Coding and Streaming System Design for Interactive 360° Video Applications and Scalable Octree-based Point Cloud Coding

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Yixiang Mao

May 2022

Coding and Streaming System Design for Interactive 360° Video Applications and Scalable Octree-based Point Cloud Coding

---

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Electrical Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Yixiang Mao

May 2022

Approved:

Ivan Selesnick

Department Chair Signature

May 12, 2022

Date

University ID:	**N10158903**
Net ID:	**ym1496**

Approved by the Guidance Committee:

Major: Electrical and Computer Engineering

Wang Yao

Yao Wang
Professor
Electrical and Computer Engineering

5/12/2022
Date

[illegible]

Yong Liu
Professor
Electrical and Computer Engineering

05/12/2022
Date

S. S. Panwar

Shivendra S. Panwar
Professor
Electrical and Computer Engineering

5/12/2022
Date

Vita

Yixiang Mao was born in Anhui, China in 1994. He received his B.S. degree in Physics from Peking Univerisity, China in 2016. He received his M.S. degree in Electrical Engineering from Tandon School of Engineering of New York University in 2018. He started his doctoral training in Tandon School of Engineering of New York University in Fall 2018.

His PhD research focuses are coding and streaming for 360$\circ$ video, point cloud compression, video processing, computer vision, and machine learning. During his PhD years, he also interned at Tencent America and Apple Inc. on video related projects.

Acknowledgements

My PhD study and this thesis would not be possible without the help from people who provided both academic and moral support over the years.

First and foremost, I would like to thank my PhD advisor, Prof. Yao Wang, for her invaluable vision, support, and guidance throughout my doctoral studies. I would like to thank Prof. Yong Liu for the guidance and insightful advice on computer networking and media streaming design throughout the collaborated projects. I would also like to thank Prof. Shivendra S. Panwar for being my PhD guidance committee member, and for his insightful comments and suggestions.

Furthermore, I would like to thank my research collaborators: Fanyi Duanmu, Liyang Sun, Yueyu Hu, and Tongyu Zong.

Last but not least, I would like to express my sincere gratitude to my parents for their support throughout my life. I would also like to thank my girlfriend and my friends for their moral support and companionship.

Yixiang Mao, New York University, Tandon School of Engineering

May 12, 2022

ABSTRACT

Coding and Streaming System Design for Interactive 360° Video Applications and Scalable Octree-based Point Cloud Coding

by

Yixiang Mao

Advisor: Prof. Yao Wang

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Electrical Engineering)

May 2022

Efficient coding and streaming of 360-degree video and point cloud video are critical for the continued development of lifelike virtual reality (VR) experiences.

Interactive 360-degree video applications, e.g. video conferencing, require an extremely low delay in video delivery and robustness to both network dynamics and field of view (FoV) prediction errors. We propose a frame-level FoV-adaptive coding structure that varies the bit rates for different regions of a coded frame based on the predicted FoV. Integrating such frame-level FoV adaptation with temporal predictive coding is challenging due to the temporal variations of the FoV. We propose novel ways for modeling the influence of FoV dynamics on the quality-rate performance of temporal predictive coding. Compared with other benchmark systems, our system shows significantly improved rendered video quality, while achieving very low end-to-end delay and low frame-freeze probability.

Octree-based point cloud representation and compression have been adopted by the MPEG G-PCC standard. However, it only uses handcrafted methods to predict the probability that a leaf node is non-empty, which is used for entropy coding. We propose a 3D convolution-based machine learning model to predict such probabilities for geometry coding using the context information from the previous and currently coded octree level. We further propose a convolution-based model to upsample the decoded point cloud at a coarse resolution on the decoder side. Integration of the two approaches significantly improves the octree-based geometry coding performance. A key advantage of our work from the prior related studies is that our octree-based entropy coding model is naturally scalable. This benefits the future design of the point cloud streaming system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The application and the technology of Virtual Reality (VR) and Augmented Reality (AR) is growing fast in recent years. One approach to facilitate the VR experience is using the 360° video, where the video is recorded in every direction at the same time, and the viewer has control of the viewing in any direction during playback. The users viewing the 360° video usually have 3 degrees of freedom (3-DoF, or yaw, pitch, and raw). However, 6DoF is preferred for a true VR experience, where the user also moves the physical locations in addition to changing the view angles. In this case, the 3D modeling of the space is needed, and point cloud representation is the foundation of 3D models for both VR and AR applications. Point cloud is a set of data point in the space, where each point has a coordinate in X, Y, Z and other attributes information (e.g. color, or transparency). Our study of efficiently coding and streaming for both the 360° video and the point cloud is critical to help the development of next-level VR and AR experiences.

1.2 Introduction to 360° Video Interactive Streaming

Effectively coding and streaming 360° video is critically important for VR and AR applications. However, to achieve similar viewing quality, the required network bandwidth for sending the omnidirectional video is much higher than that required for the traditional planar video. For example, the 360° video at 24K (23040×11520 pixels) resolution provides a similar premium quality as 8K (7680×4320 pixels) planar video. Previous study [1] showed that transmitting such high-resolution video at 120 frames per second (fps) easily consumes Gigabits-per-second bandwidth. *FoV-adaptive streaming* is an effective way to reduce such high bandwidth requirement [2] [3] [4]. In a 360° video session, a viewer only watches the content within a limited Field-of-View (FoV) at any time, which is a small portion of the 360° scope. [5] A viewer's FoV in each frame can be predicted using various methods. FoV-adaptive streaming leveraging tile-based coding refers to the strategy that encodes and transmits the tiles inside the predicted FoV at premium quality, while discarding or encoding and transmitting the remaining tiles at significantly lower quality. Such strategy has been widely developed and evaluated in 360° video-on-demand systems [6–13,13–15] and 360° live streaming systems [16–21]. However, this approach has not been sufficiently investigated for interactive applications, such as VR cloud gaming, VR video conferencing, and AR remote collaboration, etc. [22–24]. The major challenges in interactive applications can be summarized as:

1. The 360° video must be coded and transmitted in real-time with extremely low latency (e.g. $\leq$100ms). Hence, the 360° video should be coded and delivered at the frame level.

2. The periodic intra-frame structure is not suitable for the interactive applications, because it leads to periodic rate spikes and consequently increased delay.

3. The coded regions in each frame depend on the predicted FoV. User FoV in successive frames are often not aligned, and this brings challenges to integrate temporal predictive coding and to accurately estimate the rate during streaming.

To address those challenges, we propose a novel low-latency FoV-adaptive coding and streaming solution for interactive 360° video. The contribution of our work can be summarized as:

1. We adopt motion-compensated temporal predictive coding to maximize the quality-rate efficiency for the predicted FoV region and plus a surrounding border.

2. To reduce the system latency, we propose using a rotating-intra region in each frame to replace the periodic intra-frame.

3. We explicitly model the impact of the misalignment of coded regions on the quality-rate performance of temporal predictive coding, to enable accurate rate allocation among different regions.

4. We design a push-based frame delivery scheme that periodically adapts the sizes and the target bit rates of different coding regions at both the segment level and the frame level.

5. We adopt deep learning models for both FoV prediction and bandwidth prediction.

1.3 Introduction to Point Cloud Geometry Coding

Efficient coding of point cloud data is critical for the continued development of lifelike virtual reality (VR) experiences. For geometry-based point cloud compression, in order to overcome the memory inefficiency from using uniform voxel grids, the 3D space is typically recursively subdivided into smaller cubes using octree where only non-empty nodes are further subdivided [25]. The octree coding mode in in the MPEG G-PCC standard uses a handcrafted context table for context-based entropy coding [26].

To better explore the context information with deep learning models, while still using the octree representation, we propose a novel entropy coding method for the octree-based geometry coding. Our contribution can be summarized as:

1. To predict the probability that 8 children nodes of each non-empty parent node are occupied , we propose a novel way to form an initial "noise" context cube on the octree level of children nodes with values of 0, 1, or 0.5.

2. We propose to denoise the values in the cube using a 3D convolution-based neural network and use the output values of the denoised cube at the center $2 \times 2 \times 2$ voxels as the predicted probability.

3. We further propose a 3D convolution-based neural network to upsample the decoded point cloud if the decoder receives a bit-stream only including a coarse (lossy) octree representation.

4. We also develop alternative probability estimation and upsampling models with significantly less complexity while maintaining comparable coding efficiency.

5. Compared with G-PCC [26] (using a handcrafted entropy model) and VoxContext-Net [27] (using a machine learning method), our methods significantly improve the quality of the reconstructed point cloud.

6. Our octree-based entropy coding model is naturally scalable and this scalable coding setting benefits the future design of streaming systems.

1.4 Organization of the thesis

Chapter 2 introduces the proposed 360° video coding framework and the optimal tile size, and formulates the expected rendered quality considering FoV hit rates and the frame delivery rate, and also describes the optimization problem to maximize the expected rendered quality. Chapter 3 describes the proposed FoV-adaptive streaming system for interactive applications, including the deep learning models used for bandwidth prediction and FoV prediction, the adaptation of region size and rate allocation at the segment level and the bit budget adjustment at the frame level, the setup of the trace-driven simulations, and the system performance comparison with benchmarks using intra- or inter- coding. Chapter 4 describes our proposed octree-based scalable point cloud geometry coding with the learned entropy model and resolution enhancement model. The last chapter summarizes the contributions and takeaways of our works.

Chapter 2

Frame-level FoV-adaptive 360° Video Coding using Spatial and Temporal Prediction

2.1 Introduction

A major challenge in interactive applications is that 360° video must be coded and transmitted in real-time with extremely low latency (e.g. $\leq$100ms). **To achieve low latency, 360° video should be coded and delivered at the frame level, and each frame should consume similar rates.** To accomplish this, previous work on interactive 360° video [22] [28] intra-code all frames without using motion-compensated temporal prediction. A big drawback of using such intra-only coding mode is that it significantly reduces the coding efficiency, which translates to significantly lower video quality under the same bandwidth. On the other hand, integrating temporal predictive coding with frame-level FoV adaptation faces the following challenges:

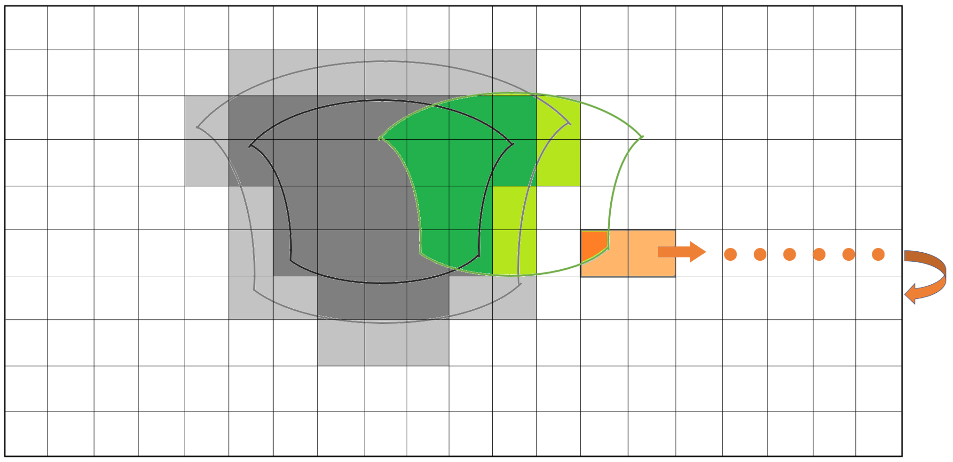

Figure 2.1: The tiled ERP frame and different coding regions. Dark grey: tiles to cover the PF region, coded at the rate R_e. Light grey and orange: tiles to cover PF+ and RI, coded at the rate R_b. Green: user's actual FoV, which may intersect with PF, PF+, RI, and un-coded tiles.

1. The periodic intra-frame structure is not suitable for the interactive applications, because it leads to periodic rate spikes and consequently increased delay. On the other hand, it is important to periodically update the entire 360° scope to limit error propagation after a frame loss due to transmission errors, and to mitigate the quality degradation in un-coded regions, which in turn affects temporal prediction accuracy for future frames.

2. The coded regions in each frame depend on the predicted FoV. User FoV in successive frames are often not aligned. Such misalignment causes prolonged time lapse for temporal prediction for some tiles and leads to reduced coding efficiency. How to properly consider such reduced coding efficiency is important for accurate rate control and essential for minimizing the latency, while optimizing for the rendering quality.

