RS) or RD performance of our proposed method.

[4] wg11.sc29.org/svn/repos/MPEG- 4/test/trunk/3D/view_synthesis/VSRS

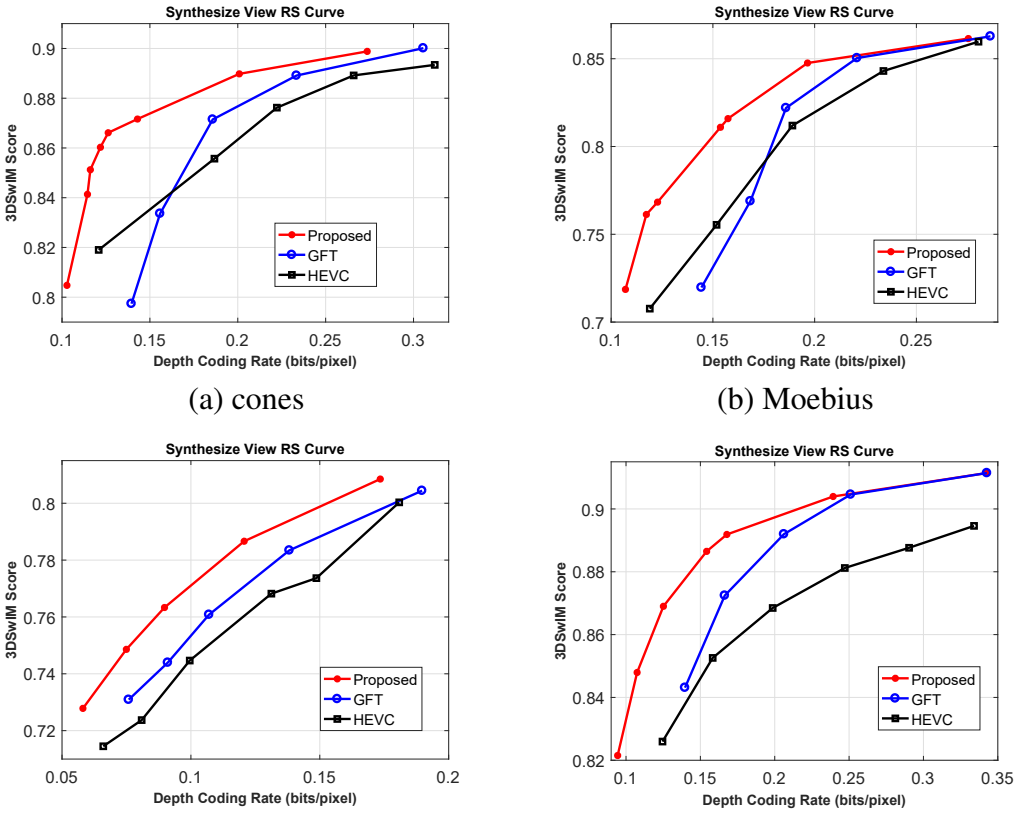


Fig. 4.8. Synthesized virtual view RS curves for cones, Moebius, Lampshade and Aloe, respectively.

4.4.2 Objective Results Compared to `MR-GFT` and `HEVC`

To demonstrate the efficiency of our proposed contour approximation method (termed as `Proposed`), we compare our method with the `MR-GFT` method (depth images are compressed by multi-resolution GFT [18] directly with the original detected contours that are losslessly coded). We also consider using HEVC intra to compress the original depth images (termed as `HEVC`). The corresponding color images for both `MR-GFT` and `HEVC` methods are also compressed by HEVC intra. Since here we want to assess how object contour approximation affects depth image coding efficiency, we only consider the depth images with almost the same coding rate by all these three methods. For the corresponding color images, we select an identical QP_C to compress them (meaning that color images coding rates are also

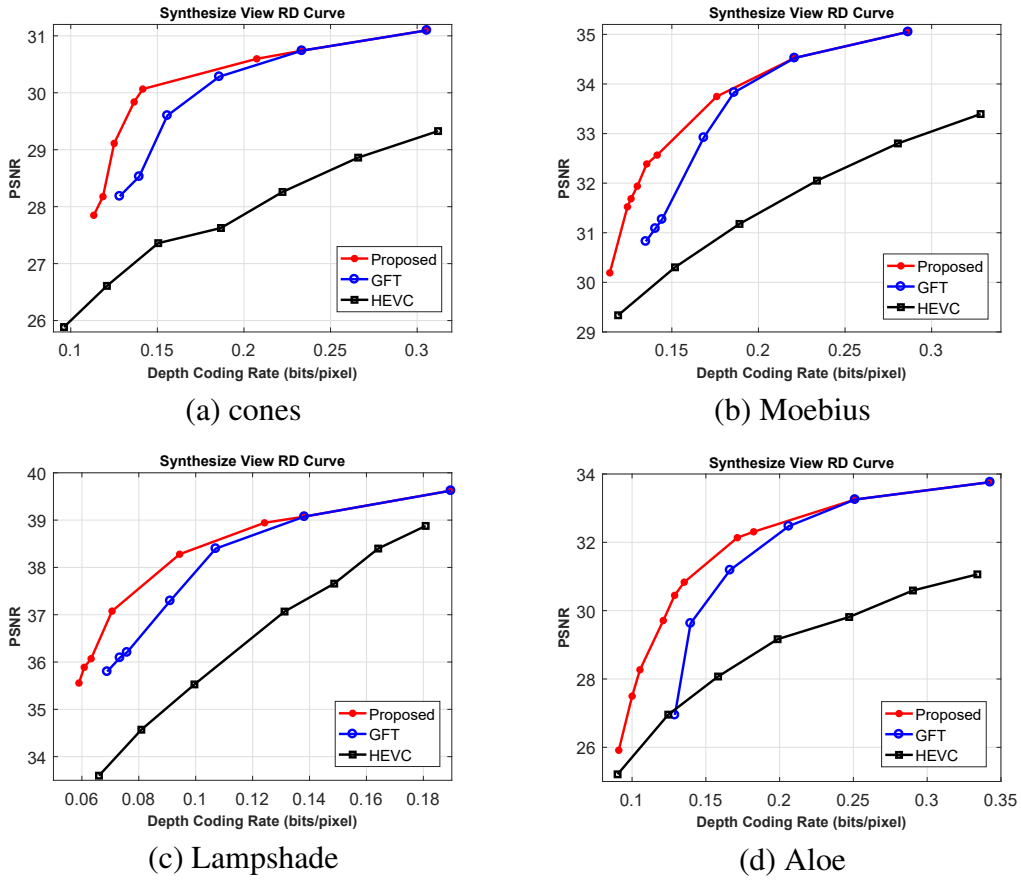


Fig. 4.9. Synthesized virtual view RD curves for cones, Moebius, Lampshade and Aloe, respectively.

almost the same for these three methods). As recommend in the standard [43], the QP offset ΔQP for the depth QP in relation to the color $\mathrm{QP_C}$ is determined as $\mathrm{QP} = \mathrm{QP_C} + \Delta\mathrm{QP}$, where $\Delta\mathrm{QP} \leq 9$. Finally, we synthesize virtual views using the decoded depth and color image pairs.

Fig. 4.8 shows the RS curves for `cones`, `Moebius`, `Lampshade` and `Aloe` image sets. The x-axis is the depth image coding rate in bits per pixel (bpp), and the y-axis is the synthesized view quality—scores measured by 3DSwIM. These figures show that our proposed method outperforms both `MR-GFT` and `HEVC`. The improved coding performance compared to `MR-GFT` validates the benefit of effectively approximating contours before edge-adaptive depth image compression. It demonstrates that a depth contour can be properly approximated with little

TABLE 4.1
BG RATE GAIN (RG) AND PSNR GAIN (PG) FOR RATE-SCORE (RS) AND RATE-DISTORTION (RD) CURVES

	RS-RG (rate%)		RD-RG (rate%)		RD-PG (dB)	
	MR-GFT	HEVC	MR-GFT	HEVC	MR-GFT	HEVC
teddy	15.78	17.51	11.11	39.54	0.55	1.66
cones	26.10	29.98	15.01	52.90	0.92	2.84
Dolls	19.38	11.46	9.64	27.70	0.19	1.28
Moebius	19.01	19.17	13.02	40.60	1.03	2.48
Books	26.69	23.68	20.53	43.68	0.47	1.86
Lamp.	18.35	27.78	16.66	44.51	1.29	3.30
Aloe	18.82	36.41	16.35	38.55	1.20	2.90
Dancer	7.31	19.47	3.89	45.30	0.23	5.94
GTFly	8.52	20.03	4.34	48.29	1.55	7.70
Avg.	18.11%	22.83%	12.28%	42.34%	0.83dB	3.33dB

degradation to the synthesized view quality. Both our proposed method and MR-GFT have better performance than HEVC, since the multi-resolution GFT on PWS image compression is more efficient than DCT based transform [18]. Fig. 4.9 illustrates the corresponding RD curves, where the x-axis is the depth coding rate and the y-axis is the PSNR value of synthesized virtual view. They also show that our proposed method has the best performance especially when the bit rate budget is low.

The computed BD gains [87] are shown in Table 4.1. Comparing with MR-GFT and HEVC, we compute the rate gain (RG) (in percentage) for both RS and RD curves. We also compute the PSNR gain (PG) (in dB) for RD curve. The results in Table 4.1 show that our proposed method can save a maximum of 29.69% bit rate in RS measure compared to MR-GFT and the average rate gain achieves non-trivial 18.11%. We also achieve an average of 22.83% rate gain in RS measure compared to HEVC. In RD measure, our method can also achieve 12.28% rate gain and 0.83dB PSNR gain compared to MR-GFT, and a 42.34% rate gain and 3.33dB PSNR gain compared to HEVC. These results prove the efficiency and effectiveness of our proposed method.

We also compare the performance of our proposed two contour approximation

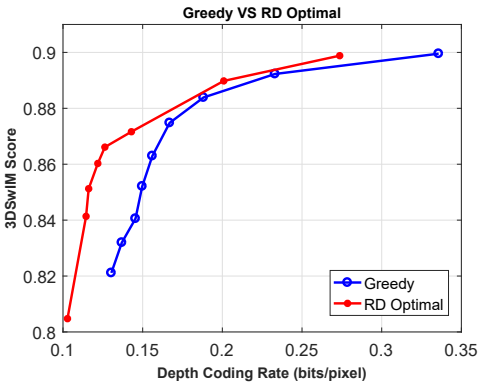


Fig. 4.10. Performance comparison: greedy versus RD optimal, for sequence `cones`.

methods: greedy algorithm in Chapter 3 and RD optimal algorithm proposed in this chapter. The resulting RS curve for image sequence `cones` is shown in Fig. 4.10, where our RD optimal contour approximation method has better performance than the greedy approximation algorithm. The reason is that the RD optimal method controls the induced distortion on synthesized virtual view during contour approximation.

4.4.3 Subjective Results Compared to `HEVC`

We generated selected subjective results to visually examine images outputted by our proposed method, namely: sub-regions from the synthesized views of `teddy`, `Dolls` and `cones`, as shown in Fig. 4.11.

In Fig. 4.11, the first row corresponds to images synthesized from compressed color and depth images by our proposed method, and images on the second row are synthesized from compressed images by `HEVC`. In each column, the coding bit rates for the depth and color images by the two methods are almost the same. We observe that the synthesized images from our proposed method are more visually pleasing since edges in these images remain sharp. In contrast, there are noticeable bleeding effects around edges in the synthesized images from `HEVC`. The results show that edge-adaptive depth image coding can lead to better synthesized view quality than fixed block transforms in compression standards like HEVC, which is consistent with results in previous edge-adaptive coding work [18], [23], [39].

(a) teddy (b) Dolls (c) cones

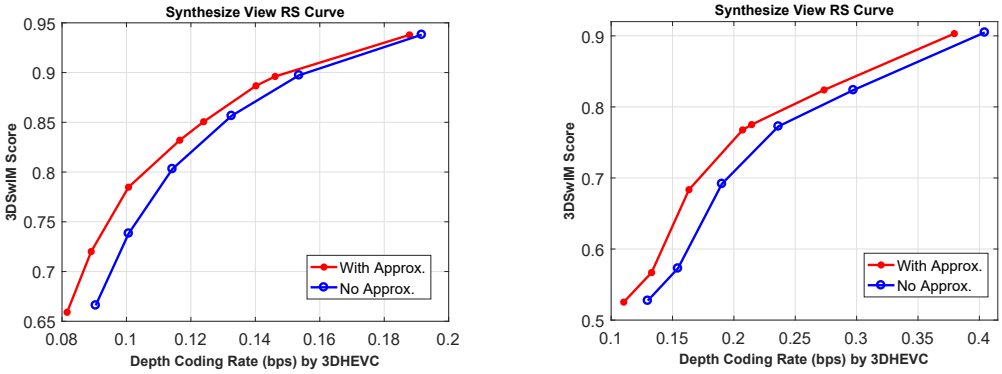
Fig. 4.11. Sub-regions of the synthesized views for teddy, Dolls and cones, respectively. Images in the first row is synthesized by approximated and multi-resolution GFT compressed depth images (proposed), where the second row is synthesized by HEVC compressed depth images. The depth and color coding rates are almost the same for each image sequence. (We strongly recommend readers to read an electronic version to distinguish the differences.)