We propose a novel low-latency FoV-adaptive coding and streaming solution for interactive 360° video. We adopt motion-compensated temporal predictive coding to maximize the quality-rate efficiency, and address the challenges

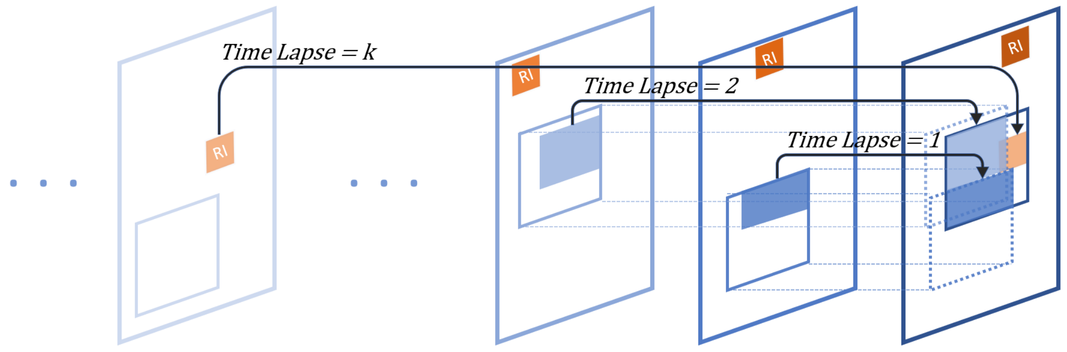

Figure 2.2: Variable time lapses between the coded tiles inside the PF and PF+ regions. The frame on the right is the current frame, its previous frames are on its left. The square region covered by a solid-line border in each frame indicates the coded region in that frame. Different tiles in the coded region in the current frame have different time lapses to the latest frame when the corresponding tiles were coded.

brought by temporal prediction in frame-wise FoV-adaptive coding. To reduce the system latency, we propose using a rotating-intra region in each frame to replace the periodic intra-frame. We also explicitly model the impact of the misalignment of coded regions on the quality-rate performance of temporal predictive coding, to enable accurate rate allocation among different regions. The salient features of our proposed solution include:

1. To achieve low latency, the sender codes and transmits video at the granularity of frames, instead of segments (e.g. groups of pictures) commonly used for video-on-demand and live streaming.

2. For each new frame, the sender predicts the FoV of the receiver, and codes only a region covering the predicted FoV (denoted PF) plus a surrounding border (denoted PF+), with the border size adapted to the anticipated FoV prediction errors. Both regions will be coded using temporal prediction

but at different rates. [1]

3. We code a small rotating region using intra-coding for each frame using spatial prediction only, enabling gradual refreshment of the entire 360° scope after a certain period. See Fig. 2.1. Such rotating-I (RI) regions reduce the frame size burstiness and hence reduces delay, while providing robustness against FoV prediction errors and frame losses.

4. While modeling the quality-rate (Q-R) relations of coded regions, we take into account the spatially and temporally varying time lapses in temporal prediction due to FoV dynamics (see Fig. 2.2). We further model the decay of the rendering quality of non-coded regions as a function of the time lapse since these regions were last coded.

In this chapter, Sec. 2.2 introduces the proposed video coding framework with three coding regions. Sec. 2.3 presents the experiment of choosing the optimal tile size that trades off the coding efficiency and the unused coded area. Sec. 2.4 explores the quality-rate models, including the "ideal" quality-rate models, and the rate-increase and the quality-decay factor. Sec. 2.5 formulates the expected rendered quality considering FoV hit rates and the frame delivery rate, and also describes the process to optimize the region size and rate.

2.2 The Proposed Video Coding Scheme

A novel FoV-adaptive coding structure is proposed to replace the conventional group of pictures (GOP) structure (periodic intra frames). We use the Equirectangular projection (ERP) to present the 360° video. In the proposed structure,

[1]When a remote site has multiple participants, we can take the union of the FoVs of all the participants as the ground truth "FoV" of this site, and predict the future FoV union.

only the first frame of a video stream was encoded entirely using spatial prediction (i.e. intra-coding) only. For each following frame, we first predict the client's FoV at that time. As illustrated in Fig. 2.1, we then code the predicted FoV region (called "PF") and a small border around it (called "PF+") on the ERP using temporal prediction (i.e. inter-coding) based on the previously decoded reference frame. The border region is coded in case the PF is slightly off from the client's actual FoV. The normalized bit rate (in terms of bits/degree^2) allocated for the PF+ is lower than that for the PF. In addition to the PF and PF+, we also code and send a rotating-I region (called "RI") using spatial prediction only, to ensure all pixels on the EPR will be refreshed by intra coding at a certain frequency. For each successive frame, the intra-coded RI region rolls to a new location on the EPR (from top to bottom and left to right). The RI region refreshes all pixels on the ERP at a certain rolls in successive frames from top to bottom and left to right in the ERP. The RI is introduced to ensure that all pixels in the ERP will be refreshed using intra-coding after a certain period. For instance, if the size of the RI region is 1/36 of the ERP frame, the RI will refresh the entire 360° scope every 36 frames. This periodic refreshment makes the system robust to both FoV prediction errors and frame losses due to packet losses. Since the RI region has a low chance to be viewed, it is allocated with a rate lower than PF and PF+ regions. Note that the first frame (entire 360° scope) needs to be intra-coded at a lower rate (high quantization level) to reduce the bits of the initial frame, and hence reduce the initial buffering time.

We use the tile-based coding that provides easy control of coding methods and rates for different regions [29] [30]. The entire ERP frame is divided into non-overlapping small tiles. All tiles covering the PF region are inter-coded at the normalized rate R_e, and all the remaining tiles covering the PF+ region are inter-coded at the normalized rate R_b. Note that the shape of a PF or PF+

region on the EPR depends on its latitude, as shown in Fig. 2.1. Therefore, the number of tiles needed to cover the PF or PF+ region may differ in each frame. On the other hand, the RI is a rectangular region on the ERP consisting of a fixed number of tiles, irrespective of FoV. To simplify the rate allocation, we code the RI and PF+ region using the same normalized rate R_b. Since the RI uses intra-coding and the PF+ uses inter-coding, the quality of RI is lower than the quality of PF+, even though they share the same average allocated rate R_b. Because the RI region rotates on the entire ERP, it may have overlap with the PF or PF+ region in some frames. Those tiles in the RI that fall within the PF or PF+ region are treated as RI and coded using the intra-mode, in order to eliminate the decoding error propagation caused by potential frame losses. The decoder only decodes and updates the tiles in the PF, PF+, and RI regions. In the un-coded regions, content from the latest decoded frames will be replicated. Other more sophisticated error concealment methods can be incorporated in the future to enhance the quality of these un-coded regions.

2.3 The Optimal Tile Size

In tile-based video coding, the tile size affects both the video coding efficiency and transmission flexibility. Larger tile sizes provide higher coding efficiency. However, it also leads to more unused areas outside the FoV and the bits for coding those unused areas are wasted. Early work only explored this trade-off for 1080p and 4K videos [31].

We conduct a similar experiment for the 8K resolution JVET 360° test videos (the detail of the video coding set-up and the test sequences is described in Sec. 3.6). The optimal tile size should minimize the total bit consumption of all tiles needed to cover the FoV averaged over all possible viewports for a fixed

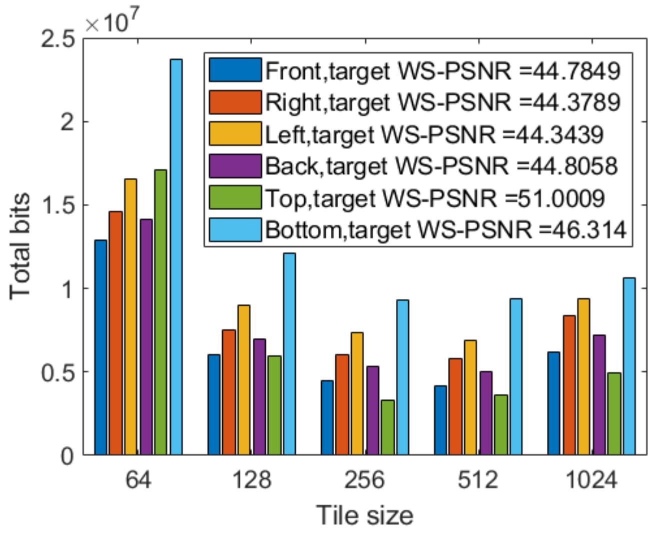

Figure 2.3: Total bit consumption under the same QP inside the FoV for 5 different tile sizes for 8K video. The horizontal axis indicates the number of pixels in each side of the square tile. All the tiles are coded in the inter-coding mode except the first frame. A constant QP=30 was used. The resulting WS-PSNR in the FoV region are reported in the figure legends. Different color bars are results for FoVs in different directions. The results are for sequence “Trolley", similar trends are observed for “Chairlift".

quality. Fig. 2.3 illustrates the total bit rates needed when the tiles are coded in the inter-coding mode at a constant quantization parameter (QP) to cover a $90^\circ \times 90^\circ$ FoV for 5 different tile sizes. Generally, the number of tiles needed differs depending on the viewport direction. We found for the 4 FoVs centered on the equator, they achieve minimal bit rate consumption when using tiles of size 512×512 pixels, slightly better than using a tile size of 256×256 pixels. For the 2 FoVs facing the top or bottom directions, there are relatively more wasted pixels in the boundary tiles. In this case, 256×256 tile size is slightly better than 512×512. Given that a smaller tile size offers more granularity in varying the sizes of RI and PF+, we choose 256×256 pixels as the tile size for the 8K video sequences. Note that all coding experiments and streaming simulations in this work are conducted using this tile size.

2.4 Quality-Rate Models

The proposed system adapts the region sizes and rates for each video segment (1 sec. long in our experiments) based on the expected quality-rate (Q-R) functions in PF, PF+ and RI regions. The main challenge to model those Q-R relations is that the FoV location and consequently the coded regions vary temporally on the ERP. In this subsection, we first introduce the objective quality metric for the 360° video. Next, we model the "ideal" Q-R functions based on the coding experiments on two standard 360° test videos. The "ideal" Q-R function for inter-coding for a given FoV is determined by predicting the current frame from the previous decoded frame, assuming the FoV is fixed over the entire video. Since the FoV and hence the coded regions may change in each frame, some tiles may be predicted from uncoded regions in the previous frame in real applications. The quality of these uncoded regions depends on the time lapse since they were last coded. We introduce the rate-increase factor to account for such time lapse for temporal prediction. Furthermore, we use the quality-decay factor to model the degraded quality of the non-coded tiles for the present frame.

2.4.1 Objective Quality Metric

Weighted-to-spherically-uniform peak-signal-to-noise ratio (WS-PSNR) is an objective quality metric to evaluate the 360° video recommended by JVET [32]. WS-PSNR is defined as:

$$\text{WS-PSNR} = 10 \log_{10} \frac{\text{MAX}_\text{I}{}^2}{\text{WS-MSE}}, \tag{2.1}$$

with

$$\text{WS-MSE} = \frac{1}{\sum_{i,j} w(i,j)} \sum_{i,j} [I(i,j) - K(i,j)]^2 w(i,j), \tag{2.2}$$

where i, j is the coordinate of a pixel on the ERP frame, $K(i,j)$ and $I(i,j)$ are the color intensity of the pixel (i,j) on the raw and the encoded ERP frames, respectively. The weight $w(i,j) = \cos((\frac{i}{m} - \frac{1}{2})\pi)$ is a factor to model the geometric distortion due to ERP projection along the pitch axis. To calculate the quality of pixels inside an actual rendered FoV, we calculate WS-MSE of pixels inside the projected FoV on the ERP and then derive the WS-PSNR of the FoV. This metric is used to measure the rendered quality. The same method is applied to calculate the quality of the PF, PF+, RI, or the remaining regions.

2.4.2 "Ideal" Quality-Rate Models For Different Coded Regions

Because the shapes and consequently the Q-R relations of the PF and PF+ regions depend on the FoV location, we first empirically determine these Q-R relations separately for six different FoV locations: front, left, right, back, top and bottom. To model the "ideal" Q-R functions, we fix the FoV locations through the entire sequences, so that all tiles in the regions are continuously updated in each frame. We select two JVET 360° test sequences to represent different video content: one stable video shot by a fixed camera and another dynamic video captured by a moving camera. We conduct the coding experiments to derive the Q-R functions for these videos. The reference HEVC Test Model (HM) software [33] is used under JVET common test condition (CTC). Each tile in the sampled PF and PF+ regions is encoded independently using low delay P (LDP) configuration with IntraPeriod = -1, meaning only the first frame is I frame and all the following frames are P frames. Each tile from the RI region is encoded using intra-only configuration with IntraPeriod = 1, meaning every frame is coded as I frame. We determine the rate and the corresponding quality (WS-PSNR) under four different quantization parameters (QP): $27, 32, 37, 42$. Then, to generate the Q-R curve for a FoV region, we determine the tiles needed to cover this FoV, and calculate the WS-PSNR for all the pixels inside the FoV and the total bits of all tiles needed to cover the FoV, for each QP. Figures 2.4(a) and 2.4(e) each shows six curves for the six sample FoV orientations. Here we assume the FoV size is $90^\circ \times 90^\circ$. The normalized bit rate (bits/degree^2) is determined by dividing the total bits by $90^\circ \times 90^\circ$. We further determine the average "ideal" Q-R curve for PF, by averaging the Q-R functions for the six FoVs, based on their probabilities. From the statistics of the viewers' FoV behavior [34], more than 90% FoV centers are located in the range of equator$\pm 45^\circ$. Therefore, we assume that the probabilities for watching the front, left, right, back, top, and bottom FoV are $0.2, 0.2, 0.2, 0.2, 0.1, 0.1$, respectively. The average Q-R curves are also shown in Fig. 2.4(a) and 2.4(e).

The PF+ region covers a border outside the PF region, and the number of tiles needed to cover the PF+ region depends on the width of the border (in degree). In our experiments, we set the border width to $10^\circ, 20^\circ, 30^\circ, 40^\circ, 50^\circ$ (10° means that the extended degree in each direction is 5°). For each of the six FoV orientations, we determine the WS-PSNR among pixels falling in the border region, and count the total number of bits used by the tiles needed to cover the PF+ region, for each target PF+ size in degree. The normalized rate is determined by dividing the total rate by the border size in square degree. For example, with FoV size of $90^\circ \times 90^\circ$, and border size of 10°, the border area is approximated by $100^\circ \times 100^\circ - 90^\circ \times 90^\circ$. The Q-R plots of the six FoV orientations and the weighted average Q-R curve for the border width of 10° are shown in Figs. 2.4(b) and 2.4(f). Figures 2.4(c) and 2.4(g) show the average Q-R curves for different border widths. Note that the coding efficiency is higher for a wider border due to the fact that lower percentage of pixels in the coded PF+ tiles are wasted in such a case.

The RI region is coded using the intra-mode. The quality is the average WS-PSNR of all pixels in a RI, while the rate is the total bits of all pixels in the RI normalized to the average spherical area represented by RIs in different locations on the ERP. Fig. 2.4(d) and 2.4(h) show the average Q-R functions. Note that the Q-R function is the same regardless the RI size, because all the tiles in a RI are considered equally useful.

The normalized bit rate in the coding experiment and Fig. 2.4 is defined in terms of bits/degree^2, i.e. the number of bits needed to cover a unit area on the sphere (the shape of FoV projection on ERP depends on the actual location of the FoV). As shown in Fig. 2.4, we find that the weighted average Q-R curves for PF, PF+ and RI regions can all be approximated well by the logarithmic model:

$$Q(R) = a + b \log R. \tag{2.3}$$

We will use $Q_{\text{PF}}(R), Q_{\text{PF+}}(R)$ and $Q_{\text{RI}}(R)$ to denote the "ideal" Q-R functions for the PF, PF+ and RI regions, respectively. Similarly, $a_{\text{PF}}, b_{\text{PF}}$ denote the parameters for $Q_{\text{PF}}(R)$, $a_{\text{PF+}}, b_{\text{PF+}}$ for $Q_{\text{PF+}}(R)$, and $a_{\text{RI}}, b_{\text{RI}}$ for $Q_{\text{RI}}(R)$. Since the values of these parameters depend on the video content, we show the results of two videos with different characteristics.

2.4.3 Rate-Increase Factor

The "ideal" Q-R models described in Sec. 2.4.2 are based on the assumption that PF and PF+ never change their location over time, which is not true in actual 360° video viewing behavior. When the FoV changes between frames, a tile of the PF (or PF+) region in the current frame may not be coded in the previous frame, or even past several frames, as shown in Fig. 2.2. *The coding time lapse* τ is defined as the frame distance from the frame that the tile was last coded to the current frame. Generally in inter-coding, the accuracy of the temporal prediction reduces when τ increases. In other words, more bits are required to encode the frame at the same quality when τ is larger. Therefore, we define *rate-increase factor* $\rho(\tau)$ to be the ratio of the rate needed for a given τ over the rate for $\tau = 1$ to achieve the same quality. Note that generally $\rho(\tau)$ depends on the QP and the video content under the same τ.

In order to model $\rho(\tau)$, we code the testing videos using fixed QPs with different time lapses to measure the additional bits needed to achieve the same video quality. Fig. 2.5 shows the measured $\rho(\tau)$ for $QP = 22, 27, 32, 37$ for two testing videos. As shown in the figure, the rate-increase factor can be well fitted by a reverse exponential decay function:

$$\rho(\tau) = 1 + c\left(1 - e^{-d(\tau-1)}\right). \tag{2.4}$$

The parameters c and d depend on the QP and the content.

To achieve a given quality Q, the rate needed when $\tau = 1$ is given by the "ideal" rate $R(Q)$ determined using the "ideal" Q-R models of the PF or PF+ region. The rate corresponding to other τ is given by

$$\tilde{R}(Q;\tau) = \rho(\tau)R(Q). \tag{2.5}$$

In reality, the time lapse of tiles are temporally and spatially variant due to the FoV dynamics. To adapt the coding rates and the region sizes at the beginning of each video segment, we adjust the Q-R functions derived in Sec. 2.4.2 for the PF and PF+ regions based on the distribution of τ in the previous segment as follows:

$$\tilde{R}(Q) = \left(\sum_{\tau} p(\tau)\rho(\tau) \right) R(Q), \tag{2.6}$$

where $p(\tau)$ is the probability of τ measured among all the tiles in the PF or PF+ region in the previous segment.

2.4.4 Quality-Decay Factor

As shown in Fig. 2.2, the PF, PF+, and RI regions may not cover the entire actual FoV. For a tile in the actual FoV that is not be coded and updated, it will remain the same as when this tile was last coded. The rendered quality depends on how long ago (time lapse τ) when it was last coded, and also the quality of the last coded tile. We define *quality-decay factor* $\kappa(\tau)$ as the ratio of the quality of such a tile after time lapse τ over the quality of the last coded tile.

In our experiments, we estimate the decay factor through simulation. For a given video sequence and a given QP, for each frame, we use the WS-PSNR derived from the "ideal" experiment described in Sec. 2.4.2 as the WS-PSNR with $\tau = 0$, denoted as WS-PSNR(0). For the following τ-th frame, we calculate the WS-PSNR between the coded first frame and the raw τ-th frame, represented by WS-PSNR(τ). The quality-decay factor is defined by $\kappa(\tau) =$ WS-PSNR$(\tau)/$WS-PSNR(0). We repeat this experiment using each of the first 200 frames in each of the two videos as the initial frame, and use the average decay factors for all the 200 samples as the decay factor for the given τ. We repeat this process for τ between 1 and 100 to determine the $\kappa(\tau)$ function. As shown in Fig. 2.6, the quality-decay factor can be well fitted by a modified exponential decay model:

$$\kappa(\tau) = e^{-g\tau^{h}}. \tag{2.7}$$

Note that the values of g and h also depend on the video content and QP. For a given rate, if the quality of the tile when it was last coded is $Q(R)$, the quality of the rendered tile that is τ frames away is determined by

$$\tilde{Q}(R;\tau) = \kappa(\tau)Q(R). \tag{2.8}$$

2.5 Optimizing Rate Allocation and Region Sizes

2.5.1 Expected Video Quality

The perceived quality of a rendered pixel depends on the coding region it falls in. Let α_{PF} denote the probability that a rendered pixel is in the PF region without overlapping with the RI region, to be called the *hit rate* of the PF region. Similarly, $\alpha_{\text{PF+}}$ and α_{RI} denote the probabilities that a rendered pixel is in the PF+ region (without overlapping with RI) and the RI region, respectively. Obviously α_{PF}, $\alpha_{\text{PF+}}$, and α_{RI} depend on the accuracy of FoV prediction. $\alpha_{\text{PF+}}$ and α_{RI} also depend on the sizes of the PF+ and the RI regions.

When a pixel in the actual FoV falls in the PF, PF+, or RI region, with the probability α_{PF}, $\alpha_{\text{PF+}}$, and α_{RI}, respectively, it is decoded with quality $Q_{\text{PF}}(R_e)$, $Q_{\text{PF+}}(R_b)$ and $Q_{\text{RI}}(R_b)$, correspondingly. For pixels not covered by these regions, they repeat the content last decoded. The quality for such a pixel is denoted as $\kappa(\tau)Q_{\text{last}}$, where τ is the time lapse (frame distance) since it was last updated and Q_{last} is quality when last coded, as explained in Sec. 2.4.4. Generally, τ is varying in both space and time, and Q_{last} can be either $Q_{\text{PF}}(R_e)$, $Q_{\text{PF+}}(R_b)$ or $Q_{\text{RI}}(R_b)$. In practice, since the rendered pixels have very small chance to fall in the un-coded region (lower than 1% in our simulations), we can use the worst-case $\kappa_{\min}Q_{\text{RI}}(R_b)$ to conservatively estimate its quality, where $\kappa_{\min} = \kappa(\tau_{\max})$, with $\tau_{\max}$ being the full-ERP intra-refresh time (inversely proportional to the RI size). Hence, the rendering quality of the actual FoV can be written as

$$\begin{aligned} Q_1 =& \alpha_{\mathrm{PF}} Q_{\mathrm{PF}} + \alpha_{\mathrm{PF+}} Q_{\mathrm{PF+}} + \alpha_{\mathrm{RI}} Q_{\mathrm{RI}} \\ & + (1 - \alpha_{\mathrm{PF}} - \alpha_{\mathrm{PF+}} - \alpha_{\mathrm{RI}}) \kappa_{\min} Q_{\mathrm{RI}}. \end{aligned} \tag{2.9}$$

Q_1 is the quality at the receiver when all the coded bits for this frame arrive in time. When the bits for a frame arrive later than its display deadline, the previously decoded frame is simply repeated and we can conservatively estimate the average quality as $Q_2 = \kappa_{\min} Q_{\mathrm{RI}}$. Let γ denote the *frame delivery rate*, which is the probability of in-time delivery. The overall expected quality can be expressed as

$$\begin{aligned} \bar{Q}(R_b, R_e, A_{\mathrm{PF+}}, A_{\mathrm{RI}}) &= \gamma Q_1 + (1-\gamma) Q_2 \\ &= \gamma(\alpha_{\mathrm{PF}} Q_{\mathrm{PF}}(R_e) + \alpha_{\mathrm{PF+}} Q_{\mathrm{PF+}}(R_b) + \alpha_{\mathrm{RI}} Q_{\mathrm{RI}}(R_b)) \\ &+ (1 - \gamma(\alpha_{\mathrm{PF}} + \alpha_{\mathrm{PF+}} + \alpha_{\mathrm{RI}})) \kappa_{\min} Q_{\mathrm{RI}}(R_b), \end{aligned} \tag{2.10}$$

where A_{PF}, $A_{\mathrm{PF+}}$, and A_{RI} are the sizes of the PF, PF+, and RI region (in unit of the square degree), respectively. Note that $\alpha_{\mathrm{PF+}}$ and $Q_{\mathrm{PF+}}(R)$ depend on $A_{\mathrm{PF+}}$, and α_{RI} is determined by A_{RI}. Therefore, Eq. (2.10) is a function of $A_{\mathrm{PF+}}$, A_{RI}, R_e, and R_b, for a given FoV prediction accuracy characterized by α_{PF}, and the frame delivery rate γ.

2.5.2 Optimization Problem Formulation and Solution

Given the target bit budget B_t of a frame, the region sizes A_{PF}, $A_{\mathrm{PF+}}$, and A_{RI} and the corresponding normalized rates R_b and R_e need to satisfy:

$$\lambda_{\mathrm{PF}} A_{\mathrm{PF}} R_e + (\lambda_{\mathrm{PF+}} A_{\mathrm{PF+}} + A_{\mathrm{RI}}) R_b \le B_t, \tag{2.11}$$

where λ_{PF} is the average ratio of tiles in the PF and not covered by the RI region, and $\lambda_{\text{PF+}}$ is the same for the PF+. Both ratios are estimated by dividing the number of the RI tiles by the number of all tiles on the ERP frame.