4.4.4 Results Compressed by 3D-HEVC intra

The results demonstrated thus far show that our proposed object contour approximation can improve coding performance of edge-adaptive transform coding schemes like [18]. In theory, smoother contours can also improve other depth image codecs. To validate this point, we employ 3D video coding standard 3D-HEVC [43] intra HTM 6.0 to compress the original and the approximated color and depth images.

We test 3D-HEVC intra on `Undo Dancer` and `GTFly` sequences. We first approximate and alter the color-plus-depth image pairs with different values of λ. We then compress the original and the approximated color-plus-depth image pairs using 3D-HEVC, where the depth QP and color QP_{C} pairs are the same for all the image pairs. The resulting RS curves are shown in Fig. 4.12. We see

that using contour approximation, depth coding rate can be reduced by 6.84% and 11.55% for `Undo Dancer` and `GTFly`, respectively. The coding gain can be explained as follows. In 3D-HEVC, a depth block is approximated by a model that partitions the block into two smooth sub-regions according to detected edges. The simplified (smoothed) contours can facilitate the partitioning of blocks into smooth sub-regions, resulting in lower depth coding rates.



(a) Undo Dancer (b) GTFly

Fig. 4.12. RS curve by 3D-HEVC, where “No Approx." and “With Approx." mean using 3DHEVC to compress original and approximated color-plus-depth image pairs, respectively.

4.5 Conclusion

Efficient coding of depth image is essential for decoder-side virtual view synthesis via DIBR. Existing works employ either fixed transforms like DCT that blur a depth image’s sharp edges at low rates, or edge-adaptive transforms that require lossless coding of detected edges as side information, which accounts for a large share of the bit budget at low rates. In this chapter, we pro-actively alter object contours in an RD-optimal manner. We first propose a distortion proxy that is an upper bound of the established synthesized view quality metric, 3DSwIM. Given coding rate computed using AEC and our distortion proxy, contours are approximated optimally via a DP algorithm in an inter-view consistent manner. With the approximated contours, depth and color images are subsequently augmented and coded using a multi-resolution codec based on GFT [18] and HEVC respectively. Experiments show significant performance gain over previous coding schemes

using either fixed transform or edge-adaptive transform with lossless coding of detected contours.

4.6 Appendix 4.A: Derivation of the Maximum Likelihood Estimation of Parameters σ in (4.9)

Assuming the M coefficients $\{c_1, \cdots, c_M\}$ are *independent and identically distributed* and follow the probability density function (4.9), the likelihood function for M coefficients becomes

$$\begin{aligned} L_\sigma(c) &= \prod_{i=1}^{M} \frac{1}{2\sigma} \exp\left(-\frac{|c_i|}{\sigma}\right) \\ &= (2\sigma)^{-M} \cdot \exp\left(-\frac{1}{\sigma}\sum_{i=1}^{M}|c_i|\right). \end{aligned} \tag{4.25}$$

Take the $\log$ likelihood function as $l_\sigma(c) = \log(L_\sigma(c))$ and we get

$$l_\sigma(c) = -M\ln(2\sigma) - \frac{1}{\sigma}\sum_{i=1}^{M}|c_i|. \tag{4.26}$$

Take the derivative of $l_\sigma(c)$ with respect to σ

$$\frac{\partial l}{\partial \sigma} = -\frac{M}{\sigma} + \frac{1}{\sigma^2}\sum_{i=1}^{M}|c_i|. \tag{4.27}$$

To solve $\frac{\partial l}{\partial \sigma} = 0$, the maximum likelihood estimation of σ is

$$\sigma = \frac{1}{M}\sum_{i=1}^{M}|c_i|. \tag{4.28}$$

Chapter 5

Live Free Viewpoint Video Streaming: Peer Grouping Optimization

Having investigated how contour approximation can improve depth image coding efficiency in term of synthesized virtual view quality, we now look at how we can improve free viewpoint video streaming[1].

5.1 Introduction

In free viewpoint video [7], a user can select any virtual view from which an image of the 3D scene is rendered for observation. Specifically, given a 1D array of cameras with positions $\mathcal{V} = \{1, \ldots, V\}$, an image of virtual view u is typically synthesized using color and depth images from two nearby captured reference views, v^l and v^r, where $v^l < u < v^r$ and $v^l, v^r \in \mathcal{V}$, via DIBR. For users who are observing the same free viewpoint video synchronized in time—e.g., during a live video broadcast of a public event like a piano recital—but not necessarily from the same viewpoint, they have incentive to pull color and depth video streams from the same reference views, so that the streaming cost can be shared. On the other hand, it has been shown [24], [25] that in general distortion of the synthesized view increases with its distance to the reference views. Thus, a user also has incentive to select video of reference views that tightly "sandwich" his chosen virtual view, in order to minimize visual distortion. This poses an interesting dilemma for users:

[1] A version of this chapter has been published in *IEEE International Conference on Image Processing (ICIP), Melbourne, Australia, September, 2013.*

how to best select and share video streams of different reference views, so that the streaming cost and the resulting collective synthesized view distortion are optimally traded off?

In a previous work [96], reference view sharing strategies are studied for the case where users are first divided into groups, and then each group independently pulls and shares the streaming cost of two reference views, using which virtual views of the group's users are synthesized. While the developed algorithms are simple and intuitive, it is easy to see that this type of groupings is sub-optimal. First, it is possible for multiple groups to be independently pulling the same video view, when the cost of this common view can actually be shared by the union of these groups. Second, members belonging to the same group must share *both* reference views. But it may be more beneficial for them to share only one reference view, and separately find appropriate groups to share a different second reference view for view synthesis.

In this chapter, we propose to generalize the previous notion of user grouping, so that a user can simultaneously belong to two groups, and each group pulls one reference view and shares the streaming cost of a single view. If two groups have the same reference view, we merge the two groups into one single group. Doing so means that a video view is never pulled more than once, and its cost is shared only by those who are using this view as reference for view synthesis. To study a stable user grouping, we exploit tools from game theory [26], and seek a Nash Equilibrium (NE) solution of reference view selection, from which no one has incentive to unilaterally deviate. Specifically, we first derive a lemma based on known properties of synthesized view distortion functions. We then design a search algorithm to find locally optimal groupings, leveraging on the derived lemma to reduce search space, thus reducing computation complexity.

The outline of the chapter is as follows. We first formulate our problem. We then derive our lemma and the corresponding optimization algorithm. Finally, experimental results and conclusion are presented respectively.

5.2 Problem Formulation

In this section, we first describe the free viewpoint video model we choose for our problem formulation. We then describe properties of the synthesized view distortion and subscription fee sharing.

5.2.1 Free Viewpoint Video Model

Let $\mathcal{V} = \{1, \ldots, V\}$ be a discrete set of *captured views* for V equally spaced cameras in a 1D array. Each camera captures both a color image and a depth image at the same resolution. The color image from an intermediate *virtual view* between any two cameras can be synthesized using color and depth images of the two camera views (*reference views*) via a DIBR technique like 3D warping [97]. DIBR essentially maps color pixels in the reference views to appropriate pixel locations in a virtual view; such locations are derived from the corresponding depth pixels in the reference views. Disoccluded pixels in the synthesized view—pixel locations that are occluded in the two reference views—can be completed using depth-based inpainting techniques [33], [34]. Because inpainting offers only a best-guess solution, the larger the disoccluded regions are, the lower the synthesized view image quality will be in general.

More specifically, let u be the virtual view that a peer currently requests for observation. We assume u can be written as $u = v + \frac{k}{K}$, $v \in \{1, \ldots, V-1\}$ and $k \in \{0, \ldots, K\}$, for some large pre-determined constant K. In other words, u belongs to an ordered discrete set of intermediate viewpoints—the set of views between (and including) camera views 1 and V, spaced apart by integer multiples of $1/K$. A discrete distribution function q_u describes the fraction of peers who currently request the virtual view u. Hence, Nq_u is the number of users requesting the virtual view u, where N is the total number of all the users.

5.2.2 Synthesized View Distortion

Typically, to construct a virtual view u that is not itself a camera-captured view, DIBR requires left and right reference views v^l and v^r such that $v^l < u < v^r$,

where $v^l, v^r \in \{1, \cdots, V\}$. Note that v^l and v^r do not have to be the closest captured views to u. The distortion of the synthesized view, $d_u(v^l, v^r)$, varies with the choices of reference views, v^l and v^r. We assume that $d_u(\cdot)$ has the following three properties.

First, further away reference views v^l and v^r to virtual view u induce no smaller distortion, that is,

$$\begin{aligned} d_u(v^l, v_1^r) &\leq d_u(v^l, v_2^r) \quad \text{if } v_1^r < v_2^r, \text{ and} \\ d_u(v_2^l, v^r) &\geq d_u(v_1^l, v^r) \quad \text{if } v_2^l < v_1^l. \end{aligned} \tag{5.1}$$

We call this the *monotonicity in reference view distance* for synthesized view distortion. This is reasonable in general, since further reference views usually result in more disoccluded pixels in the virtual view image. The disoccluded regions can be filled using inpainting algorithms [98], [99], but in general the larger the disocclusion, the higher the penalty in virtual view quality. Experiments on a large number of multi-view image suqences in [24], [25] also demonstrate this monotonicity assumption.

Second, given virtual view u and left and right reference views v^l and v^r, define $\varepsilon = \min\left(|v^l - u|, |v^r - u|\right)$ as the *minimum reference view distance* between u and v^l, v^r. We assume that a smaller ε induces no larger distortion, i.e.,

$$\begin{aligned} d_{u_1}(v^l, v^r) &\leq d_{u_2}(v^l, v^r) \quad \text{if } \varepsilon_1 < \varepsilon_2 \\ \text{where } \varepsilon_1 &= \min\left(|v^l - u_1|, |v^r - u_1|\right) \\ \text{and } \varepsilon_2 &= \min\left(|v^l - u_2|, |v^r - u_2|\right). \end{aligned} \tag{5.2}$$

We call this the *monotonicity in minimum reference view distance*. This is also reasonable, since it is observed empirically that when both reference views are encoded at the same quality (using the same quantization parameter (QP)), the worst synthesized view distortion tends to take place at the middle view (with the largest ε) [24]. Further, as the minimum reference view distance ε approaches zero, the

synthesize virtual view point is essentially one of the two reference views with no distortion since there is no need for synthesis. Our monotonicity in minimum reference view distance assumption ensures that synthesized view distortion for $\varepsilon = 0$ is the minimum.

Third, we assume that the rate of distortion increase with respect to minimum reference view distance is no smaller if the current distortion is higher. Mathematically, we write:

$$\frac{\partial d_u(v^l, v^r)}{\partial v^r} = \phi(d_u(v^l, v^r)), \text{ if } |v^r - u| \leq |v^l - u| \tag{5.3}$$

where the left side of the equation means the rate of distortion increase with respect to v^r if virtual view u is closer to v^r than it is to v^l. The right side $\phi(\cdot)$ is a monotonically non-decreasing function; a larger d_u results in a no smaller $\phi(d_u)$. Similar assumption applies when virtual view u is closer to the left reference view. We call this the *monotonicity in reference view slope*. We give below two examples of $d_u(v^l, v^r)$ that follow this property: i) $d_u(v^l, v^r)$ is a linear function of v^r, thus $\phi(\cdot) = \frac{\partial d_u}{\partial v^r} = c$ for a constant c; ii) $d_u(v^l, v^r)$ is an exponential function of v^r, in which case $\phi(\cdot) = c \cdot d_u(v^l, v^r)$.

Using the chain rule, one can see that this assumption implies *convexity* of distortion $d_u(v^l, v^r)$ in v^r:

$$\frac{\partial^2 d_u(v^l, v^r)}{\partial^2 v^r} = \frac{\partial \phi(d_u(v^l, v^r))}{\partial d_u(v^l, v^r)} \frac{\partial d_u(v^l, v^r)}{\partial v^r} \geq 0. \tag{5.4}$$

The first term $\frac{\partial \phi(d_u(v^l, v^r))}{\partial d_u(v^l, v^r)}$ is non-negative since $\phi(\cdot)$ is a monotonically non-decreasing function. The second term $\frac{\partial d_u(v^l, v^r)}{\partial v^r}$ is also non-negative since $d_u(v^l, v^r)$ is a monotonically non-decreasing function in reference view v^r by the first assumption. Hence the second-order derivative $\frac{\partial^2 d_u(v^l, v^r)}{\partial^2 v^r}$ is non-negative, and $d_u(v^l, v^r)$ is convex in reference view v^r. The assumption of convexity in distortion function is common in classical rate-distortion analysis. An example of distortion function $d_u(\cdot)$ satisfying the above properties is shown in Fig. 5.1, where both u'

and u are closer to right reference view. $d_{u'}(v^l, v^r) \geq d_u(v^l, v^r)$ is due to property of monotonicity in minimum reference view distance. When v^r changes from v_1^r to v_2^r, the distortion increase $\delta_{u'}$ and δ_u satisfy $\delta_{u'} \geq \delta_u$.

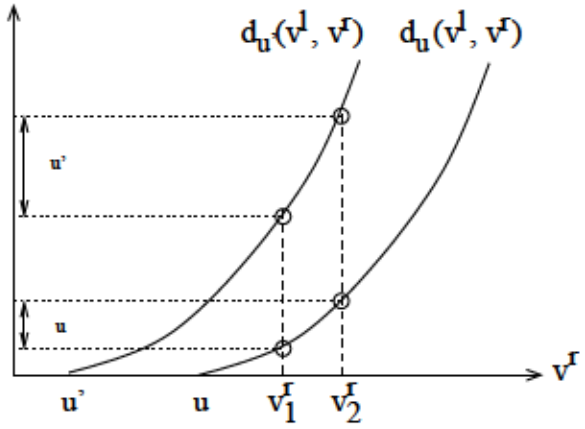


Fig. 5.1. Example of synthesized view distortion as function of right reference view v^r for two virtual views u and u'.