Since the PF size is fixed, the goal is to find the optimal region size combination ($A_{\text{PF+}}$, A_{RI}) and corresponding rates (R_b and R_e) to maximize the quality shown in Eq. (2.10) subject to the target bit budget constraint in Eq. (2.11). Generally, $\alpha_{\text{PF+}}$ and α_{RI} increase with larger $A_{\text{PF+}}$ and A_{RI}, and $\kappa_{\min}$ also increases with larger A_{RI}. However, the rates R_e and R_b decrease with larger $A_{\text{PF+}}$ and A_{RI} due to the target bit budget constraint.

To simplify the practical system setting, we limit the possible sizes of the PF+ and the RI within a finite candidate set. For each possible combination of the PF+ and RI size, only R_e and R_b in Eq. (2.10) are the free variables. Given that the optimal solution lies when the bit budget is met exactly in Eq. (2.11), we have $R_b = (B_t - \lambda_{\text{PF}} A_{\text{PF}} R_e)/(\lambda_{\text{PF+}} A_{\text{PF+}} + A_{\text{RI}})$. Then, the optimal R_e can be derived by setting $\frac{\partial \bar{Q}}{\partial R_e} = 0$. We apply the log Q-R model introduced in Sec. 2.4.2 and get the analytical solution as:

$$R_e = \frac{X}{X+Y} \frac{B_t}{\lambda_{\text{PF}} A_{\text{PF}}}, R_b = \frac{Y}{X+Y} \frac{B_t}{\lambda_{\text{PF+}} A_{\text{PF+}} + A_{\text{RI}}}, \tag{2.12}$$

where

$$\begin{aligned} X &= \gamma \alpha_{\text{PF}} b_{\text{PF}}, \\ Y &= \gamma \alpha_{\text{PF+}} b_{\text{PF+}} + \gamma \alpha_{\text{RI}} b_{\text{RI}} + \kappa_{\min} b_{\text{RI}} \\ &\quad - \gamma \kappa_{\min} b_{\text{RI}} (\alpha_{\text{PF}} + \alpha_{\text{PF+}} + \alpha_{\text{RI}}). \end{aligned}$$

We enumerate all possible region sizes and the corresponding optimal rate combinations to find the optimal combination maximizing $\bar{Q}$.

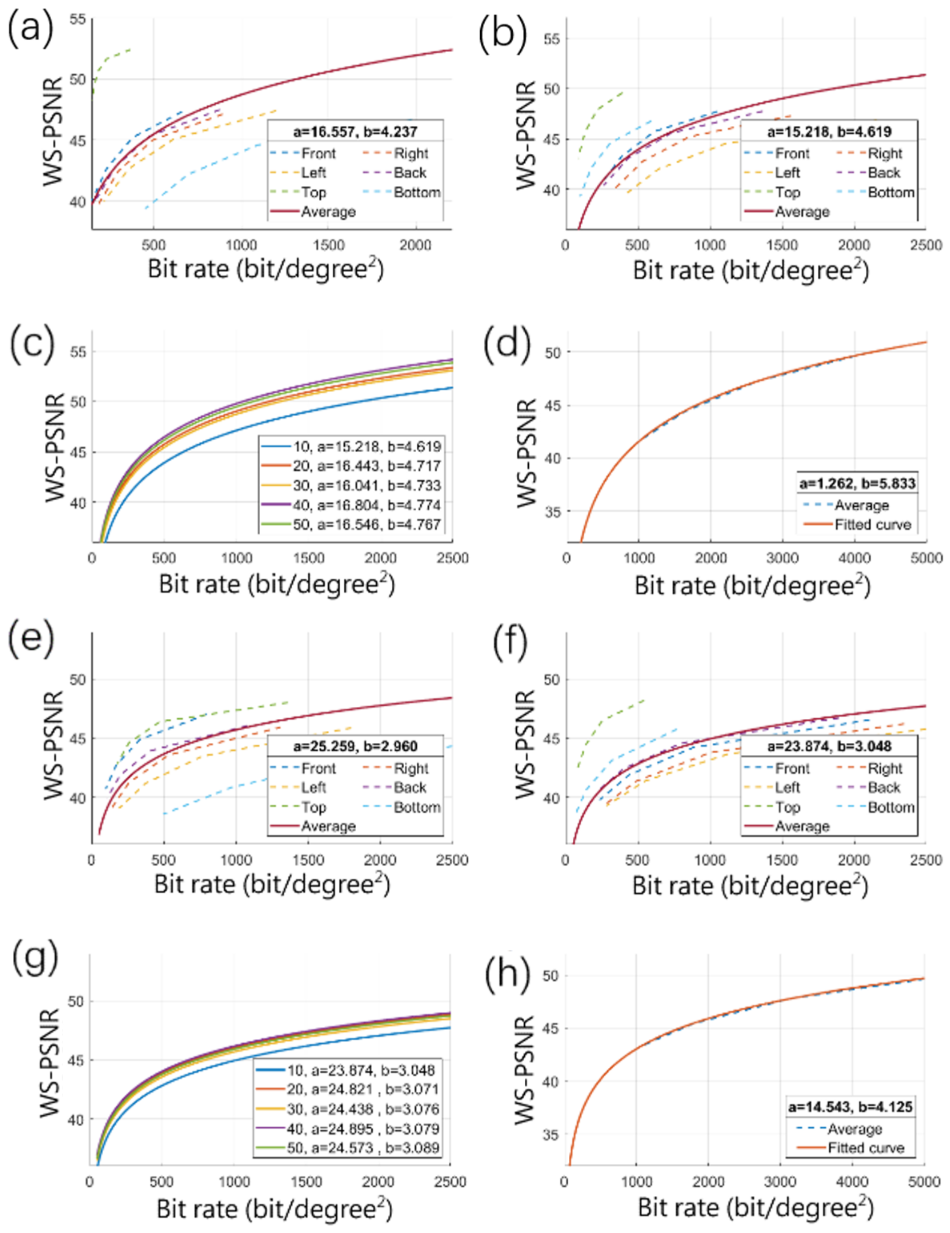

Figure 2.4: Q-R models for "Trolley" (a)-(d) and "Chairlift" (e)-(h). (a)(e): WS-PSNR vs. normalized rate for the PF regions of six viewing orientations, and the averaged WS-PSNR vs. normalized rate. (b)(f): WS-PSNR vs. normalized rate for the PF+ regions when its size is 10°. (c)(g): the averaged WS-PSNR vs. normalized rate for different PF+ region sizes. (d)(h) WS-PSNR vs. normalized rate for the RI region.

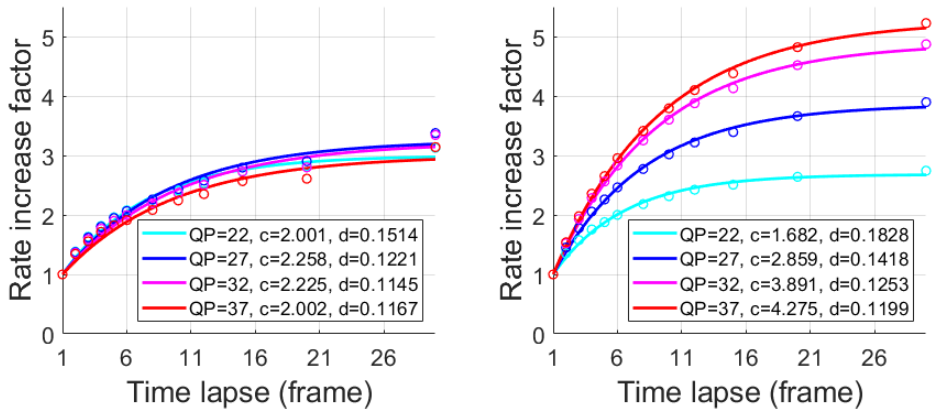

Figure 2.5: The rate-increase factor to maintain the same quality as a function of the time lapse between the inter-coded frame and the reference frame. Left: “Trolley", fixed camera. Right: “Chairlift", moving camera.

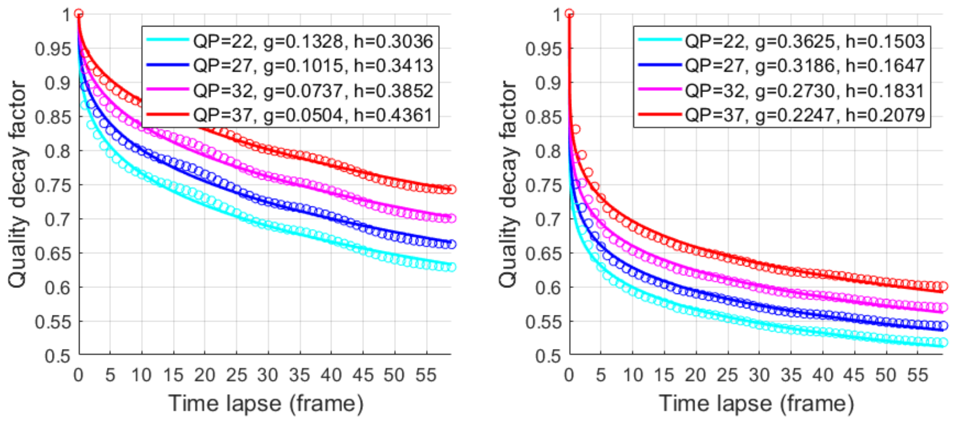

Figure 2.6: The quality-decay factor of pixels due to frame copy as a function of the time lapse between the last-coded frame and the current frame. Left: “Trolley", fixed camera. Right: “Chairlift", moving camera.

Chapter 3

Frame-level FoV-adaptive 360° Video Interactive Streaming with Rate and Region Size Adaptation

3.1 Introduction

Based on our frame-level FoV-adaptive 360° video coding structure with spatial and temporal prediction, we further design a streaming system with region size and rate adaption to maximize the quality and minimize the frame freeze and delay. We design a push-based frame delivery scheme with short sender and receiver buffers to avoid self-congestion. The streaming system adjusts the frame-level bit budget in real-time and controls sender buffer overflow, to maximize the frame delivery rate before the display deadline. The streaming system periodically adapts the sizes and the target normalized bit rates of different coding regions at the segment level, based on the predicted network bandwidth and

FoV prediction accuracy, guided by the developed Q-R models. Accurately predicting the FoV and the bandwidth is critical for FoV-adaptive 360° video interactive streaming. We develop LSTM-based deep learning models for frame-level FoV prediction and segment-level bandwidth prediction, respectively. The FoV and bandwidth prediction modules in the streaming system can be replaced by more powerful prediction algorithms in the future.

In this chapter, Sec. 3.2 presents the push-based FoV-adaptive streaming system. Sec. 3.3 and Sec. 3.4 describe the deep learning models used for bandwidth prediction and FoV prediction. The adaptation of region size and rate allocation at the segment level and the bit budget adjustment at the frame level are then presented in Sec. 3.5. Sec. 3.6 explains the setup of the trace-driven simulations, describes the evaluation metrics, and compares system performance with benchmarks using intra- or inter- coding.

3.2 Proposed Streaming System Overview

The proposed 360° video interactive streaming system uses the "server push" solution, where the server (or Sender as in Fig. 3.3) controls the schedule of video coding and packet sending. The system predicts the network bandwidth ($\tilde{B}_t$) and region hit rates ($\alpha_{\text{PF}}, \alpha_{\text{PF+}}, \alpha_{\text{RI}}$) for each segment at the beginning of encoding the segment (each segment is 1 second long including 30 frames in our experiments), based on the network throughput and the FoV history continuously fed-back by the receiver. Using the estimated bandwidth and region hit rates, the system performs the optimization described in Sec. 2.5 to calculate the sizes and average rates of the RI and PF+ regions for this segment. The video frames in the segment are sequentially coded. The bit stream for each encoded frame is appended to the end of the sender buffer if the buffer is not full, as indicated by Process 1 in Fig. 3.3. If the sender buffer reaches its maximum capacity B_{max}, this newly encoded frame will be dropped to reduce the accumulated delay. We set $B_{\text{max}} = 10$ frames in simulations. The server keeps pushing out as many frames as possible in the sender buffer to fully utilize the available bandwidth, as shown in Process 2 in Fig. 3.3.

Each newly received frame is decoded using the current reference frame in the receiver and appended to the end of the display buffer, as indicated by Process 3 in Fig. 3.3. The reference frame on the receiver will be updated to this newly decoded frame. Even if a frame arrives later than its display deadline, the receiver still decodes it to update the reference frame to avoid any possible mismatch with the encoder.