For a virtual view u that itself is a camera-captured view, it can also be synthesized by a pair of left and right reference views v^l and v^r where $v^l < u < v^r$. The distortion $d_u(v^l, v^r)$ follows the same properties as discussed above. Alternatively, it can be perfectly constructed with the camera-captured view u with zero distortion. Based on the above discussion, for any virtual view u, let $\mathcal{V}_u$ denote its selected reference view set, and the corresponding distortion is:

$$D_u(\mathcal{V}_u) = \begin{cases} d_u(v^l, v^r), \text{ if } \mathcal{V}_u = \{v^l, v^r\}, \\ 0, \text{ if } u \text{ is an camera view and } \mathcal{V}_u = \{u\}. \end{cases} \tag{5.5}$$

5.2.3 Subscription Fee Sharing

We assume that the server charges subscription fee A for streaming one video view (color and depth). Assuming N users are close to each other in network distance, and users who request the same camera view can request only one copy from the server, and share the view and the subscription fee A with each other. Recall that q_u is the fraction of users who are requesting the virtual view u. Let $n_v = \sum_u q_u I[v \in \mathcal{V}_u]$ be the fraction of users utilizing view v as reference, where $I[x]$ is an indicator

function that equals 1 if clause x is true and 0 otherwise. Thus, each user requesting view v can access view v with the subscription payment $s(v) = A/(Nn_v)$.

Based on the above discussion, given reference view selection $\mathcal{V}_u$, a user of virtual view u has the overall cost $c_u(\mathcal{V}_u)$ that includes two terms: synthesized view distortion and the subscription payment

$$c_u(\mathcal{V}_u) = D_u(\mathcal{V}_u) + \lambda \sum_{v \in \mathcal{V}_u} s(v), \tag{5.6}$$

where λ is a parameter adjusting the weight between synthesized view distortion and subscription fee.

5.3 Aggregate Performance Optimization

Assuming that multiple users watching the same virtual view u will select the same reference views $\mathcal{V}_u$. Given the number of users and the virtual view each user is watching, we now derive an algorithm to find a stable NE solution for users' reference view selections $\{\mathcal{V}_u^*\}$. It means that user of any virtual view u cannot deviate from the NE solution $\mathcal{V}_u^*$ and further reduce its own cost, given that users at all other virtual views u' follow the NE solution $\{\mathcal{V}_{u'}^*\}$. Mathematically, we have $c_u(\mathcal{V}_u^*) \leq c_u(\mathcal{V}_u)$ for any u given that all other virtual views follow $\{\mathcal{V}_{u'}^*\}$. Thus, no one has incentive to unilaterally change his reference view selection. We first describe a condition for reference view selection in an NE solution. We then propose an efficient algorithm to find an NE solution $\{\mathcal{V}_u^*\}$.

5.3.1 Condition for an Equilibrium Solution

Given the properties of the synthesized view distortion described in Section 5.2.2, we state formally the following important lemma on the selection of right reference views in the equilibrium solution.

Lemma 1: In the equilibrium solution, suppose user of virtual view u chooses left and right reference views v^l and v_1^r, where $|v^l - u| > |v_1^r - u|$. Then, user of view $u' < u$ with left reference view w^l, where $w^l \leq v^l$, cannot choose right reference view v_2^r that satisfies $v_2^r \geq v_1^r$. In other words, v_2^r cannot be within the

range $\left(v_1^r, \min\{2u' - w^l, 2u - v^l\}\right]$.

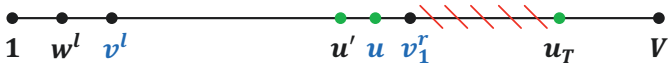


Fig. 5.2. Relative positions of w^l, v^l, u', u, v_1^r, u_T, where $u_T = \min\{2u - v^l, 2u' - w^l\}$. The black points means the anchor (reference) views. Virtual viewpoints between neighboring anchor views are not shown in this figure.

The relative positions of w^l, v^l, u', u, v_1^r, u_T are illustrated in Fig. 5.2, where $u_T = \min\{2u - v^l, 2u' - w^l\}$. Lemma 1 can be explained as follows. Considering that user of view u' with a left reference view w^l now needs to choose the proper right reference view $v_2^r \in \{\lceil u' \rceil, \cdots, V\}$, where V is the maximum number of reference views. If there is user of view u that already selects left and right reference views v^l and v_1^r, and these views satisfy $|v^l - u| > |v_1^r - u|$, $u' < u$ and $w^l \leq v^l$, then the range $\delta = \left(v_1^r, \min\{2u' - w^l, 2u - v^l\}\right]$ can be taken out when selecting v_2^r. The condition $|v^l - u| > |v_1^r - u|$ makes sure u is more closer to right reference view [2]. We now prove the above lemma as follows.

Proof of Lemma 1 We prove by contradiction. Suppose that in the equilibrium solution, a user of virtual view u', $u' < u$, with left reference view w^l, where $w^l \leq v^l$, selects right reference view $v_2^r \in \left(v_1^r, \min\{2u' - w^l, 2u - v^l\}\right]$. We consider two cases: i) $w^l = v^l$ and ii) $w^l < v^l$.

Consider first the case where $w^l = v^l$. Since $v_2^r \in \left(v_1^r, \min\{2u' - w^l, 2u - v^l\}\right]$ and $u' < u$, the largest value that v_2^r can take is $2u' - v^l$. Hence

$$v_2^r \leq 2u' - v^l, \tag{5.7}$$

which leads to

$$v_2^r - u' \leq u' - v^l. \tag{5.8}$$

That means virtual view u' is always closer to right reference view v_2^r than it is to

[2] If u is closer to its left reference view, a similar lemma can be written for the selection of left reference views in the equilibrium solution as well.

left reference view v^l. Given $u' < u$, virtual view u is also closer to right reference view v_2^r than it is to left reference view v^l, and we have $|v^r - u'| > |v^r - u|$, for $v^r \in [v_1^r, v_2^r]$. Further, by monotonicity in minimum reference view distance, $u' < u$ means:

$$d_{u'}(v^l, v^r) \geq d_u(v^l, v^r) \,, \quad \forall\, v^r \in [v_1^r, v_2^r]. \tag{5.9}$$

Consequently, by monotonicity in reference view slope, we have

$$\frac{\partial d_{u'}(v^l, v^r)}{\partial v^r} \geq \frac{\partial d_u(v^l, v^r)}{\partial v^r} \,, \quad v_1^r \leq v^r \leq v_2^r. \tag{5.10}$$

Hence, we can conclude that:

$$\begin{aligned} d_{u'}(v^l, v_2^r) - d_{u'}(v^l, v_1^r) &\geq d_u(v^l, v_2^r) - d_u(v^l, v_1^r) \\ &> \lambda\,(s(v_1^r) - s(v_2^r)). \end{aligned} \tag{5.11}$$

The last inequality stems from the fact that user of virtual view u selects right reference view v_1^r over v_2^r in the NE solution, which means that $d_u(v^l, v_1^r) + \lambda s(v_1^r) < d_u(v^l, v_2^r) + \lambda s(v_2^r)$. Eq. (5.11) also means

$$d_{u'}(v^l, v_2^r) + \lambda s(v_2^r) > d_{u'}(v^l, v_1^r) + \lambda s(v_1^r) \tag{5.12}$$

which implies user of virtual view u' can achieve a lower cost by choosing right reference view v_1^r over v_2^r. A contradiction happens.

We next consider the case where $w^l < v^l$. Because $w^l < v^l < u$, user of virtual view u can also select w^l as left reference view for view synthesis. From the assumption of monotonicity in reference view distance, we have

$$d_u(w^l, v^r) \geq d_u(v^l, v^r) \,, \quad \forall\, v^r \in [v_1^r, v_2^r]. \tag{5.13}$$

Again, by monotonicity in reference view slope, we have

$$\frac{\partial d_u(w^l, v^r)}{\partial v^r} \geq \frac{\partial d_u(v^l, v^r)}{\partial v^r} \,, \quad v_1^r \leq v^r \leq v_2^r. \tag{5.14}$$

Given the above two observations, we can conclude that

$$\begin{aligned} d_u(w^l, v_2^r) - d_u(w^l, v_1^r) &\geq d_u(v^l, v_2^r) - d_u(v^l, v_1^r) \\ &> \lambda(s(v_1^r) - s(v_2^r)), \end{aligned} \tag{5.15}$$

which leads to $d_u(w^l, v_2^r) + \lambda s(v_2^r) > d_u(w^l, v_1^r) + \lambda s(v_1^r)$. It means that if using the left reference view w^l, user of virtual view u still prefers choosing right reference view v_1^r over v_2^r to achieve a lower cost.

Following similar steps as in the first case, it can be shown that

$$\begin{aligned} d_{u'}(w^l, v_2^r) - d_{u'}(w^l, v_1^r) &\geq d_u(w^l, v_2^r) - d_u(w^l, v_1^r) \\ &> \lambda(s(v_1^r) - s(v_2^r)), \end{aligned} \tag{5.16}$$

which leads to $d_{u'}(w^l, v_2^r) + \lambda s(v_2^r) > d_{u'}(w^l, v_1^r) + \lambda s(v_1^r)$. This also contradicts the assumption that user of virtual view u' selects v_2^r over v_1^r in the NE solution. Since both cases are shown to be contradictions, the lemma is proven. □

Lemma 1 shows that when a user of virtual view u selects left and right reference views v^l and v^r, for users in any virtual view u', $u' < u$, with left reference view w^l, $w^l \leq v^l$, they will not select right reference view w^r in the range $\delta = \big(v^r, \min\{2u' - w^l, 2u - v^l\}\big]$. Lemma 1 can help reduce the search space when we seek for the equilibrium solution.

Consider first an exhaustive search algorithm, where for each virtual view u', it tries all possible left and right reference views w^l and w^r and chooses the optimal pair to minimize cost for view u'. If we apply lemma 1 for each virtual view u' and left reference w^l, we first need to find out the excluded range δ and then try the remaining candidate right reference views. In practice, determining δ can itself be computation-expensive since we need to check every possible view $u > u'$. When the length of δ, denoted as $|\delta|$, is small, the saving in a reduced search space is outweighed by the computation of δ. Thus lemma 1 should only be selectively applied to speed up a search algorithm.

It is clear that when $(2u' - w^l)$ or $(2u - v^l)$ is small, $|\delta|$ will also be small. Further, if u' is close to V, this means there are not many candidate right reference views in the first place. Thus, we set two necessary conditions before applying lemma 1 to a search algorithm: i) $2u' - w^l \geq \tau_1$, and ii) $u' \leq \tau_2$, where τ_1 and τ_2 are pre-determined parameters. We detail our search algorithm next.

5.3.2 Efficient Algorithm for Nash Equilibrium Solution

Using lemma 1, we describe an algorithm that finds an NE solution.

Algorithm 2: Nash Equilibrium Solution Search

```
1: Identify range [u^l, u^r] that contains all peers.
2: Initialize V_u = {⌊u^l⌋, ⌈u^r⌉}, ∀u.
3: repeat
4:    for each virtual view u' with viewers do
5:       for each left reference w^l ∈ [⌊u^l⌋, ⌊u'⌋] do
6:          δ = ∅.
7:          if (2u' − w^l) ≥ τ_1 and u' ≤ τ_2 then
8:             for each virtual view u > u' do
9:                δ = δ ⋃ (v^r, min{2u' − w^l, 2u − v^l}].
10:            end for
11:         end if
12:         for each w^r ∈ [⌈u'⌉, ⌈u^r⌉] \ δ do
13:            if c_{u'}({w^l, w^r}) is the smallest cost so far then
14:               Update V_{u'} = {w^l, w^r}.
15:            end if
16:         end for
17:      end for
18:   end for
19: until {V_u} is stable.
```

We first initialize a tight virtual range $[u^l, u^r]$ that contains all peers. The tightest reference views that sandwich this range are $\lfloor u^l \rfloor$ and $\lceil u^r \rceil$, which we use to initialize $\mathcal{V}_u$'s.

Then, given solution $\{\mathcal{V}_u\}$ in the last iteration, for each virtual view u' with viewers, we search its optimal reference view selection $\mathcal{V}_{u'}$ assuming that users of other views follow $\{\mathcal{V}_u\}$. Specifically, for each possible left reference w^l, we search its optimal right reference w^r that gives the lowest cost $c_{u'}(\{w^l, w^r\})$. The search range for w^r can be decreased by δ, if the two conditions $(2u' - w^l) \geq \tau_1$ and

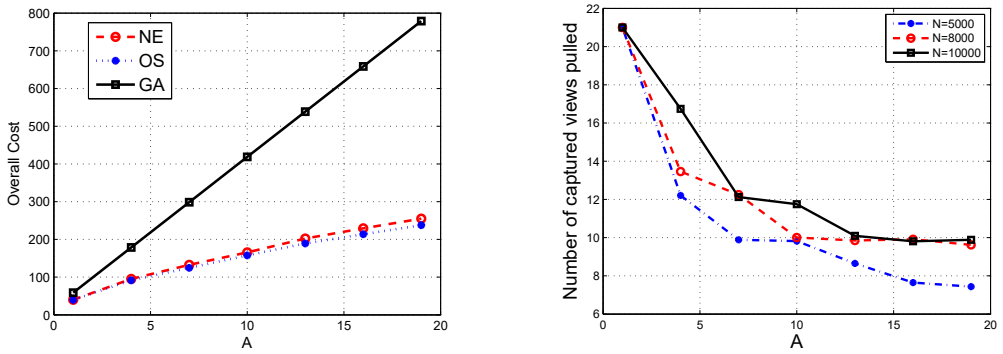
$u' \leq \tau_2$ are satisfied and lemma 1 is applied. The algorithm is repeated until the solution $\{\mathcal{V}_u\}$ is stable or when a pre-defined maximum number of loops is reached.

5.4 Experimentation

For our experiments, we use the same synthesized view distortion function as that in [96]:

$$d_u(v^l, v^r) = \gamma e^{\alpha(v^r - v^l)}(e^{\beta * \min(u - v^l, v^r - u)} - 1) \tag{5.17}$$

which meets all the properties described in Section 5.2.2. We also use the same parameters: $\gamma = 0.06$, $\alpha = \beta = 0.2$. We set $\lambda = 1$ for Eq. (5.6). For available views for free viewpoint navigation, we assume 21 captured views and 221 virtual views. We further assume each user's view distribution follows a uniform distribution, and he selects a particular view with probability $1/221$. We test our system under different network size N (*i.e.*, the number of users) and subscription fee A. We run each simulation for 200 times and will show the average results in the following discussions.



(a) overall cost (b) captured views pulled

Fig. 5.3. Overall cost and number of captured views pulled versus subscription fee. In (a), `NE`, `OS` and `GA` are compared for fixed network size $N = 5000$. In (b), network size varies $N = 5000$, 8000 and 10000.

Fig. 5.3(a) shows the performance of our proposed *NE solution* (`NE`) comparing with the *optimal reference view selection* (`OS`) and the *grouping algorithm* (`GA`) studied in [96]. The `OS` assumes that all users cooperatively purchase a subset of reference views $\mathcal{V}' \subseteq \mathcal{V}$, and each peer will select the tightest left/right references

from $\mathcal{V}'$ to minimize its distortion. OS aims to find the optimal $\mathcal{V}'$ to minimize the overall cost (i.e., the total distortion and the total subscription payment of all users). Although OS gives the global optimal reference selection, it is not stable; it does not take into account users' selfish tendency to seek to reduce their own cost instead of the overall cost. In GA, all users between two neighboring camera-captured views form a group, and each group independently requests the tightest left and right reference views. The users in the same group share the subscription fee together. Fig. 5.3(a) shows the overall cost of the three algorithms. We first observe that GA gives the highest overall cost, since different groups do not share reference views. We also observe that NE results in a slightly higher overall cost than OS. Thus, it can be concluded that NE provides incentive for users to form stable reference selection with only a small loss of the overall performance.

Fig. 5.3(b) shows the total number of requested reference views selected by users via NE for different network sizes. We first observe that the number of requested reference views decreases when subscription fee A increases. This is because when A is larger, users request a smaller number of reference views with more users for each reference to share the fee. We observe also that the number of requested reference views increases with increased network size. This is because a larger network has more users to share the cost. Since more reference views can effectively reduce users' distortions, each user can have a lower cost on average in a larger network.

5.5 Conclusion

In this chapter, we study the challenging problem of the optimal reference view selection in a cooperative free viewpoint streaming system, where a user can simultaneously belong to two groups and share the streaming cost of a single view in each group. To do this, we first derive a lemma based on known properties of synthesized view distortion functions. We then design a search algorithm to find locally optimal groupings (NE), leveraging on the derived lemma to reduce search space, thus reducing computation complexity. Experimental results show that the

NE gives a stable reference view selection, from which no user has incentive to unilaterally deviate. The stable NE does not increase the overall cost much, when compared with the unstable optimal reference selection that gives the lowest overall cost. Furthermore, a larger network has the ability to request more reference views to reduce users' distortion without much increase of the subscription fees shared by each user. Thus, everyone can reach a lower cost in a larger network.

Chapter 6

Interactive Light Field Streaming: Landmarks Inserting

We now look at how to improve the performance of interactively streaming light field images. [1]

6.1 Introduction

Light field (LF) cameras such as Lytro[2] employ a 2D array of microlenses before the image sensor to capture multiple light ray intensities and directions per pixel, so that a user can navigate and observe a static 3D scene from different viewpoints post-capture. However, the volume of captured LF data is large, and downloading the entire data prior to user's viewpoint navigation would incur a large startup delay.

Previous works propose an *interactive light field streaming* (ILFS) framework [27]–[30], where a user periodically requests a desired view, and in response a server transmits a pre-synthesized and encoded viewpoint image to the user for his observation. The image can either be independently encoded as an I-frame, or be differentially encoded as a P-frame using another image as a predictor, where the coding bit-count for a P-frame is much smaller than coding an I-frame. The technical challenge is to design and pre-encode a storage-constrained frame

[1] A version of this chapter has been accepted by *IEEE International Conference on Image Processing (ICIP), Beijing, China, September, 2017*. An extension of a journal version is in preparation.

[2] https://illum.lytro.com/

structure containing I- and P-frames[3] to facilitate user-requested view-switches during an ILFS session. Pre-encoding only I-frames for all views would lead to a large transmission cost, while pre-encoding P-frames for all possible user view-switch requests from any view i to j for an LF array of N views would require $O(N^2)$ P-frames to be stored in advance in the server and thus is expensive in storage cost of the server.

To lower transmission cost while reducing storage requirement, we design a new frame structure[4] to facilitate ILFS via optimal selection of *landmarks*. A landmark operates like an airline hub in commercial aviation[5]: by creating direct flights to/from a designated hub for all cities—$O(2N)$ flights for N cities—a passenger can travel from any city to any other city via only two flights (one connecting flight to the hub first, then from the hub to the destination). Similarly, adding P-frames to/from a landmark view for all the remaining views means that a user's request to switch from any viewpoint image to any other view can be essentially accomplished by decoding two differential P-frames (transition to landmark first, then to designation view). Hence the number of stored P-frames is only $O(2N)$. The crux is how to select locations of landmark views, and P-frame connections to/from landmarks for the remaining views.

In this chapter, we propose to optimally insert landmarks for ILFS. We first use a Lloyd's algorithm variant [100] to recursively find locally optimal locations of landmarks. We then add P-frames connecting the remaining views to/from their closest landmarks and P-frames connecting among landmarks. Finally, we greedily add/remove P-frames from the initialized structure based on a rate-storage criterion.

The outline of the chapter is as follows. We first discuss our ILFS system and view navigation model. We then compute the expected ILFS transmission cost. We next discuss our frame structure design using landmarks. Results and conclusion

[3]To be discussed in Section 6.2, an M-frame [83] is also needed after decoding each P-frame to identically merge to a target I-frame reconstruction.

[4]Here *frame structure* means which I-, P- and M-frames will be pre-encoded and stored at the server.

[5]As an example, United Airline has its largest domestic hub in Chicago O'Hare International Airport.

are presented at the end, respectively.

6.2 System Overview

For smooth transition to later sections describing our core contribution on optimally inserting landmarks to facilitate ILFS, as introduction materials we first review the ILFS system model. We then present a view navigation model for ILFS that describes a typical user's view-switching behavior. We also describe I-, P- and merge (M-) frames [83] in our coding structure to facilitate a user's view-switching requests.

6.2.1 System Overview

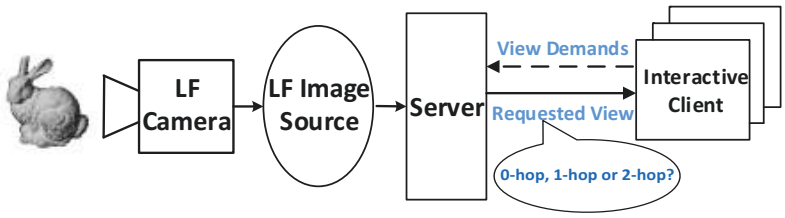


Fig. 6.1. Interactive light field streaming system.

The system model we consider for ILFS problem is shown in Fig. 6.1. An LF camera first captures a static 3D scene with spatially correlated LF images from multiple viewpoints. A server encodes these source images offline, building an optimal frame structure θ^* containing I-, P- and M-frames for each image. The server stores the structured representations of the LF source images locally, using which the server serves multiple streaming clients subsequently. This process results in a storage cost at the server. The clients interactively request demanding images along their view-switching trajectory from the server. The server then sends the requested views to clients by choosing from three different transmission types: *0-hop*, *1-hop* or *2-hop* (to be discussed in Section 6.3.3), resulting in a transmission cost. The problem is to find out an optimal frame structure θ^* stored at the server, so as to best trade off the storage cost and the expected transmission cost. An alternative approach for the server to interact with clients is to real-time encode

a path traversal tailor-made for each client's interactivity. However, real-time encoding for each client is computational prohibitive if the number of clients is large.

6.2.2 View Navigation Model for ILFS

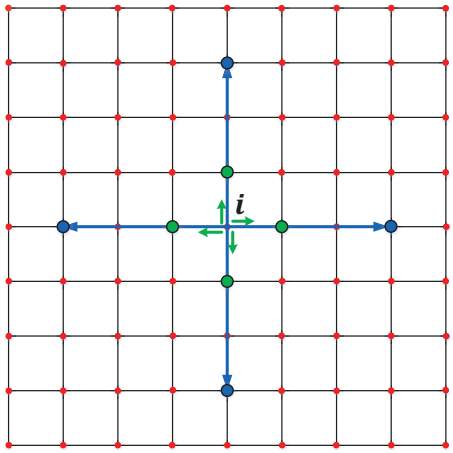


Fig. 6.2. An example of 2D grid of 9×9 views. The green and blue arrows respectively represent possible view-switching `walk` and `jump` from view i with $K = 3$.

To discuss view navigation model for ILFS, we first assume that the N viewpoints of a static scene captured by an LF camera are arranged into a $\sqrt{N} \times \sqrt{N}$ 2D grid. Existing common LF user interfaces[6] allow different kinds of view-switches, which we categorize into two types: i) switch to a horizontal / vertical adjacent view; ii) jump to a horizontal / vertical view that are K views apart, where K is a constant.

Specifically, at a viewpoint (x, y), one can choose either `walk` or `jump` to a next view. `walk` means that a user can switch from view (x, y) to a vertical $(x \pm 1, y)$ or a horizontal $(x, y \pm 1)$ adjacent view. By `jump`, we mean a user can switch a view K distance apart, *i.e.*, from view (x, y), one can switch to views $(x \pm K, y)$ or $(x, y \pm K)$. An example of view switching for a 9×9 2D grid of views with $K = 3$ is shown in Fig. 6.2, where the green and blue arrows represent possible view-switching `walk` and `jump`, respectively.

[6]For example, http://lightfield.stanford.edu/

We assume that the probability of switching to each of the four adjacent views and the four distant views is the same. Thus, from view i, a user can switch to a view $j \in \mathcal{N}(i)$ with probability $p_{j|i} = \frac{1}{8}$, where $\mathcal{N}(i)$ contains the four adjacent views and four distant views which can be switched to, starting from view i.

6.2.3 Frame Types in Coding Structure

In our proposed frame structure, by default we assume each view i has one independently encoded I-frame, denoted by I_i, which does not require any predictor frame for decoding. A pre-encoded I-frame ensures that a server can always enable a user to switch to view i by transmitting I_i to the user, albeit at a large transmission cost. To more efficiently facilitate a view-switch from view j to i, view i may contain in addition a differentially coded P-frame $P_i(j)$, which uses I-frame I_j of view j as a predictor. Thus, $P_i(j)$ is transmitted only if the user has I_j in the buffer, since the decoding of P-frame $P_i(j)$ requires an existing I_j. Moreover, the further view i is away from view j, the bigger the residual difference between view i and view j, and hence the larger coding bit-rate of $P_i(j)$.

Note that view i may also be used as a predictor for future view-switches. If view i contains only one P-frame, we can directly use the reconstruction of view i from this P-frame as the predictor for the next view-switch. However, in general, view i may contain multiple P-frames $P_i(j)$ for view-switches from different views j to i. The reconstructions of view i from different P-frames may differ slightly due to transform domain quantization of encoding the different prediction residuals for these P-frames. If we directly use the reconstructions of view i from these P-frames as predictor, the different reconstructions may lead to prediction mismatch and thus a misaligned reconstruction of the next view in trajectories. Hence, to avoid future coding drift, an identical version of I_i is necessary to be the predictor for future view-switches. Thus, we employ a *merge frame* (M-frame) M_i [83] to merge the different reconstructions of view i (from different P-frames) into an identical one.

M-frame in [83] is a new design that uses shift and rounding operations to achieve desired merging results. Specifically, the authors employ a PWC function

$f(x)$ as the merge operator to merge quantized transform coefficients from different reconstructions to the same value, where

$$f(x) = \left\lfloor \frac{x+c}{W} \right\rfloor W + \frac{W}{2} - c. \tag{6.1}$$

W stands for step size and c stands for horizontal shift. An example of $f(x)$ is shown in Fig. 6.3.