The display checks the front of the display buffer every 1/3 frame interval. If the next decoded frame exceeds the maximum display delay (20 frames in our experiments), it will be dropped and the display checks the next frame in the display buffer until a frame meets the display deadline. The viewport will be rendered and displayed for each timely, shown as Process 4 in Fig. 3.3. If there is no frame in the display buffer or all frames in the buffer are too late to display, the last displayed frame will be repeated, leading to video freeze. Note that in our trace-driven simulations, we assume encoding each frame takes a constant 33.3ms and decoding a frame takes 11.1ms (33.3ms = 1 frame interval for the 30fps test videos). The reported frame delays in Table 3.1 are already very low (average $<$ 100ms), and can be further shortened when a faster encoder or decoder is available.

3.3 Frame-Level FoV Prediction

The performance of FoV-adaptive 360° video streaming highly depends on the FoV prediction accuracy. Multiple time series prediction methods have been applied on this topic in previous works, e.g. linear regression, weighted linear regression, truncated linear prediction [7,9,11], and deep-learning (DL) methods [35–38]. Although most methods predict the short-term FoV (within the future 1 second) well with the accuracy of more than 90% [11] [37], DL-based methods still outperform conventional methods especially when the prediction horizon is long [35].

We use the popular long short-term memory (LSTM) architecture, which is one of the most suitable neural networks to make predictions based on time series data. The input to the LSTM model consists of the FoV center locations described by Cartesian coordinate (x, y, z) over the past 30 frames. Note that we choose not to use the $(yaw, pitch)$ angles to avoid the issue of 2π periodicity of yaw. The hidden states corresponding to each future frame are mapped to the predicted FoV center locations through two fully connected layers. The predicted location for each new frame is recursively fed to the input for the next frame time, until the desired prediction horizon is reached. We experimented with LSTM models with single, two, and three fully connected LSTM layers. We find the networks with two or three LSTM layers achieve similar prediction accuracy in terms of the FoV hit rate, while they both outperform the single layer model. Therefore, we adopt a model with two LSTM layers and the model structure is shown in Fig. 3.1. The two LSTM layers have 128 and 64 hidden units, and the two fully connected (FC) layers contain 64 and 30 hidden units, respectively. This simple structure provides sufficiently accurate results for the short prediction horizon of interests (typically under 300 ms or 10 frames), while enjoying relatively low computational complexity.

We use the FoV hit rate to evaluate the prediction accuracy, which is defined as the overlapping ratio of the predicted FoV and the ground truth FoV on the unit sphere. We train our model using the FoV trace data from [39]. We choose 20% of the traces as the testing set (including the traces used in the system simulation). Then we split the remaining traces into a training set (80%) and a validation set (20%). We choose the model's hyper-parameters to maximize the FoV hit rate on the validation set. Figure 3.2 shows the FoV hit rate on the test set. Compared to the truncated linear prediction method used in our preliminary study [9], which uses the last few past samples among a maximum number of past samples that can be approximated well by a linear function to predict a future sample, the LSTM model is significantly more accurate.

3.4 Segment-Level Bandwidth Prediction

Bandwidth prediction is critical to the performance of rate-adaptive streaming systems. Many methods have been proposed to predict the network bandwidth in prior works, including Harmonic Mean [40], Recursive least square (RLS) [41], Random Forest [42], and Hidden Markov Model [43]. More recently, deep learning models (including LSTM-based) have shown advantages over prior methods [44–46].

In our streaming system, we predict the average sustainable throughput from the sender to the receiver over the next segment time (1 sec.) at the beginning of the new segment, based on the measured throughput at the intervals of 200ms in the past three segments (3 sec.) returned by the receiver. We use a LSTM sequence-sequence model, with a structure very similar to that for FoV

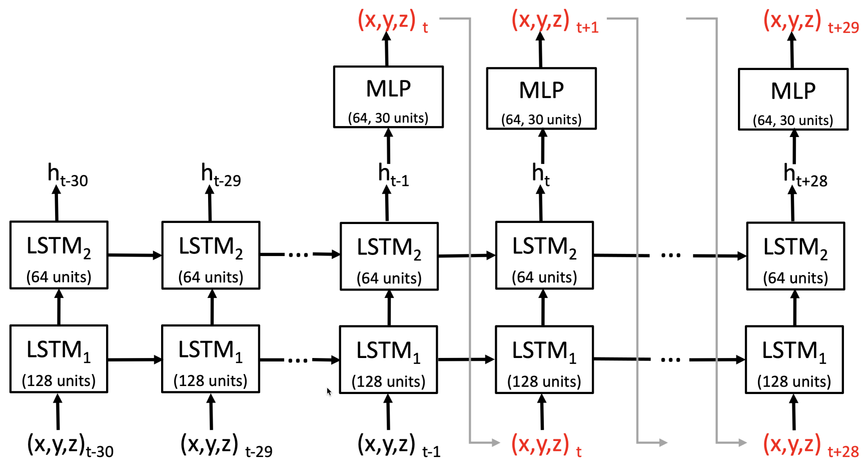

Figure 3.1: The LSTM model for FoV prediction, numbers of hidden units are indicated in layer blocks.

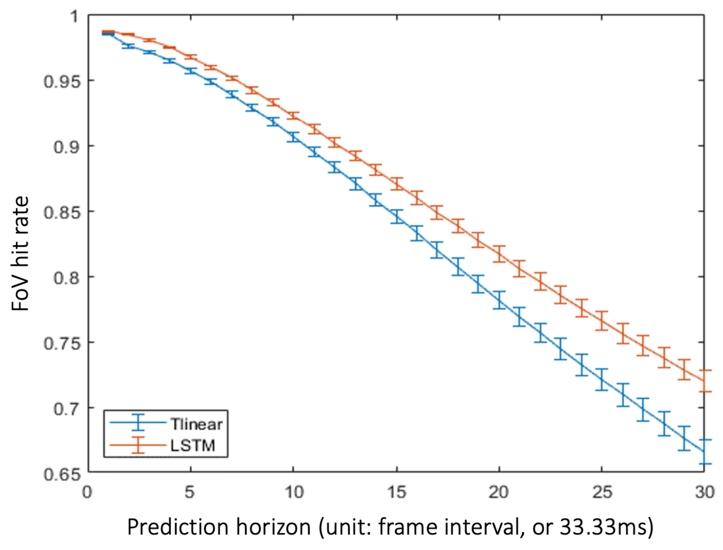

Figure 3.2: Hit rate of predicted FoVs, using the LSTM model and the Truncated Linear predictor, on the testing data. Error bar: 3× standard error of the mean (SEM).

prediction shown in Fig. 3.1, but with different numbers of hidden units in each layer. The bandwidth prediction model has two LSTM layers with 96 and 64 hidden units, followed by two FC layers with 64 and 5 hidden units, respectively. Note that we use the average throughput over a 200ms window as the input feature at each time step, hence, the model has 15 input samples. The model recursively predicts the throughputs for the five consecutive 200ms windows in the next second. The sum of these five predicted throughputs is the predicted total throughput for the next second, $\tilde{B}_t$.

We train our model using the LTE packet traces collected in [41]. We choose a trace named “att-downlink" as the testing trace and it is used in the following simulation experiment. The remaining traces are divided into overlapping 4 sec. long short sequences, and 80% of the short sequences from each trace are used to form the training set, and the remaining 20% are used for validation. We choose the model's hyper-parameters and input window length (among 1 sec., 3 sec. and 5 sec.) based on the prediction errors on the validation set. The window length of 3 sec. was found to perform slightly better than the other choices.

We compare the performance of our model with RLS [41] using Mean Absolute Percentage Error (MAPE) and normalized Mean Absolute Error (nMAE), defined as

$$\text{MAPE} = \frac{1}{T}\sum_t min\left(\frac{|\tilde{B}_t - B_t|}{B_t}, 1.0\right), \tag{3.1}$$

$$\text{nMAE} = \frac{\sum_t |\tilde{B}_t - B_t|}{\sum_t B_t}, \tag{3.2}$$

where B_t and $\tilde{B}_t$ are the actual and the predicted bandwidth at segment t, respectively. MAPE calculates the relative error at each segment and it is more meaningful for our segmentation-level rate-adaptive streaming. We cap the relative error to 1.0 to prevent the large error resulting from when the actual bandwidth is very small to dominate the reported performance.

Compared to RLS, the MAPE of the proposed model is reduced from 21.1% to 18.9% on our testing trace, while the nMAE of the proposed model is also dropped from 14.1% to 13.7%.

3.5 Adaptation of Coding Rates and Region Sizes

We first determine the target bandwidth budget for the next segment based on the predicted available bandwidth for the segment and the current sending buffer occupancy. Then, we calculate the target sizes and bit rates of different regions (PF, PF+, and RI) for all frames in the next segment by maximizing the expected video quality of the segment formulated in Eq. (2.10), using the method described in Sec. 2.5.2. The system measures the average FoV hit rates of different regions and the frame delivery rate of the current segment and assume the FoV hit rate and frame delivery rate remain unchanged when solving the optimization problem for the next segment. The system also calculates the time lapse distribution $p(\tau)$ in the current segment to determine the average rate increase factor using Eq. (2.6) and correspondingly adjust the Q-R functions derived in Section 2.4 for the next segment. Specifically,

1. Calculate the τ distribution of tiles in PF regions in the previous segment.

2. Locate the quality and rate values for each of the 4 QP values on the original average Q-R functions in Figure 2.4.

3. For the rate in each sample point, calculate the rate-increase factor for each τ value using equation (2.4) and hence the adjusted rate. Then determine the average rate among all possible τ's based on the distribution of τ. This will form a new Q-R point, where Q is the same as before, but R increased.

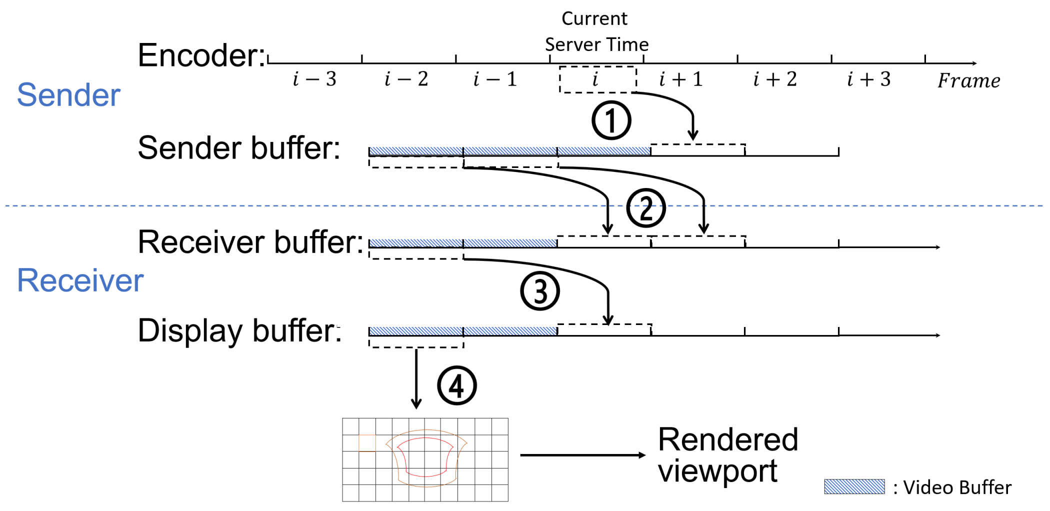

Figure 3.3: The proposed streaming system.

4. Use the new Q-R points to fit a new average Q-R

The adjustment for the Q-R functions of PF+ region with variable region sizes can be done similarly.

3.5.1 Assigning the Total Bit Budget for a Segment Considering Sending Buffer Status

The packet can be occasionally backlogged in the sender buffer over time because of the error of the segment-level bandwidth prediction and the fluctuation of the actual network bandwidth within a segment. To avoid those packets from accumulating in the sending buffer, once we have the predicted bandwidth for next segment s, obtained using the bandwidth prediction method described in Sec. 3.4, we calculate the target bit budget by subtracting the bits q_s currently left in the sender buffer from the predicted bit budget of the segment $\hat{b}_s$. Moreover, the bandwidth utilization ratio η is applied to further lower the probability of exceeding the actual network capacity. Experimental work on real LTE traces shows that this probability can be kept lower than 5% by setting $\eta <= 66\%$ [41].