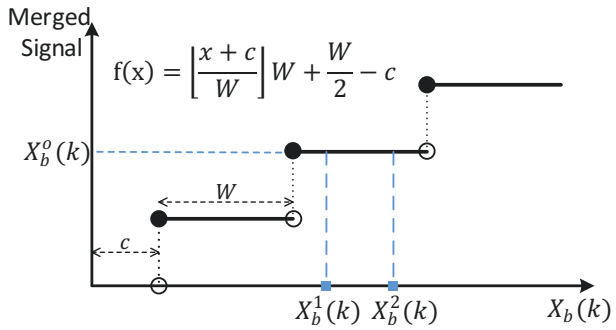


Fig. 6.3. An example of the PWC function $f(x)$ with step size W and horizontal shift c. $f(x)$ merges two quantized coefficients $X_b^1(k)$ and $X_b^2(k)$ into an identical value $X_b^o(k)$.

Considering the I-frame I_i as a target for merging the different reconstructions of view i from multiple different P-frames, for each quantized transform coefficient $X_b^j(k)$ of the k-th frequency in a block b of the reconstruction from P-frame $P_i(j)$, a proper selection of W and c of $f(x)$ can round down the coefficient $X_b^j(k)$ to a value $X_b^o(k)$, *i.e.*

$$X_b^o(k) = \left\lfloor \frac{X_b^j(k)+c}{W} \right\rfloor, \ \forall j \text{ s.t. } P_i(j) \text{ exists}, \tag{6.2}$$

where $X_b^o(k)$ is the quantized k-th frequency in the block b of the I-frame I_i. The proper selection of W and c for each coefficient of each block, which is optimized in a RD optimal manner in [83], is then coded as M-frame. By definition, an M-frame M_i *plus* any decoded $P_i(j)$ will result in an identically reconstructed I_i. See Fig. 6.4 for an illustration. Like I_i, M_i is also pre-encoded by default in our structure.

When a user requests switching from view j to i, the server can either transmit an I-frame I_i, or an M-frame M_i *plus* a P-frame $P_i(j)$. The technical challenge is

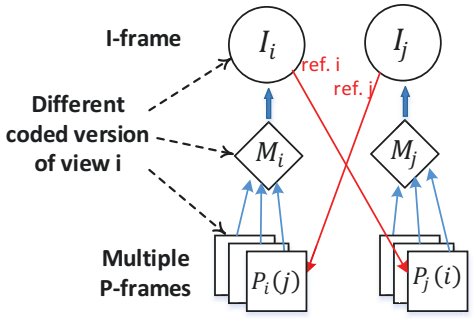


Fig. 6.4. Example for frame types of view i and j, with I-(circle), P-(square) and M-frames (diamonds).

how to select an appropriate set of P-frames for pre-encoding in the frame structure to facilitate user's view-switch requests, when the server's storage size is limited. We focus on this problem in the sequel. We will first introduce how to compute the expected transmission cost given a frame structure θ. We then discuss how to insert landmarks to help design an optimal frame structure θ^* that optimally trades off expected transmission cost and storage cost.

6.3 Expected Transmission Cost

We can compute the expected transmission cost of an ILFS session for a given frame structure θ using a set of recursive equations. To keep the computation tractable, we first describe a Poisson distribution of the lifetime of an ILFS session and then describe a *flexible* one-frame reference buffer model.

6.3.1 A Poisson Distribution of Lifetime

We assume that a user starting an ILFS session at time $t = 0$ at an initial center view s switches its view at each time instant until T view-switches (called *lifetime*) are performed, upon which he exits the session. As often done in lifetime modeling, we assume that random variable T follows a Poisson distribution [101], and the probability of $T = k$ view-switches is

$$p(T = k) = \frac{\mu^k e^{-\mu}}{k!} \tag{6.3}$$

where μ is the expected lifetime of an ILFS session and $k!$ is the factorial of k. Since the Poisson probability will be small when k becomes very large and to avoid an infinity computation, we define a certain large T_o as a threshold of the maximum lifetime. Defining instant t as the t-th view-switch, the probability $g(t)$ that there are at least t view-switches in a session can be computed as

$$\begin{aligned} g(t) &= \mathbf{1}(t \leq T_o) \sum_{k=t}^{T_o} p(T = k) \\ &= \mathbf{1}(t \leq T_o) e^{-\mu} \sum_{k=t}^{T_o} \frac{\mu^k}{k!} \end{aligned} \tag{6.4}$$

where $\mathbf{1}(x)$ is an indicator function that equals 1 if clause x is true and 0 otherwise. Hence $g(t)$ is the total probability that the lifetime T is no smaller than t.

6.3.2 Flexible 1-frame Reference Buffer

A flexible one-frame reference buffer means that, besides a current frame in the display buffer for the user's observation, there is in addition a reference buffer to store one additional frame. When a user observing view i with frame (view) γ in the reference buffer switches to view j, the user can use *either* frame γ *or* frame i as predictor to decode P-frame $P_j(\gamma)$ or $P_j(i)$ for view-switch to j. It means that, if there is a landmark view γ in the user's reference buffer, using the flexible one-frame buffer a user can switch from view i to j by directly decoding $P_j(\gamma)$ without first decoding $P_\gamma(i)$ to reconstruct the landmark view. To facilitate future view switches, the most valuable frame will be selected to be stored in the reference buffer.

Using a flexible one-frame reference buffer, we consider three different transmission types during a view-switch: *0-hop*, *1-hop* and *2-hop* transmissions as follows.

- *0-hop transmission* means that an I-frame I_j is transmitted for the requested view j, resulting in an overhead $r_j^I = |I_j|$, where $|\cdot|$ stands for the coding rate of a frame.

- *1-hop transmission* means that a P-frame $P_j(\gamma)$ or $P_j(i)$ is transmitted with an M-frame M_j, resulting in an overhead $r_j^P(\gamma) = |P_j(\gamma)| + |M_j|$ or $r_j^P(i) = |P_j(i)| + |M_j|$, respectively.

- *2-hop transmission* means that a P-frame $P_\eta(\gamma)$ or $P_\eta(i)$ is first transmitted along with an M-frame M_η to transition to an *intermediate view* η, then P-frame $P_j(\eta)$ and M_j are transmitted to arrive at the designation view j. The overhead is thus $r_\eta^P(\gamma)$ or $r_\eta^P(i)$ *plus* $r_j^P(\eta)$. 2-hop transmission enables a user to first switch to a landmark, which may facilitate future view switching.

6.3.3 Expected Transmission Cost

Given a frame structure θ, we now derive recursive equations to compute the expected minimal transmission cost. Denote by $c_i^{(t)}(\gamma_t)$ the expected transmission cost from current instant t to the end of an ILFS session, given that at instant t a user is at view i and with view γ_t in the reference buffer. We can write $c_i^{(t)}(\gamma_t)$ as a recursive formula:

$$c_i^{(t)}(\gamma_t) = \sum_{j \in \mathcal{N}(i)} p_{j|i} \ \min \left[h_i^{(t)}(\gamma_t, j), \dot{h}_i^{(t)}(\gamma_t, j), \ddot{h}_i^{(t)}(\gamma_t, j) \right] \tag{6.5}$$

where $p_{j|i}$ is the view switching probability from view i to view j, and $h_i^{(t)}(\gamma_t, j)$, $\dot{h}_i^{(t)}(\gamma_t, j)$ and $\ddot{h}_i^{(t)}(\gamma_t, j)$ are transmission cost of a user from current instant t to the end of an ILFS session, given that the user (who is at view i and with view γ_t in the reference buffer at instant t) switches to view j at instant $t+1$ by using 0-hop, 1-hop and 2-hop transmissions, respectively.

The 0-hop transmission cost $h_i^{(t)}(\gamma_t, j)$ is the sum of r_j^I *plus* the recursive cost $c_j^{(t+1)}(\gamma_{t+1})$ if the maximum lifetime T_o has not been reached. The frame γ_{t+1} that will be stored in the reference buffer for the $(t+1)$-th instant should be selected between views γ_t and i, whichever has smaller subsequent transmission cost. Thus $h_i^{(t)}(\gamma_t, j)$ can be written as

$$h_i^{(t)}(\gamma_t, j) = r_j^I + g(t+1) \min_{\gamma_{t+1} \in \{\gamma_t, i\}} c_j^{(t+1)}(\gamma_{t+1}), \tag{6.6}$$

where $g(t+1)$ is computed by Eq. (6.4).

The 1-hop transmission cost is the sum of either $r_j^P(\gamma_t)$ or $r_j^P(i)$ plus the recursive cost $c_j^{(t+1)}(\gamma_{t+1})$. The frame γ_t or i which is used as predictor to view j will become the new reference in the recursive future term. Thus we write $\dot{h}_i^{(t)}(\gamma_t, j)$ as

$$\dot{h}_i^{(t)}(\gamma_t, j) = \min_{\gamma_{t+1} \in \{\gamma_t, i\}} \left[r_j^P(\gamma_{t+1}) + g(t+1) c_j^{(t+1)}(\gamma_{t+1}) \right]. \tag{6.7}$$

We define $r_j^P(\gamma_{t+1}) = \infty$ to signal a violation if P-frame $P_j(\gamma_{t+1})$ is not in the structure θ.

The 2-hop transmission cost is, for an intermediate view η, the sum of either P-frame cost $r_\eta^P(\gamma_t)$ or $r_\eta^P(i)$, plus P-frame cost $r_j^P(\eta)$, plus recursive cost $c_j^{(t+1)}(\eta)$.

$$\ddot{h}_i^{(t)}(\gamma_t, j) = \min_{\eta} \left[r_j^P(\eta) + g(t+1) c_j^{(t+1)}(\eta) + \min_{\tau \in \{\gamma_t, i\}} r_\eta^P(\tau) \right]. \tag{6.8}$$

In this case, the reference frame for $(t+1)$-th view-switch is $\gamma_{t+1} = \eta$.

Examples of 0-hop, 1-hop and 2-hop transmissions for view-switching with flexible 1-frame reference buffer are shown in Fig 6.5. For a user observing view i with reference view γ_t in the buffer at time instant t, he can switch to view j by either 0-hop, 1-hop or 2-hop transmission. If the 0-hop or 1-hop transmission is used, the reference frame γ_{t+1} must be selected between view i and view γ_t. If the 2-hop transmission is used, then the reference frame γ_{t+1} must be the intermediate view η.

Having defined the above, $c_s^{(0)}(\emptyset)$ will compute the expected transmission cost starting from an initial given view s with an empty reference buffer $\emptyset$. The definition of $c_s^{(0)}(\emptyset)$ is similar to (6.5):

$$c_s^{(0)}(\emptyset) = \sum_{j \in \mathcal{N}(s)} p_{j|s} \ \min \left[h_s^{(0)}(\emptyset, j), \dot{h}_s^{(0)}(\emptyset, j), \ddot{h}_s^{(0)}(\emptyset, j) \right]. \tag{6.9}$$

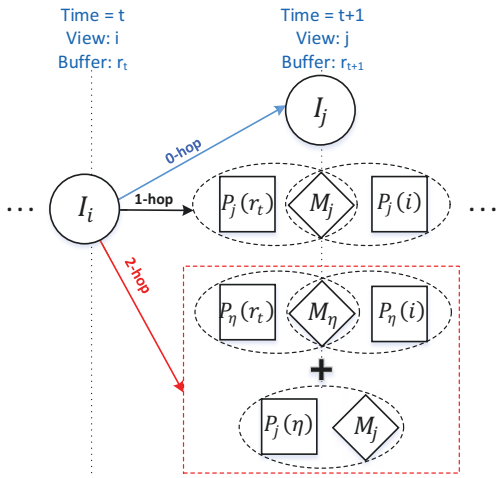


Fig. 6.5. Example transmissions for view-switching with flexible 1-frame reference buffer, with I-, P- and M- frames shown with solid circles, squares and diamonds, respectively.

6.3.4 Complexity Analysis

The complexity of computing $c_s^{(0)}(\emptyset)$ can be analyzed as follows. To avoid re-computation of recurring sub-problems, assuming a DP table is used. The complexity of DP algorithm is upper-bounded by the size of the DP table multiplied with the complexity to compute each table entry. In our case, the size of the DP table is bounded by $O(8N^2T_o)$, where the maximum size of $\mathcal{N}(i)$ for each view i is 8. The steps required to compute (6.5), (6.6), (6.7) and (6.8) are bounded by $O(N)$. Thus, the overall complexity of computing the expected transmission cost is $O(N^3T_o)$, which is in polynomial time.

6.4 Frame Structure Design

6.4.1 Problem Formulation

Given I- and P- frames coded by HEVC [17] and M-frames coded by [83]—all at a sufficiently fine pre-fixed quantization parameter (QP) for a target image quality—our problem is to determine which P-frames to be differentially encoded *a priori* at the server to minimize the weighted sum of expected transmission cost and the storage cost. For a given structure θ, we define the storage cost $b(\theta)$ as the total size

of all the pre-encoded differential P-frames:

$$b(\theta) = \sum_{P_j(i) \in \theta} |P_j(i)| \tag{6.10}$$

where $|P_j(i)|$ is the HEVC coding rate of P-frame $P_j(i)$. I- and M-frames are not considered since they are pre-encoded by default for each view in the structure.

Having defined the expected transmission cost and the storage cost for a given structure θ, we next define the optimal frame design problem: find a frame structure θ^* that optimally trades off the expected transmission cost and the storage cost, *i.e.*,

$$\theta^* = \arg\min_{\theta} c(\theta) + \lambda b(\theta). \tag{6.11}$$

where λ is a given weight parameter and each λ corresponds to one tradeoff of expected transmission cost and storage cost. $c(\theta) = c_s^{(0)}(\emptyset)$ is the expected transmission cost computed by Eq. (6.9) for a given frame structure θ.

6.4.2 Structure Design Algorithm

In this subsection, we design our frame structure. Our basic idea is as follows. We first divide all the views into partitions each associated with a landmark view by assuming that: 1) in each partition the landmark view has a P-frame to and from any other view, and 2) any two landmark views have P-frames from each other. Then we use a greedy algorithm to remove or add some P-frames. This is because it may be beneficial to remove some P-frames within a partition or add some P-frames between two non-landmark views in two different partitions. An example is shown in Fig. 6.6 (to be discussed later).

To minimize (6.11), we propose to initialize a frame structure θ with "landmarks" first. Similar to an airline hub in functionality, by adding P-frames to/from a landmark view for all neighboring views, any view can transition to any other view by decoding essentially only two P-frames (transition to landmark, then to designation view). Further, having multiple landmarks means that the P-frames used to arrive at/depart from a landmark can be smaller; however, the transition

between two views connected to two different landmarks can become costly. The challenge then is to identify the appropriate number and locations of landmarks.

To select landmarks, we adopt a *tree-structure vector quantizer* (TSVQ) method [102], which we define recursively as follows. At a recursive instant, a landmark l is associated with a neighborhood of views or *partition* Ψ; all views in the partition have P-frames to/from the landmark. In terms of two cost function $\phi(\cdot)$ and $\delta(\cdot)$, this instant must decide whether to split the partition Ψ into two sub-partitions Ψ_1 and Ψ_2 (with corresponding landmarks $l_1 \in \Psi_1$ and $l_2 \in \Psi_2$), where $\Psi = \Psi_1 \cup \Psi_2$. Specifically, given that $\phi(\cdot)$ stands for the cost of transitions within a partition and $\delta(\cdot)$ stands for the cost of transitions between two views connected to two different partitions (partition boundary crossing), if

$$\phi(\Psi_1, l_1) + \phi(\Psi_2, l_2) + \delta(\Psi_1, \Psi_2, l_1, l_2) < \phi(\Psi, l), \tag{6.12}$$

it means that the splitting would be more beneficial and hence the instant Ψ will be replaced by two new instants Ψ_1 and Ψ_2 that are also solved recursively following the same splitting strategy. The definition of $\phi(\cdot)$ and $\delta(\cdot)$ and the optimal partition splitting will be discussed in the following.

6.4.2.1 Definition of $\phi(\cdot)$ and $\delta(\cdot)$

One choice of the cost function $\phi(\Psi, l)$ is a version of (6.11) computed for views in a partition Ψ. However, recursively computing expected transmission cost is costly. Hence, we define a simpler cost function $\phi(\Psi, l)$ for views in the partition Ψ as follow:

$$\phi(\Psi, l) = \sum_{i \in \Psi} \left[q_i T \left(r_i^P(l) + r_l^P(i) \right) + \lambda \left(|P_i(l)| + |P_l(i)| \right) \right] \tag{6.13}$$

where q_i is the probability that a view i would be visited during lifetime T transitions (which can be computed in advance). $r_i^P(l)$ is the transmission overhead with a P-frame $P_i(l)$ plus an M-frame M_i. A P-frame $P_i(l)$ or $P_l(i)$ may be requested when a user switches from (to) any other view to (from) view i. Hence,

the $q_i T \left(r_i^P(l) + r_l^P(i) \right)$ is a simple proxy of the expected transmission cost for view switching to/from view i. The $(|P_i(l)| + |P_l(i)|)$ is the corresponding storage cost. Hence, $\phi(\Psi, l)$ defined in (6.13) is a simpler version of (6.11) for view-switches within the partition Ψ.

Consider the case with two partitions Ψ_1 and Ψ_2, to facilitate transitions between views that are connected to two different landmarks, we also add P-frames $P_{l_1}(l_2)$ and $P_{l_2}(l_1)$ that connect between the two landmarks into structure θ, such that a new landmark can be reached and stored in the reference buffer to facilitate future view switching after a 2-hop transmission. For example, given that view i and j are connected to two landmarks l_1 and l_2, respectively, and l_1 is in the reference buffer for view i switching to view j, the two P-frames $P_{l_2}(l_1)$ and $P_j(l_2)$ can enable the user to first switch to landmark l_2 and then switch to view j by 2 hops. It also enables landmark l_2 to be stored in the reference buffer for future view-switches starting from view j. The increased transition cost and storage cost of dealing with partition boundary crossings is then monitored by:

$$\delta(\Psi_1, \Psi_2, l_1, l_2) = \tag{6.14}$$
$$\underbrace{\lambda \, (\, |P_{l_1}(l_2)| + |P_{l_2}(l_1)| \,)}_{\text{term 1}} + \underbrace{\sum_{i \in \Psi_1} \sum_{j \in \mathcal{N}(i) | j \in \Psi_2} \left[q_i T p_{j|i} \cdot r_{l_2}^P(l_1) + q_j T p_{i|j} \cdot r_{l_1}^P(l_2) \right]}_{\text{term 2}},$$

where in term 2 of (6.14), $q_i T p_{j|i}$ computes the expected occurrences of view-switching from view i to view j, during which the P-frame $P_{l_2}(l_1)$ is required for the 2-hop transmission. Note that when view i can switch to view j, *i.e.*, $j \in \mathcal{N}(i)$, we can also get $i \in \mathcal{N}(j)$. Hence, in (6.14), term 1 and term 2 are respective the increased storage cost and expected transmission cost for dealing with partition boundary crossings after splitting a partition into two sub-partitions.

6.4.2.2 Optimal Partition Splitting

After defining the two functions $\phi(\Psi, l)$ and $\delta(\Psi_1, \Psi_2, l_1, l_2)$ for computing the total cost for views in a partition Ψ and for dealing with partition boundary crossings, respectively, we next look at how to optimal split a partition Ψ. To choose

the best landmarks l_1 and l_2 and sub-partitions Ψ_1 and Ψ_2 during splitting, we use the Lloyd's algorithm [100] that iterates between two alternating steps until convergence. First, given sub-partitions $\Psi_k, k \in \{1, 2\}$, we find each locally optimal landmark l_k by minimizing the following:

$$l_k = \arg\min_{l \in \Psi_k} \phi(\Psi_k, l), \tag{6.15}$$

where $\phi(\Psi_k, l)$ is defined in (6.13). In words, an optimal view $l_k \in \Psi_k$ that minimizes the total cost in a partition Ψ_k is selected as a new landmark.

Second, given landmarks l_1 and l_2, we assign each view j in partition Ψ to the closer of the two landmarks:

$$z = \arg\min_{m \in \{1,2\}} |P_j(l_m)| + |P_{l_m}(j)|, \tag{6.16}$$

where z is the partition to which view j is assigned. We iteratively solve (6.15) and (6.16) until convergence. Empirical data show that the Lloyd's algorithm can converge quickly.

Our proposed TSVQ algorithm to optimally select landmarks is operated as follows. Given an initialized partition Ψ with all the views in 2D grids, we first find the landmark l by solving Eq. (6.15). Then, by iterating between (6.15) and (6.16), we optimally split Ψ into two sub-partitions Ψ_1 and Ψ_2, as well as find the two corresponding landmarks l_1 and l_2. Computing Eq. (6.13) and (6.14) based on these partitions and applying the results on Eq. (6.12), if Eq. (6.12) holds, we confirm splitting partition Ψ with two sub-partitions Ψ_1 and Ψ_2. We keep applying the same splitting strategy on partitions Ψ_1 and Ψ_2 until Eq. (6.12) does not hold any more.

6.4.3 Frame Structure Design

Having identified landmarks via TSVQ, we now look at how to find the optimal frame structure using landmarks. We first initialize a frame structure θ in which there are P-frames connecting any view to/from its landmark and P-frames

connecting any two landmarks. Recall that we assume each view has an I-frame in our structure and one can choose from 0-hop, 1-hop and 2-hop transmissions for each view-switching, our initialized frame structure θ may not be the optimal frame structure that best trades off the expected transmission cost and storage cost. Thus, we adopt the greedy algorithm proposed in [81] to iteratively add/remove P-frames to θ using Eq. (6.11) until the object function cannot be further improved. Specifically, at each iterative step, we check each P-frame as follows: if the P-frame is not in θ, we check whether adding the frame to θ reduces the objective cost function; if P-frame is in θ, we check whether removing the frame from θ reduces the objective cost function. Then, at each iteration step, we select the most "beneficial" P-frame (which induces the largest objective cost decrease) to be added or removed. We stop the iteration if no more cost decrease happens. Hence, the optimal frame structure θ^* can be achieved.

During ILFS, with our proposed optimal landmarks insertion and frame structure designing methods, the server first figures out an optimal frame structure and stores the corresponding frames in advance. When a client sends a view-switching request to the server, based on the client's currently watching view and the reference buffer view, the server sends to the client an I-frame, a P-frame *plus* an M-frame , or a pair of P-frames *plus* M-frames which minimizes the expected transmission cost to meet the client's view-switching request. Meanwhile, the server tells the client which view to be stored in the reference buffer.

An example of the designed frame structure θ^* by inserting landmarks is shown in Fig. 6.6, where the dash blue line divides the nine views $\{1, \cdots, 9\}$ into two sub-partitions $\{1, 4, 5, 7, 8\}$ and $\{2, 3, 6, 9\}$ connecting to two different landmarks 4 and 6, respectively. The black arrows stand for the P-frames in θ^*, *e.g.*, the arrow from view 6 to view 2 means that P-frame $P_2(6)$ is in θ^*. To show how ILFS works, we give some view-switching examples starting at instant t in the following.

- when a user requests a view switching from view 5 to view 8 (within a partition), if $\gamma_t = 4$, *i.e.*, the user's reference buffer stores the landmark view 4, the server may send a P-frame $P_8(4)$ for a 1-hop transmission and ask the

user to continue storing view 4 in the reference buffer ($\gamma_{t+1} = 4$) for future; if $\gamma_t \neq 4$, the server may send a pair of P-frame $P_4(5)$ and $P_8(4)$ for a 2-hop transmission and ask the user to store $\gamma_{t+1} = 4$.

- When a user requests switching from view 4 to view 1, view 7 or view 8, no matter what γ_t is, the server may send $P_1(4)$, $P_7(4)$ or $P_8(4)$ for a 1-hop transmission and ask the user to store $\gamma_{t+1} = 4$.

- When a user requests switching from view 5 to view 2 (partition boundary crossing) with reference view $\gamma_t = 4$, the server has two options to send either a P-frame $P_2(5)$ (1-hop) or a pair of P-frames $P_6(4)$ and $P_2(6)$ (2-hop) depending on how far an ILFS session will end after this view-switching. For example, if this is the last view switch, *i.e.,* $t = T_o$, the server may send $P_2(5)$ directly and do not care about γ_{t+1}; if $t < T_o$, to facilitate future view-switching starting from view 2, the server may send P-frames $P_6(4)$ and $P_2(6)$ and ask the user to store $\gamma_{t+1} = 6$.

- When a user requests switching to view 3 starting from view 2 or view 6, since there is no P-frame connecting to view 3, the server should send an I-frame I_3 for a 0-hop transmission; the selection of reference view γ_{t+1} depending on γ_t and the current watching view with Eq. (6.6) (6.7) (6.8).

- When a user requests view-switching from view 8 to view 9, if $\gamma_t = 4$, the server may send P-frames $P_6(4)$ and $P_9(6)$ for a 2-hop transmission and ask the user to store $\gamma_{t+1} = 6$; if $\gamma_t \neq 4$, 1-hop and 2-hop transmission is not applicable, and thus, the server should send the I-frame I_9 for a 0-hop transmission.

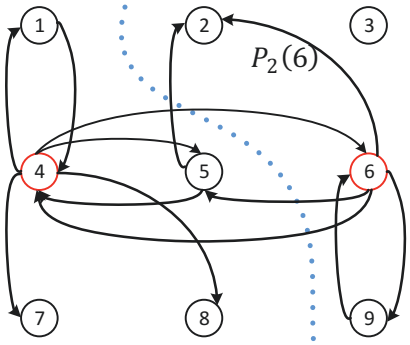


Fig. 6.6. An example of designed frame structure with landmarks.

6.5 Experimentation

6.5.1 Experiment Setup

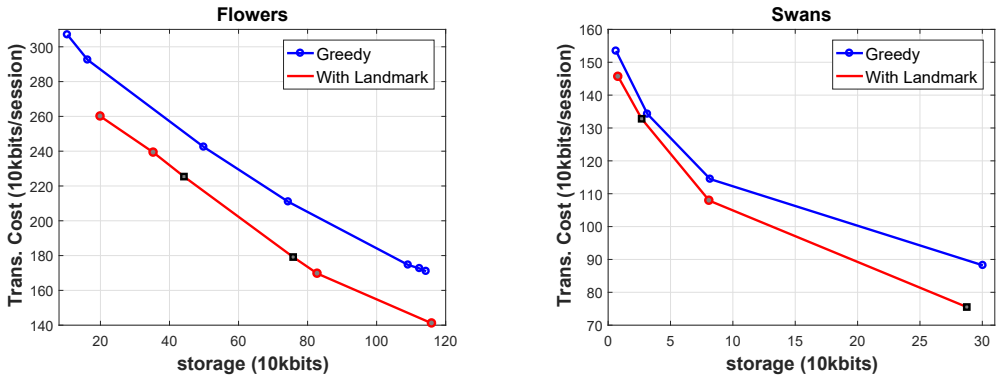
To test the performance of our designed structure using landmarks, we download two LF image sets `Swans` and `Flowers` from [103] [7]. We select a subset of 9×9 2D grid of images for each set, where each image is of size 432×624. We use HEVC HM 15.0 [17] to code I- and P-frames, and use [83] to code M-frames. Quantization parameters are set so that PSNR of the encoded frames are around 36dB.