The target bit budget to encode segment s is

$$b_s = \eta(\hat{b}_s - q_s). \tag{3.3}$$

3.5.2 Frame-level Bit Budget Update

The bandwidth can be unstable within a segment, especially over the LTE or 5G wireless connection. The bandwidth prediction model in Sec. 3.4 only predicts the total bit budget inside a segment, so a more detailed adjustment at the frame level is necessary. The system adapts this frame-level bit budget by checking the space left in the buffer and the remaining bit budget of the segment. Each segment contains N frames ($N = 30$ in our simulation), on average $\frac{n}{N}b_s$ bits should have been used at the time of coding the n-th frame in the segment ($n = 0$ for the first frame). However, the actual bits already used $S(n)$ in this segment when coding the n-th frame could differ from this average. The streaming strategy is designed to be conservative to reduce the risk of freeze and high delay, by setting the remaining bit budget in the segment when coding the n-th frame as

$$b_s(n) = b_s - max\left(S(n), \frac{n}{N}b_s\right). \tag{3.4}$$

The system further adjusts the rate of the n-th frame based on the sender buffer occupation $B(n)$. If the sender buffer is full ($B(n) = B_{\max}$), this frame would not be coded or transmitted. If the buffer is nearly full, the target rate should be reduced from the average rate $\frac{b_s(n)}{N-n}$. The target rate to code the n-th frame when the buffer is not full is determined by

$$B_t(n) = \frac{b_s(n)}{N-n} a \exp(-bB(n)/B_{\max}), \tag{3.5}$$

where a and b are parameters that can be adjusted empirically. We choose $a = 1.20$, $b = 1.00$, and $B_{\max} = 10$ frames in our simulations.

3.6 Trace-Driven Simulation Results

3.6.1 Test Sequences, Bandwidth and FoV Traces

We performed trace-driven simulations to evaluate the proposed coding and streaming system using real viewers' FoV traces and LTE network bandwidth traces. The LTE bandwidth traces are derived from the packet arrival time sequences collected in the real world as described in [41]. To challenge our system, we run the simulation on a dynamic network trace (500 sec. long) with bandwidth variance over mean ratio std/mean $= 0.673$, as shown in Fig. 3.4. This trace includes periods where the bandwidth is high, low, and has sudden drops. To match the rate range for the 8K testing video, we scale up the range of our LTE bandwidth traces to have an upper-bound of 200 Mbps, which is realistic under future 5G networks. We set the one-way propagation time to 15ms in our simulations, which is typical for the network delay within the US [47].

We use two JVET 360° test sequences in 8K ERP format, "Trolley" and "Chairlift" [32], to evaluate the performance of our proposed and other benchmark streaming systems. "Trolley" contains a stable scene where the background is stationary, while "Chairlift" shows a more dynamic scene with a dynamic background. Each sequence has 300 frames in YUV 4:2:0 format with the resolution of 8192×4096 at 30 frames per second. The bit-depths of "Trolley" and "Chairlift" are 8 and 10 bits, respectively.

We assume the category of the video content (e.g., fast-changing scene shot

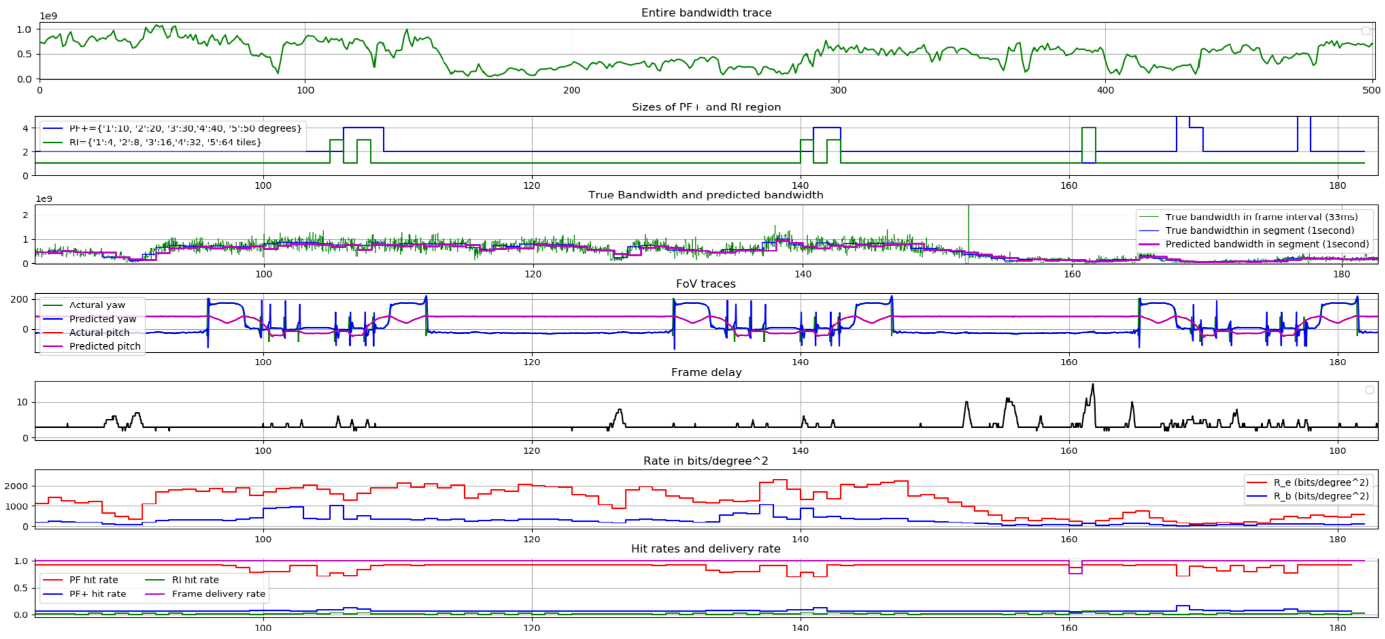

Figure 3.4: The top plot: the LTE bandwidth trace collected in real world (500 second long); Other plots: various performance indices within a short time duration (90 second - 190 second). Horizontal axis is in unit of second.

by a moving camera, or stable scene captured by a fixed camera) can be determined before or at the beginning of a video streaming session, and the parameters for the Q-R models for different categories can be pre-determined. Using these predetermined parameters, the system can perform rate and region size adaptation as introduced in Sec. 2.4 and 2.5. To handle the situation where the video scene category changes dynamically within a streaming session, some automatic ways to periodically updating the scene category need to be developed. Furthermore, we use the Q-R models introduced in Sec. 2.4 and 2.5 to determine the quality of each tile given the rate allocation, instead of doing the actual video coding and decoding. For each frame, the simulation system updates a table recording the time lapse and the quality of each tile when the tile was last coded. As introduced in Sec. 2.4.3, the actual bit rate to inter-code a tile is increased from the target rate by the rate increase factor $\rho(\tau)$ based on the time lapse τ. To calculate the WS-PSNR of each tile in the displayed FoV of each frame, we use the recorded time lapse and the quality when it is last coded to determine its current quality by using the quality decay factor $\kappa(\tau)$ introduced in Sec. 2.4.4.

Since each JVET test sequence only has a duration of 10 seconds, it is not reasonable to collect viewers' FoV trace only for such a short time period (it will be highly affected by the default initial FoV). Therefore, we choose two groups of representative traces from open-source FoV trace datasets [39, 48] for 360° videos. For the stable-scene video "Trolley", we choose the traces collected by [39] where participants watched a video shot by a fixed camera named "Weekly Idol-Dancing". For the dynamic-scene video "Chairlift", we choose the traces collected by [48] where participants watched a video captured by a moving camera named "GoPro VR-Tahiti Surf". To remove the random jitters in the raw collected traces, we apply Kalman filtering to the raw traces and use the smoothed traces in our simulations. Since the bandwidth trace is longer than the FoV trace, we extend each FoV trace by appending the temporally flipped FoV trace to itself repeatedly to match the length of the bandwidth trace. The reported results in Table 3.1 are the average results from simulations using 48 users' FoV traces, each of which is repeated to a duration of 500 seconds.

3.6.2 Streaming System Benchmarks

We also simulated three tile-based state-of-art streaming systems as the benchmarks for comparison, where BM1 and BM2 use intra-coding for all frames, and BM3 uses inter-coding. BM1 follows the coding strategy in [22], which intra-codes and sends non-overlapped vertical slices centered at the predicted FoV center in each frame. The vertical slices cover a $140° \times 180°$ region on the ERP map while the actual FoV is $90^\circ \times 90^\circ$. Note that such vertical slices cannot fully cover the FoV when the FoV was facing the poles (up and down). BM2 uses the same tile size as our proposed system, but it codes all tiles in both PF and PF+ regions only using intra-coding. For BM2, the size of PF+ is fixed to cover a 50° border around the PF. Finally, BM3 applies inter-coding with periodic intra-frames. Rather than using rotating intra regions in the proposed system, BM3 codes the entire ERP of the first frame in each segment as the I frame. The remaining frames in the segment are inter-coded in both PF and PF+ regions with the same rate, and the size of the PF+ region is also fixed to cover a 50° border. We use the Q-R models derived for the RI region (use intra-coding) to determine the WS-PSNR of coded tiles for BM1, BM2, and the I frames in BM3 for a given rate. We apply the Q-R models derived for the PF region (use inter-coding) for the inter-coded regions in BM3. For BM3, the ratio of the I-frame bit rate and P-frame rate is assumed to be equal to the average ratio measured from the actual coding experiment over a range of QP. The normalized rate for the P-frame is set so that the total bits for coding the I-frame and all P-frames in each segment is equal or below the target rate budget. For a fair comparison, all benchmark systems share the same elements with our proposed system, including the same bandwidth and FoV prediction algorithms, the same segment- and frame-level bit rate adjustments, and the same quality-decay model to calculate the quality of the un-coded region in the displayed FoV.

3.6.3 Evaluation Metrics

We evaluate each streaming system on each test video and report the average values of various metrics resulting from using the 48 extended FoV traces. Those metrics include the average frame delay and delay standard deviation (STD), the freeze frequency and duration, and the average rendering quality (average WS-PSNR of all pixels in the actual viewport) of all displayed frames. We also measure the spatial and temporal quality variation, since these quality discontinuities can affect the perceptual quality. The *temporal quality discontinuity* is the mean absolute difference between the rendering qualities of every two adjacent frames. To calculate the spatial variance of each frame, we measure the mean absolute difference between the rendering qualities of each tile and its neighboring tiles in the displayed FoV. Then the *spatial quality discontinuity* is the average of such spatial variance over every displayed frame.

3.6.4 Evaluation Results

The PF region is set to cover an FoV size of $90^\circ \times 90^\circ$. The streaming system adapts the sizes of FP+ and RI regions from the candidate sets. The candidate sizes of PF+ are $\{10^\circ, 20^\circ, 30^\circ, 40^\circ, 50^\circ\}$, while the candidate sizes of RI are $\{4, 8, 16, 32, 64\}$ tiles.

In Fig. 3.4, the first plot is the entire bandwidth trace of 500-second duration we experimented on. The following plots are for a portion of the entire trace to show the details. From the bandwidth and the FoV traces, we can see the LSTM-based segment-level bandwidth prediction and the frame-level FoV prediction is very accurate, especially when the trace does not have random sudden changes. The accurate predictions lead to the high PF hit rate and the high frame delivery rate. From the traces of the region rates and sizes, we can see our proposed streaming system is able to adapt those parameters based on the FoV and bandwidth dynamics. Specifically, we observe that the system tends to choose small RI and PF+ regions when the recent FoV predictions are mostly accurate, while it tends to use larger RI and PF+ region sizes when the recent prediction accuracy drops. When the bandwidth suddenly drops to extremely low, we see the predicted bandwidth needs time to converge to the correct bandwidth. During the transient period, the frame delay increases, the frame delivery rate drops, and a larger RI size is used to increase the refresh frequency of the entire ERP. Note that better prediction algorithms can shorten this response time.