We compare our proposed method with a greedy algorithm proposed in [81], where a locally optimal single P-frame or pair of P-frames are iteratively added to the structure at a time to reduce the objective function in (6.11). Both these two methods use a flexible 1-frame reference buffer. The maximum lifetime T_o of a session is set to two third of the number of LF images and the expected lifetime μ is set as half of T_o. To optimally insert landmarks with TSVQ method, the lifetime T used in Eq. (6.13) and (6.14) is defined as the expected lifetime. We vary λ in (6.11) to induce different tradeoffs between the expected transmission and storage costs.

6.5.2 Experiment Results

Fig. 6.7 shows plots of expected transmission cost versus the storage cost for our designed structures (red) and the greedy structures [81] (blue). To reach

[7] http://mmspg.epfl.ch/EPFL-light-field-image-dataset



(a) (b)

Fig. 6.7. Expected transmission cost versus storage size of frame structure for `Flowers` and `Swans`.

the same expected transmission cost, our proposed method can save 31.92% and 28.61% storage cost compared to the greedy algorithm for `Flowers` and `Swans`, respectively, using the computation method proposed in [86]. This is because, in our method, a landmark is usually preferred to be stored in the reference buffer. Thus only one P-frame connected from the landmark to the designated view is needed for each view-switch. Even when views switch across partition boundaries, the 2-hop transmission can enable switching from current landmark to a new landmark. However, the greedy algorithm in [81] does not generate landmarks in the frame structure by considering one frame at a time. This is because a landmark view is not useful in reducing the objective until a sufficient number of P-frames to/from neighboring frames are added. The complexity of our proposed method is also much lower than the greedy algorithm in [81], since our initialized frame structure already contains a lot of well-selected P-frames—connecting to/from landmarks and only a minor adjustment to remove/add P-frames is required during the greedy step. The computation expense for greedily removing/adding P-frames is much smaller than the method proposed in [81], where there is no P-frame in the initialized frame structure.

On the red curve in Fig. 6.7 (our proposed method), the black square dots correspond to structures with 2 landmarks, where the other dots correspond to

structures with 1 landmark.[8] When there are more than one landmark in a frame structure, the size of P-frames connecting a landmark to its neighboring views can be smaller since each view always selects a closer landmark. Although multiple landmarks may increase the transmission cost to deal with partition boundary crossings, the smaller size of P-frames will decrease the storage cost and the transmission cost within a partition. With proper selection of multiple landmarks, the total cost saving on smaller P-frames may outweigh the increased transmission cost. Hence, for the black square dots on the red curves, the structures with 2 landmarks work better than structures with 1 landmark.

6.6 Conclusion

To efficiently facilitate ILFS, we design a frame structure composed of pre-encoded I-, P- and M-frames using the idea of landmarks. We use a variant of the Lloyd's algorithm to recursively locate landmark views, then greedily add/remove P-frames subject to a rate-distortion criterion. Experimental results show that our designed frame structure has lower expected transmission costs than structures from a previous proposal [81].

[8] In Fig. 6.7, our proposed method does not generate three or more partitions.

Chapter 7

Conclusion and Future Work

This chapter summarizes the contribution of this thesis and discuss future works.

7.1 Conclusions

The ability of multi-view imaging to capture a static 3D scene simultaneously but from different viewpoints enables users to freely choose their preferred viewpoints for observation. When multiple images/videos are recorded and the 3D scene is represented by both color and depth information, the amount of data that needs to be stored and transmitted is huge. Being the major focus of this thesis, we study two major topics. The first topic is about object approximation, to lower depth image coding rate. A greedy algorithm and a rate-distortion (RD) optimal algorithm are proposed respectively. The second topic is related to behavior analysis of users and servers during multi-view image/video streaming. We first study users' optimal reference view selection problem in a cooperative live free viewpoint streaming system. We then investigate how to efficiently facilitate interactive light field streaming framework with landmarks. The effectiveness and efficiency of our proposed strategies in the two topics are verified numerically.

Specifically, In Chapter 3, a greedy contour approximation method is proposed, where the edge direction with the largest estimated probability computed by arithmetic edge coding (AEC) is selected to reduce depth coding cost. To further control induced synthesized virtual view distortion due to contour approximation, in Chapter 4 we then propose a RD optimal method for object contour approximation.

We first propose a distortion proxy that is an upper bound of the established synthesized view quality metric, 3DSwIM. Given coding rate computed using AEC and our distortion proxy, contours are approximated optimally via DP algorithm in an inter-view consistent manner. Experiments show significant performance gain over previous depth coding schemes.

In Chapter 5, we study the optimal reference view selection problem in a cooperative free viewpoint streaming system, where a user can simultaneously belong to two groups and share the cost of a single reference view within each group separately. Leveraging on the properties of the synthesized video distortion, we propose an efficient search algorithm to study a NE reference selection, which optimally trades off shared streaming reference view cost and synthesized view distortion for users.

In Chapter 6, to efficiently faciliate interactive ILFS framework, we design a frame structure composed of pre-encoded I-, P- and merge frames using the idea of landmarks: for all neighboring views, add P-frames to/from a landmark view means that any view can transition to any other view in the neighborhood by decoding only two P-frames. We use a variant of the Lloyd's algorithm to recursively locate landmark views, then greedily add/subtract P-frames subject to a RD criteria. Experimental results show that our designed frame structure have lower expected transmission cost than structures from a previous greedy proposal.

7.2 Future Work

Given the advent technologies that one can acquire both color and depth images of a static 3D scene at the same time easily, the additional geometric information captured by a depth image can help improve a lot of traditional applications which utilize only 2D color images, such as image/video segmentation, image inpainting, object tracking/recognition, pose estimation, sleep monitoring, *etc.*. Among them, depth assisted front shape segmentation during video conferencing (VC) could be a very interesting research topic. With the advent of networking technologies enabling high-bandwidth and low-delay transmission, VC connecting two parties

separated by large physical distance is now quite ubiquitous. Efficiently segmenting the frontal shape of users who are taking VC will be very useful in applications such as gaze correction [104], face beautification [105], video compression/transmission and so on. Since a depth image shows PWS properties, *i.e.* smooth regions separated by sharp edges, it is important to investigate how this additional geometry information can improve segmentation results.

Recently, virtual reality (VR) or augmented reality (AR) has attracted world-wide research interests. VR typically refers to computer technologies that generate 3D realistic images, sounds and other sensations that can be explored and interacted with by a person. VR leads to new and exciting discoveries in a wide variety of applications which include sports, medicine, video games, cinema and entertainment. Although some VR products, such as VR headset [106] and Vive[1], have been in use presently, the technologies on VR are far from mature. There is still a lot of room for further study. The contents (images and videos) of VR products are usually captured by omnidirectional cameras, also known as 360-degree cameras [107] or VR cameras, that have the ability to capture a 3D scene from all directional viewpoints. The post-processing technologies of these captured multiple viewpoints images/videos, such as compression, streaming, frame rendering and resolution adjusting, are very essential steps to build a VR system and to decrease the latency of building virtual spaces [108]. All of these multi-view image/video processing technologies in VR are very interesting topics to be investigated in the future.

[1]https://www.vive.com/ca/

References

[1] Masayuki Tanimoto, "Overview of free viewpoint television," *Signal Processing: Image Communication*, vol. 21, no. 6, pp. 454–461, 2006.

[2] Dong Tian, Po-Lin Lai, Patrick Lopez, and Cristina Gomila, "View synthesis techniques for 3d video," in *Proceedings of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 2009, vol. 7443, p. 22.

[3] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005.

[4] Ramesh Raskar, Accardi A Oh Sb, and Z Zhang, "Light fields: Present and future," *IEEE CVPR Tutorial*, 2009.

[5] Federica Battisti, Emilie Bosc, Marco Carli, Patrick Le Callet, and Simone Perugia, "Objective image quality assessment of 3d synthesized views," *Signal Processing: Image Communication*, vol. 30, pp. 78–88, 2015.

[6] Akira Kubota, Aljoscha Smolic, Marcus Magnor, Masayuki Tanimoto, Tsuhan Chen, and Cha Zhang, "Multiview imaging and 3dtv," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10–21, 2007.

[7] Masayuki Tanimoto, Mehrdad Panahpour Tehrani, Toshiaki Fujii, and Tomohiro Yendo, "Free-viewpoint tv," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 67–76, 2011.

[8] Edward H Adelson and John YA Wang, "Single lens stereo with a plenoptic camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 99–106, 1992.

[9] Ramesh Raskar, Amit Agrawal, Cyrus A Wilson, and Ashok Veeraraghavan, "Glare aware photography: 4d ray sampling for reducing glare effects of camera lenses," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 56, 2008.

[10] Donghyeon Cho, Sunyeong Kim, and Yu-Wing Tai, "Consistent matting for light field images," in *Proceedings of European Conference on Computer Vision*. Springer, 2014, pp. 90–104.

[11] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon, "Accurate depth map estimation from a lenslet light field camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1547–1555.

[12] Changil Kim, Henning Zimmer, Yael Pritch, Alexander Sorkine-Hornung, and Markus H Gross, "Scene reconstruction from high spatio-angular resolution light fields.," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 73–1, 2013.

[13] Daniel Scharstein, Richard Szeliski, and Ramin Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Proceedings of IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV)*. IEEE, 2001, pp. 131–140.

[14] Sergi Foix, Guillem Alenya, and Carme Torras, "Lock-in time-of-flight (tof) cameras: A survey," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, 2011.

[15] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[16] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[17] Gary J Sullivan, J-R Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[18] Wei Hu, Gene Cheung, Antonio Ortega, and Oscar C Au, "Multiresolution graph fourier transform for compression of piecewise smooth images," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 419–433, 2015.

[19] 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.

[20] 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 IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010.

[21] W. Hu, G. Cheung, X. Li, and O. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *Proceedings of IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[22] Y.-H. Chao, A. Ortega, W. Hu, and G. Cheung, "Edge-adaptive depth map coding with lifting transform on graphs," in *Proceedings of 31st Picture Coding Symposium*, Cairns, Australia, May 2015.

[23] Matthieu Maitre, Yoshihisa Shinagawa, and Minh N Do, "Wavelet-based joint estimation and encoding of depth-image-based representations for free-

viewpoint rendering," *IEEE Transactions on Image Processing*, vol. 17, no. 6, pp. 946–957, 2008.

[24] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P. de With, and T. Wiegand, "The effects of multiview depth video compression on multiview rendering," in *Signal Processing: Image Communication*, 2009, vol. 24, pp. 73–88.

[25] Gene Cheung, Vladan Velisavljevic, and Antonio Ortega, "On dependent bit allocation for multiview image coding with depth-image-based rendering," *IEEE Transactions on Image Processing*, vol. 20, no. 11, pp. 3179–3194, 2011.

[26] G. Owen, *Game Theory*, Academic Press, 3rd edition, 1995.

[27] Prashant Ramanathan and Bernd Girod, "Random access for compressed light fields using multiple representations," in *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, Siena, Italy, September 2004.

[28] A. Aaron, P. Ramanathan, and Bernd Girod, "Wyner-Ziv coding of light fields for random access," in *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, Siena, Italy, September 2004.

[29] W. Cai, G. Cheung, T. Kwon, and S.-J. Lee, "Optimized frame structure for interactive light field streaming with cooperative cache," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Barcelona, Spain, July 2011.

[30] Wei Cai, Gene Cheung, Sung-Ju Lee, and Taekyoung Kwon, "Optimal frame structure design using landmarks for interactive light field streaming," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 1445–1448.

[31] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al., "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.

[32] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map distortion analysis for view rendering and depth coding," in *Proceedings of IEEE International Conference on Image Processing*, Cairo, Egypt, November 2009.

[33] I. Daribo and B. Pesquet-Popescu, "Depth-aided image inpainting for novel view synthesis," in *Proceedings of IEEE Multimedia Signal Processing Workshop*, Saint-Malo, France, October 2010.

[34] K. Oh, S. Yea, and Y.-S. Ho, "Hole-filling method using depth based in-painting for view synthesis in free viewpoint television and 3D video," in *Proceedings of 27th Picture Coding Symposium*, Chicago, IL, May 2009.

[35] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *Proceedings of IEEE International Conference on Image Processing*, San Antonio, TX, October 2007.

[36] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[37] Bing Zeng and Jingjing Fu, "Directional discrete cosine transforms for image coding," in *Proceedings of IEEE International Conference on Multimedia and Expo*. IEEE, 2006, pp. 721–724.

[38] Y. Morvan, P.H.N. de With, and D. Farin, "Platelets-based coding of depth maps for the transmission of multiview images," in *Proceedings of SPIE Stereoscopic Displays and Applications*, San Jose, CA, January 2006.

[39] A. Sanchez, G. Shen, and A. Ortega, "Edge-preserving depth-map coding using graph-based wavelets," in *Proceedings of the 43th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 2009.

[40] Byung Tae Oh, Jaejoon Lee, and Du-sik Park, "Depth map coding based on synthesized view distortion function," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 7, pp. 1344–1352, 2011.

[41] Yun Zhang, Sam Kwong, Long Xu, Sudeng Hu, Gangyi Jiang, and C-C Jay Kuo, "Regional bit allocation and rate distortion optimization for multiview depth video coding with view synthesis distortion model," *IEEE Transactions on Image Processing*, vol. 22, no. 9, pp. 3497–3512, 2013.

[42] Hui Yuan, Sam Kwong, Ju Liu, and Jiande Sun, "A novel distortion model and lagrangian multiplier for depth maps coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 3, pp. 443–451, 2014.

[43] Karsten Müller, Heiko Schwarz, Detlev Marpe, Christian Bartnik, Sebastian Bosse, Heribert Brust, Tobias Hinz, Haricharan Lakshman, Philipp Merkle, Franz Hunn Rhee, et al., "3d high-efficiency video coding for multi-view video and depth data," *IEEE Transactions on Image Processing*, vol. 22, no. 9, pp. 3366–3378, 2013.

[44] Gerhard Tech, Ying Chen, Karsten Müller, Jens-Rainer Ohm, Anthony Vetro, and Ye-Kui Wang, "Overview of the multiview and 3d extensions of high efficiency video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 35–49, 2016.

[45] Philipp Merkle, Karsten Müller, Detlev Marpe, and Thomas Wiegand, "Depth intra coding for 3d video based on geometric primitives," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 3, pp. 570–582, 2016.

[46] V Ralph Algazi, Gary E Ford, Adel I El-Fallah, and Robert R Estes Jr, "Preprocessing for improved performance in image and video coding," in *Proceedings of International Symposium on Optical Science, Engineering, and Instrumentation*. International Society for Optics and Photonics, 1995, pp. 22–31.

[47] Inkyeom Kim, Yeonsik Jeong, and Kyu Tae Park, "The block-based preprocessing system for the coding performance improvement," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 1048–1053, 1998.

[48] Ke Shen, "Video preprocessing techniques for real-time video compression-noise level estimation techniques," in *Proceedings of the 42nd Midwest Symposium on Circuits and Systems*. IEEE, 1999, vol. 2, pp. 778–781.

[49] Mao-quan Li and Zheng-quan Xu, "An adaptive preprocessing algorithm for low bitrate video coding," *Journal of Zhejiang University SCIENCE A*, vol. 7, no. 12, pp. 2057–2062, 2006.

[50] Colin Doutre and Panos Nasiopoulos, "Color correction preprocessing for multiview video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 9, pp. 1400–1406, 2009.

[51] Sebastiaan Van Leuven, Glenn Van Wallendael, Robin Ballieul, Jan De Cock, and Rik Van de Walle, "Improving the coding performance of 3d video using guided depth filtering," in *Proceedings of IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2015, pp. 60–61.

[52] Herbert Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers*, , no. 2, pp. 260–268, 1961.

[53] Alexander Akimov, Alexander Kolesnikov, and Pasi Fränti, "Lossless compression of map contours by context tree modeling of chain codes," *Pattern Recognition*, vol. 40, no. 3, pp. 944–952, 2007.

[54] Yong Kui Liu and Borut Žalik, "An efficient chain code with huffman coding," *Pattern Recognition*, vol. 38, no. 4, pp. 553–557, 2005.

[55] Lele Zhou and Saif Zahir, "A new efficient context-based relative-directional chain coding," in *Proceedings of IEEE International Symposium on Signal Processing and Information Technology*, 2006, pp. 787–790.

[56] Herbert Freeman, "Application of the generalized chain coding scheme to map data processing.," pp. 220–226, 1978.

[57] Ian H Witten, Radford M Neal, and John G Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[58] Ismael Daribo, Gene Cheung, and Dinei Florencio, "Arithmetic edge coding for arbitrarily shaped sub-block motion prediction in depth video compression," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2012, pp. 1541–1544.

[59] Ismael Daribo, Dinei Florencio, and Gene Cheung, "Arbitrarily shaped motion prediction for depth video compression using arithmetic edge coding," *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4696–4708, 2014.

[60] Aggelos K Katsaggelos, Lisimachos P Kondi, Fabian W Meier, Jörn Ostermann, and Guido M Schuster, "Mpeg-4 and rate-distortion-based shape-coding techniques," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1126–1154, 1998.

[61] B-J Yun, S-W Lee, and S-D Kim, "Vertex adjustment method using geometric constraint for polygon-based shape coding," *Electronics Letters*, vol. 37, no. 12, pp. 754–755, 2001.

[62] Chung-Ming Kuo, Chaur-Heh Hsieh, and Yong-Ren Huang, "A new adaptive vertex-based binary shape coding technique," *Image and Vision Computing*, vol. 25, no. 6, pp. 863–872, 2007.

[63] Zhongyuan Lai, Junhuan Zhu, Zhou Ren, Wenyu Liu, and Baolan Yan, "Arbitrary directional edge encoding schemes for the operational rate-distortion optimal shape coding framework," in *Proceedings of Data Compression Conference (DCC)*. IEEE, 2010, pp. 20–29.

[64] Zhongyuan Lai, Fan Zhang, and Weisi Lin, "Operational rate-distortion shape coding with dual error regularization," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 5547–5550.

[65] Mei-Chen Yeh, Yen-Lin Huang, and Jia-Shung Wang, "Scalable ideal-segmented chain coding," in *Proceedings of IEEE International Conference on Image Processing*. IEEE, 2002, vol. 1, pp. I–197.

[66] Saif Zahir, Kal Dhou, and BC Prince George, "A new chain coding based method for binary image compression and reconstruction," *PCS, Lisbon, Portugal*, pp. 1321–1324, 2007.

[67] P. Merkle, C. Bartnik, K. Muller, D. Marpe, and T. Weigand, "3D video: Depth coding based on inter-component prediction of block partitions," in *Proceedings of 2010 Picture Coding Symposium*, Krakow, Poland, May 2012.

[68] I. Daribo, D. Florencio, and G. Cheung, "Arbitrarily shaped sub-block motion prediction in texture map compression using depth information,"

in *Proceedings of 2012 Picture Coding Symposium*, Krakow, Poland, May 2012.

[69] Engin Kurutepe, M Reha Civanlar, and A Murat Tekalp, "Client-driven selective streaming of multiview video for interactive 3dtv," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1558–1565, 2007.

[70] H. Huang, B. Zhang, G. Chan, G. Cheung, and P. Frossard, "Coding and replication co-design for interactive multiview video streaming," in *Proceedings of mini-conference in IEEE INFOCOM*, Orlando, FL, March 2012.

[71] H. Huang, B. Zhang, G. Chan, G. Cheung, and P. Frossard, "Distributed content replication for multiple movies in interactive multiview video streaming," in *Proceedings of 19th International Packet Video Workshop*, Munich, Germany, May 2012.

[72] B. Macchiavello, C. Dorea, M. Hung, G. Cheung, and W. t. Tan, "Reference frame selection for loss-resilient depth map coding in multiview video conferencing," in *Proceedings of IS&T/SPIE Visual Information Processing and Communication Conference*, Burlingame, CA, January 2012.

[73] B. Macchiavello, C. Dorea, M. Hung, G. Cheung, and W. t. Tan, "Reference frame selection for loss-resilient texture & depth map coding in multiview video conferencing," in *Proceedings of IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[74] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for P2P streaming systems," in *IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[75] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P iptv system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[76] Y. Liu, Y. Guo, and Chao Liang, "A survey on Peer-to-Peer video streaming systems," *Journal of Peer-to-Peer Networking and Applications, by Springer New York*, Feb. 2008.

[77] Marta Karczewicz and Ragip Kurceren, "The sp-and si-frames design for h. 264/avc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 637–644, 2003.

[78] Chuo-Ling Chang and Bernd Girod, "Rate-distortion optimized interactive streaming for scalable bitstreams of light fields," in *Proceedings of SPIE*, 2004, vol. 5308, pp. 222–233.

[79] Prashant Ramanathan, Mark Kalman, and Bernd Girod, "Rate-distortion optimized interactive light field streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 4, pp. 813–825, 2007.

[80] Zhun Han, Qionghai Dai, and Yebin Liu, "A sender-driven time-stamp controlling based dynamic light field streaming service," in *Electronic Imaging*. International Society for Optics and Photonics, 2007, pp. 650827–650827.

[81] Benedicte Motz, Gene Cheung, and Antonio Ortega, "Redundant frame structure using m-frame for interactive light field streaming," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1369–1373.

[82] N.-M. Cheung, A. Ortega, and G. Cheung, "Distributed source coding techniques for interactive multiview video streaming," in *Proceedings of 27th Picture Coding Symposium*, Chicago, IL, May 2009.

[83] Wei Dai, Gene Cheung, Ngai-Man Cheung, Antonio Ortega, and Oscar C Au, "Merge frame design for video stream switching using piecewise constant functions," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3489–3504, 2016.

[84] T. Maugey, I. Daribo, G. Cheung, and P. Frossard, "Navigation domain partitioning for interactive multiview imaging," in *Special Issue on 3D Video Representation, Compression, Rendering, IEEE Transactions on Image Procesing*, September 2013, vol. 22, no.9, pp. 3459–3472.

[85] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithm*, The MIT Press, third edition, 2009.

[86] Gisle Bjontegaard, "Calcuation of average psnr differences between rd-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.

[87] Gisle Bjontegaard, "Improvements of the bd-psnr model," in *ITU-T SC16/Q6, 35th VCEG Meeting, Berlin, Germany, July 2008*, 2008.

[88] Frederic Dufaux and Fabrice Moscheni, "Motion estimation techniques for digital tv: A review and a new contribution," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858–876, 1995.

[89] M. Solh, G. AlRegib, and J. Bauza, "3vqm: A vision-based quality measure for dibr-based 3d videos," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2011.

[90] C Sidney Burrus, Ramesh A Gopinath, and Haitao Guo, "Introduction to wavelets and wavelet transforms," 1997.

[91] Hubert W Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[92] Gert Van de Wouwer, Paul Scheunders, and Dirk Van Dyck, "Statistical texture characterization from discrete wavelet representations," *IEEE Transactions on Image Processing*, vol. 8, no. 4, pp. 592–598, 1999.

[93] Søren Johansen and Katarina Juselius, "Maximum likelihood estimation and inference on cointegrationâĂŤwith applications to the demand for money," *Oxford Bulletin of Economics and statistics*, vol. 52, no. 2, pp. 169–210, 1990.

[94] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on image processing*, vol. 13, no. 9, pp. 1200–1212, 2004.

[95] Payman Aflaki, Dmytro Rusanovskyy, and Miska M Hannuksela, "Undo dancer 3dv sequence for purposes of 3dv standardization," *ISO/IEC JTC1/SC29/WG11, Doc. M*, vol. 20028, 2011.

[96] D. Ren, S.-H. G. Chan, G. Cheung, V. Zhao, and P. Frassard, "Collaborative P2P streaming of interactive live free viewpoint video," in *arXiv.org*, November 2012, http://arxiv.org/abs/1211.4767.

[97] W. Mark, L. McMillan, and G. Bishop, "Post-rendering 3D warping," in *Symposium on Interactive 3D Graphics*, New York, NY, April 1997.

[98] I. Ahn and C. Kim, "Depth-based disocclusion filling for virtual view synthesis," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Melbourne, Australia, July 2012.

[99] Smarti Reel, Gene Cheung, Patrick Wong, and Laurence S Dooley, "Joint texture-depth pixel inpainting of disocclusion holes in virtual view synthesis," in *Proceedings of Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*. IEEE, 2013, pp. 1–7.

[100] Allen Gersho and Robert M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[101] John Frank Charles Kingman, *Poisson processes*, Wiley Online Library, 1993.

[102] Pasi Fränti, Timo Kaukoranta, and Olli Nevalainen, "On the splitting method for VQ codebook generation," *Optical Engineering*, vol. 36, no. 11, pp. 3043–3051, 1997.

[103] Martin Rerabek and Touradj Ebrahimi, "New light field image dataset," in *Proceedings of the 8th International Conference on Quality of Multimedia Experience (QoMEX)*, 2016, number EPFL-CONF-218363.

[104] Xianming Liu, Gene Cheung, Deming Zhai, Debin Zhao, Hiroshi Sankoh, and Sei Naito, "Joint gaze-correction and beautification of dibr-synthesized human face via dual sparse coding," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 4697–4701.

[105] Xianming Liu, Gene Cheung, Deming Zhai, and Debin Zhao, "Sparsity-based joint gaze correction and face beautification for conferencing video," in *Proceedings of Visual Communications and Image Processing (VCIP)*. IEEE, 2015, pp. 1–4.

[106] Palmer Luckey, Brendan Iribe Trexler, Graham England, and Jack McCauley, "Virtual reality headset," Mar. 18 2014, US Patent D701,206.

[107] Joseph J Mazzilli, "360° automobile video camera system," Dec. 25 2001, US Patent 6,333,759.

[108] Daisuke Ochi, Yutaka Kunita, Akio Kameda, Akira Kojima, and Shinnosuke Iwaki, "Live streaming system for omnidirectional video," in *Virtual Reality (VR), 2015 IEEE*. IEEE, 2015, pp. 349–350.