We compare the performance of the proposed and three benchmark systems using the metrics introduced in Sec. 3.6.3. We report the average values over 48 users' traces for "Trolley" and "Chairlift" in Table 3.1. Compared with BM1 and BM2 using intro-coding only, we observe that the WS-PSNR of our proposed system is significantly higher (6-10dB higher), because the proposed system uses the region size/rate-adaptive inter-coding instead of using intra-coding only in the compared systems. However, those adapted rates of PF and PF+ regions bring a slightly higher spatial quality discontinuity (0.1-0.2dB higher) as a compromise. We observe BM2 leads BM1 in terms of the WS-PSNR because BM2's tile structure (same as the proposed) is finer and more flexible than BM1's vertical slice structure. When the PF center is close to the equator, this finer tile structure allows the system to generally encode and transmit a smaller area surrounding the predicted FoV. When the PF center is close to the north or south pole, systems using the tile structure are able to encode all tiles needed to cover the predicted FoV, while the fixed-width vertical slice of BM1 cannot cover the FoV horizontally spanned across the ERP, which also leads to a low FoV hit rate of BM1. BM1, BM2, and the proposed system achieve similar good performance in terms of the average frame delay ($<$ 100ms with low variance), the probability of freeze ($<$ 0.15%), and the duration of freeze ($<$ 1 frame time), because they share the same proposed bandwidth prediction and bit budget allocation algorithms. However, due to the variable bit rates of frames inside each segment brought by the varying coding time lapses, one small compromise of the

Trolley					
Metric	**BM1**	**BM2**	**BM3**	**Prop.**	**Simp.**
WS-PSNR in FoV (dB)	36.17	38.23	44.25	48.41	48.35
Temporal discontinuity (dB)	0.236	0.204	0.298	0.229	0.255
Spatial discontinuity (dB)	0.274	0.002	0.005	0.203	0.397
Average frame delay (ms)	89.04	89.05	119.02	90.95	90.72
Delay STD/Average	0.272	0.272	0.467	0.308	0.311
Percentage of freeze frames (%)	0.093	0.093	0.527	0.116	0.113
Average freeze duration (ms)	11.12	11.12	47.48	20.92	20.74
Display interval average (ms)	33.36	33.36	33.41	33.37	33.37
Display interval STD (ms)	11.93	11.92	19.07	12.46	12.49
Average hit rate, PF (%)	*N/A*	*N/A*	*N/A*	91.48	91.50
Average hit rate, PF+ (%)	*N/A*	*N/A*	*N/A*	7.04	7.55
Average hit rate, RI (%)	*N/A*	*N/A*	*N/A*	0.87	0.88
Average hit rate, total (%)	54.33	99.86	99.90	99.39	99.93
Chairlift					
Metric	**BM1**	**BM2**	**BM3**	**Prop.**	**Simp.**
WS-PSNR in FoV (dB)	37.19	38.69	42.92	45.34	45.29
Temporal discontinuity (dB)	0.177	0.146	0.277	0.159	0.177
Spatial discontinuity (dB)	0.192	0.001	0.006	0.199	0.274
Average frame delay (ms)	89.04	89.04	108.29	94.05	93.73
Delay STD/Average	0.270	0.270	0.392	0.312	0.309
Percentage of freeze frames (%)	0.087	0.086	0.289	0.115	0.112
Average freeze duration (ms)	11.11	11.11	41.47	30.23	24.49
Display interval average (ms)	33.36	33.36	33.39	33.36	33.36
Display interval STD (ms)	11.93	11.92	19.07	12.91	12.86
Average hit rate, PF (%)	*N/A*	*N/A*	*N/A*	91.03	91.08
Average hit rate, PF+ (%)	*N/A*	*N/A*	*N/A*	6.97	7.62
Average hit rate, RI (%)	*N/A*	*N/A*	*N/A*	0.97	0.95
Average hit rate, total (%)	78.35	99.52	99.65	98.97	99.65

Table 3.1: Streaming System Evaluation on *Trolley* (fixed camera) and *Chairlift* (moving camera).

proposed system is the slightly higher delay and freeze. Since the proposed system predicts the time lapse distribution in the new segment based on that in the previous segment, the prediction is not accurate for a segment when the FoV dynamics changes. This would increase the chance that the actual bit rate is higher than the allocated rate, leading to slightly increased frame delay and freezing probability.

Compared with BM3 using inter-coding, the proposed system achieves a significantly lower probability of freeze (60-78% lower) and frame delay (14-28ms lower), because BM3 uses the traditional GOP coding structure which experiences periodic rate spikes when coding the I-frame in each segment. The proposed system also has 2-4dB higher average WS-PSNR than BM3, because the proposed system optimizes the sizes and rates of PF and PF+, and codes fewer tiles using the intra-mode (determined by the adapted RI size and rate).

To evaluate the benefit from adapting the region sizes, we also simulate a “simplified system", which uses fixed PF+ size of 50°, and fixed RI size of 4 tiles, which is noted as “Simp." in Table 3.1. We can see the performance degradation from the proposed to the simplified system is very small. Therefore, this simplified system might be more preferable for practical adoption, especially for low-power mobile devices.

Our preliminary experiments reported in [49] used truncated-linear FoV prediction and RLS bandwidth prediction. Compared to the results in [49], the current system using LSTM-based bandwidth prediction reduces the average frame delay by about 3-5ms and decrease the percentage of freeze frames by 60-70%, and the total freeze duration by 10-15 ms. Meanwhile, the LSTM-based FoV prediction leads to increased FoV hit rate (about 1% higher), which in turn results in better rendered quality (about 0.1dB increase in WS-PSNR). Note that although the LSTM-based FoV prediction provides significant improvement in

FoV hit rate in long-term prediction as shown in Fig. 3.2, the proposed system enjoys short frame delay and typically only needs to predict the FoV of future 3-5 frames, for which the gain from the LSTM-based FoV prediction is limited.

Chapter 4

Octree-based Scalable Point Cloud Geometry Coding with Learned Entropy Model and Resolution Enhancement

4.1 Introduction

We first introduce our novel entropy coding method for the octree-based geometry coding. For each non-empty parent node, to predict the probability that each of its 8 children nodes is occupied , we form an initial "noise" context cube. For example, if we use a context that includes $5 \times 5 \times 5$ parent nodes ($k = 5$), the context cube will include $10 \times 10 \times 10$ children nodes. The nodes that have been coded in the context cube will have context values of either 1 (occupied) or 0 (empty), nodes that have not been coded will have context values of 0 if they correspond to empty parent nodes, and finally the uncoded nodes that correspond to occupied parent nodes will be assigned a value of 0.5. We then apply a 3D convolution-based neural network to denoise the values in the cube and use the output values of the denoised cube at the center $2 \times 2 \times 2$ voxels as the predicted probability. As with G-PCC, we code the non-empty nodes from the top level of the octree to the next level, sequentially, naturally yielding a scalable bit-stream.

On the decoder side, if the received bit-stream only includes a partial representation of the full octree, corresponding to a coarse (lossy) representation of the original point cloud, we further propose a 3D convolution-based neural network to upsample the reconstructed point cloud. We have found that such upsampling at the decoder side can significantly improve the quality of the reconstructed point cloud.

In addition to fully convolutional models, we also develop alternative probability estimation and upsampling models with significantly less complexity while maintaining comparable coding efficiency. We train our proposed and baseline models on a subset of ShapeNet [50], and evaluate the performance on the 8iVSLF dataset recommended by MPEG [51]. Compared with G-PCC [26] (using a handcrafted entropy model) and VoxContext-Net [27] (using a machine learning method), our method saves around 80% bits for achieving the same reconstruction quality during lossy compression (by stopping before the final octree level), or saves around 30% bits when the point cloud is losslessly coded.

A key advantage of our work from the prior related studies is that our octree-based entropy coding model is naturally scalable. The bitstream can be organized into segments, where each segment corresponds to an octree level. Hence, decoding an octree level only requires the information from the earlier part of the bitstream. This scalable coding setting benefits the future design of streaming systems, enabling the streaming system to dynamically change the delivery rate based on the channel conditions. It would also enable the streaming systems to perform intelligent prefetching and correction. For example, the system may prefetch future video segments at a lower rate to prevent freeze, and fetch additional bits to enhance the quality of the prefetched version at a later time when more bandwidth is available.

In this chapter, we first summarize the related works on point cloud compression in Sec. 4.2. Then, we present our proposed methods for point cloud geometry coding in Sec. 4.3, including our notation and basic ideas, the conditional probability estimation model through denoising a “noisy" context cube, and the resolution enhancement model of decoded point clouds. Next, we demonstrate our experimental results in Sec. 4.4, including the comparison with benchmark systems and ablation studies.

4.2 Related works

The point cloud compression (PCC) methods in the literature can be categorized into two classes: video-based (V-PCC) and geometry-based (G-PCC) [52]. Video-based methods usually first generate 3D surface segments by dividing the point cloud into some connected regions, called 3D patches. Then, each 3D patch is projected independently into a 2D plane, and those patches on the 2D plane are organized and coded by traditional video encoders. Video-based methods are heavily investigated and the performance benefits from the well-developed 2D video encoders. MPEG-PCC already released video-based PCC standard (ISO/IEC 23090-5) in 2021 [53]. Meanwhile, geometry-based methods encode the coordinates and colors of the points directly in 3D space. Geometry-based methods are developing fast in recent years and experts are exploring both traditional and deep-learning methods.

For the handcrafted geometry-based methods, tree structures are usually used (e.g., octree [25] or KD-tree [54]) to recursively divide the 3D space. Since the first work that uses the octree to present the 3D geometry [25], more traditional methods [55–59] emerged using octree variants or considering temporal information to further improve the coding efficiency. The MPEG group has been developing a geometry-based PCC standard (G-PCC) [60] using a handcrafted entropy model, and its corresponding test model (TMC13) [26, 61] has been made available and supports the octree coding mode. Another widely-used open-source G-PCC software, Google's Draco, uses a KD-tree compression method [62].

Several prior studies also exploited geometry-based point cloud compression using deep learning models. Some works construct the uniform voxel grids of the entire 3D space to represent the point cloud [63–67]. Those works perform the 3D convolution on the voxel grids and consume an extensive amount of memory space to save the voxel grids, which leads to the inefficiency of processing large or sparse point cloud data. A recent work [68] performs sparse convolution on the voxel grids to reduce the time and space complexity. Another work [69] improves the performance of G-PCC by both spatially and temporally adapting to the optimal deep learning entropy model based on the characteristics of the point cloud. However, switching between entropy models makes this method not scalable. Previous study [70] introduces a set of improvements to the entropy model and training strategy to achieve a lower rate but the codec is also not scalable.

While the MPEG group is working on standardizing the G-PCC, using the octree is recognized as a good approach with several benefits (e.g., saving memory consumption and being scalable). Recently, several studies apply the deep learning model on the octree nodes [27, 71] that shows strong potential for performing 3D convolution while still using octree structure. To predict the probability for the leaf nodes, the earlier work [71] forms the node's context only from the node's ancestors (parent nodes); the later work, VoxContext-Net (VCN) [27] uses the context from the spatial neighbors in the previous coded level (one level above). However, none of them use the strong context information from the currently coded octree level. Additionally, the coordinate refine model proposed in VCN [27] only adjusts the node location, which does not realize the full potential of post-processing at the decoder side.

In this work, we propose to use the context information from the currently coded octree level for the entropy coding model; we further propose to use the upsampling as the post-processing step after lossy decoding. Both approaches significantly improve the coding performance.

4.3 Proposed Methods

4.3.1 Context-Based Entropy Coding for Octree Geometry: Notation and Basic Ideas

To achieve scalability, we adopt the octree representation. A point cloud is represented in a sequence of L occupancy levels: $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_L$. The octree is coded from the first level occupancy $\mathbf{X}_1$ to the last level $\mathbf{X}_L$, sequentially and losslessly. The point cloud can be reconstructed to level l if the coded bit-streams for $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_l$ are given. When the coded bit-streams for all occupancy levels are given, we can losslessly reconstruct the original point cloud.

Each non-empty node $x_{l-1,i}$ at level $l-1$, location i has 8 children at level l, represented in occupancy $\tilde{v}_i^l = \{v_j \in \{0,1\} : j = 1, 2, \cdots, 8\}$. With context-based entropy coding, we code $\tilde{v}_i^l$ into a bit-stream based on the conditional probability mass function $p(\tilde{v}_i^l | \tilde{C}_i^l)$, where $\tilde{C}_i^l$ represents the context. In general $\tilde{C}_i^l$ may include nodes in $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_{l-1}$, as well as the adjacent nodes in $\mathbf{X}_l$ that

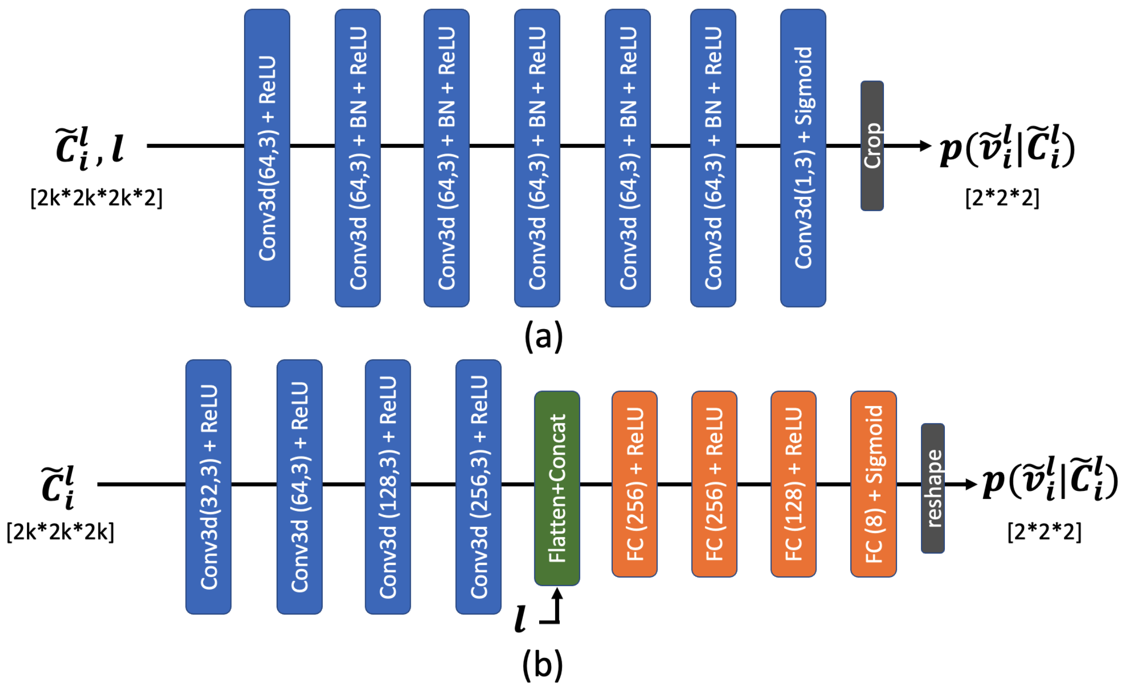

Figure 4.1: Network architectures for entropy coding: (a) Model A: the proposed convolution only architecture, (b) Model B: the convolution + fully connected architecture. “Conv(n,l)" means a 3D convolutional layer outputting n feature channels with a 3D filter of kernel size of $l \times l \times l$.

have been coded. With the same context available at the decoder, the decoding process is to map the bit-stream back to $\tilde{v}_i^l$.

4.3.2 Conditional Probability Estimation through Denoising a "Noisy" Context Cube

The main challenge in context-based entropy coding is how to form the context $\tilde{C}_i^l$) and how to estimate the probability distribution $p(\tilde{v}_i^l|\tilde{C}_i^l)$. To accurately estimate the probability distribution, it is important to design a context that fully utilizes all the information from previously decoded nodes. When coding an octree node at the current level $\mathbf{X}_l$, the context should take into account of both the information at the upper level $\mathbf{X}_{l-1}$ and the current level. However, since not all the information at the current level is known when the current node is being coded, we need to treat the known and the unknown part separately to ensure that the context for probability estimation used during the encoding is available during the decoding process.

In the proposed probability estimation method, we code the nodes $\tilde{v}_i^l$ following a fixed spatial order. We ensure that when a node at the spatial position (x_i, y_i, z_i) is being coded, nodes at the same level with coordinates $c \in \{(x,y,z) : z < z_i\} \cup \{(x,y,z) : y < y_i, z = z_i\} \cup \{(x,y,z) : x < x_i, y = y_i, z = z_i\}$ have already been coded. When coding $\tilde{v}_i^l$, we form a context cube $\tilde{C}_i^l$ of size $2k \times 2k \times 2k$ centered at $\tilde{v}_i^l$, corresponding to a cube of size $k \times k \times k$ at the level $l-1$. The known half of the context cube is filled with the true occupancy, *i.e.* 0 for empty voxels and 1 for occupied ones. For the other half of $\tilde{C}_i^l$ that has not been coded, we fill them with 0 if their parent nodes indicate that these children nodes are empty, and 0.5 if their parent nodes indicate that the unknown voxel can be potentially non-empty. Since this context involves uncertainty in signal values in the uncoded voxels, we call it "noisy" context. We propose to use a convolution neural network to "denoise" this context, so that the output of the network represents $p(\tilde{v}_i^l|\tilde{C}_i^l)$, the predicted probabilities that the center $2 \times 2 \times 2$ voxels are non-empty. These predicted probabilities will then be used by an entropy coder.

To reduce the complexity, we train one neural network for probability estimation at all octree levels. To make use of the level information, along with the original noisy context we add another 3D input channel with the same size of $2k \times 2k \times 2k$, and all elements in this channel are set to the level index l. The resulting 3D tensor provides the information about the neighboring voxels' occupancy (which is noisy for the unknown half) and the level in the octree. Inspired by the architecture of DnCNN for image denoising [72], we design the network architecture shown in Fig. 4.1(a). The network takes the context cube and the octree level channel as input. A sigmoid activation is used at the final layer to generate the probability distribution $\mathbf{p} \in \mathbb{R}^8$, with each element $p_j \in [0, 1], j = 1, 2, \cdots, 8$, where $p_j := Pr\{v_j = 1\}$.

Compared to the network used in VoxelContext-Net [27], this network does not have fully connected layers at the end in order to preserve more spatial information. We also develop another probability estimation network with fully connected layers, similar to the one used in the VoxelContext-Net [27], shown in Fig. 4.1(b). However, the context in [27] only uses the occupancy information in the parent level. We will provide performance comparison of these two network architectures in Sec. 4.4.3.

4.3.3 Resolution Enhancement of Decoded Point Clouds

When the bit-rate is restricted due to the network throughput constraint, the

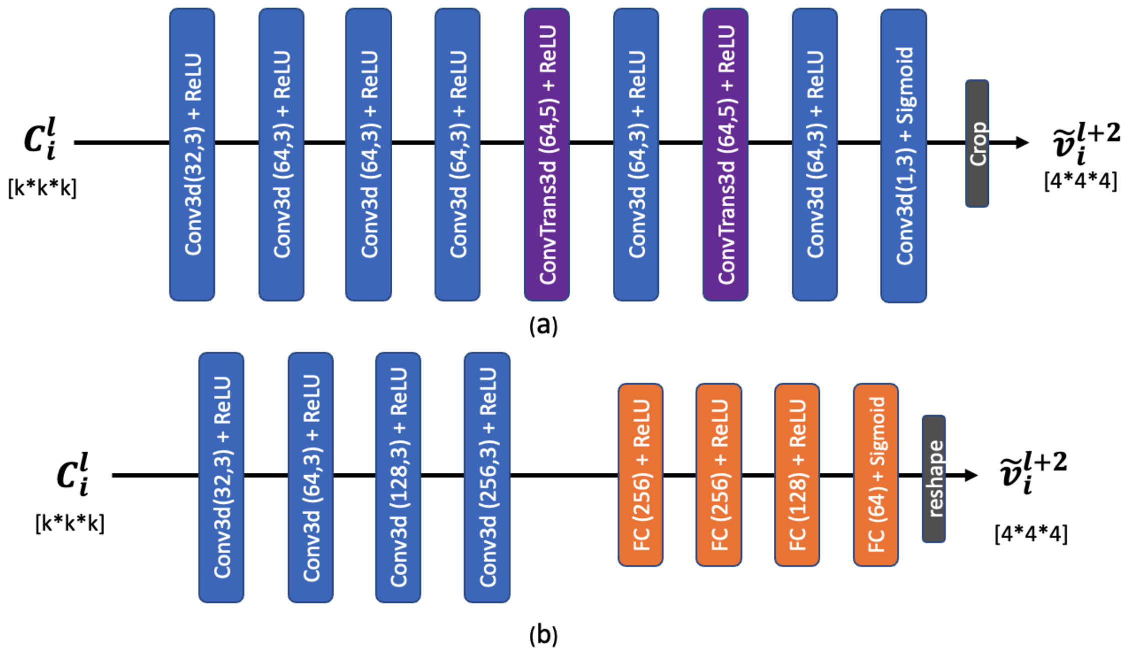

Figure 4.2: Network architectures for upsampling: (a) the proposed convolution-only architecture, (b) the convolution + fully connected architecture.

point cloud cannot be transmitted in full precision. With the proposed scalable coding method, the bitstream may only contain information up to level l of the octree. To further enhance the quality of the reconstructed point cloud, we propose to estimate a finer resolution representation of the point cloud at the decoder side. After losslessly decoding the bitstream to the l-th level of the octree, we upsample each octree node at the l-th level, to a $4 \times 4 \times 4$ voxel grid, to generate a lossy reconstruction of the original octree up to level $l + 2$ without using additional bits.

We upsample a node based on its neighboring context. For a node at location i from the decoded l-th level point cloud, a local voxel context C_i^l centered at this node is formed. The network takes C_i^l as the input and maps it to $\tilde{v}_i^{l+2}$ that is 4 times larger along each dimension than the input. The center $4 \times 4 \times 4$ voxels of the C_i^{l+2} are then binarized and used to form the upsampled point cloud at level $l+2$. This model is repeatedly applied to each node at the l-th level without conditional dependency. Thus, the upsampling of all nodes can run simultaneously on a multi-threaded processing unit (e.g., a GPU) to vastly reduce the processing time.

The network architecture is shown in Fig. 4.2 (a). The network takes an input tensor with size of $k \times k \times k$, and maps it to a tensor with size $4k \times 4k \times 4k$. From the output tensor, the center $4 \times 4 \times 4$ cube is cropped, and binarized with a threshold t to the final predicted occupancy voxel grid.

Since the density and pattern of points at different levels on the octree are diverse, we train the upsampling network separately for every depth level. Note that the model upsamples the octree by two levels only when the decoded point cloud level $l \leq L-2$, where L is the maximal level of the original point cloud. When $l = L-1$, a similar model with only one upsampling layer is used to estimate the decoded point cloud from the l-th level to the full levels. When $l = L$ (lossless coding), no post-processing model is applied.

For comparison, we also develop an alternative architecture with fully connected layers at the end to predict the upsampled points, similar to the one used in the VoxelContext-Net [27], as shown in Fig. 4.2 (b). The performance comparison is provided in Sec. 4.4.3.

4.3.4 Loss Function

We train the probability estimation network to directly minimize the expected bits needed to code the occupancy. This is equal to the binary cross entropy (BCE) loss function over the ground truth occupancy and the predicted probability. The loss for each training sample (corresponding to one non-empty parent node) is

$$\mathcal{L}_e(\mathbf{x}, \mathbf{q}) = -\sum_{j=1}^{8} x_j \log q_j + (1 - x_j) \log(1 - q_j), \tag{4.1}$$

where $x_j \in \{0, 1\}$ denotes the occupancy ground truth and q_j is the estimated probability, $q_j = Pr\{v_j = 1\}$.

We adopt the same loss function for the training of the upsampling network, which upsamples the decoded point cloud from level l to level $l + 2$ (when $l < L - 1$). The loss function is calculated between the ground truth and the predicted probabilities both at the target upsampled level.

4.4 Experiments

4.4.1 Experimental Setup

Datasets

We train the network based on point clouds sampled from the ShapeNetCore [50] dataset. The dataset consists of a total number of 51,300 3D object models in 55 categories, each with mesh and texture. We randomly choose 1024 objects across all categories from ShapeNetCore, and densely sample point clouds on the mesh given by the models. The coordinates of the sampled points are quantized to a bit-depth of 10, with duplicate points removed. We build voxel grids and octrees of depth 10 on the point clouds. The voxel grids and octrees are used to train the proposed networks.

To ensure that our trained networks generalizes to other dense point clouds besides the simple objects in the training set, we evaluate the proposed method on the 8i Voxelized Surface Light Field (8iVSLF) dataset [51], which is recommended by MPEG for dense point cloud coding experiments. We compare

Table 4.1: Bits per point (bpp) used for losslessly coding 8i dense point clouds, by MPEG G-PCC, VoxContextNet(VCN), and our Model A and our Model B. (VCN, Model A and Model B all use $k = 5$)

Models	G-PCC	VCN	Model A	Model B
Longdress	1.02	1.25	0.67	0.72
Loot	0.95	1.23	0.65	0.69
Redandblack	1.08	1.31	0.77	0.82
Soldier	1.01	1.28	0.69	0.74
Average (bpp)	1.02	1.26	0.69	0.74
Rate reduction	0	+23.5%	-32.1%	-27.3%

with two baseline methods, G-PCC and VCN, on the four 10-bit point cloud frames: *longdress_vox10_1300*, *loot_vox10_1200*, *redandblack_vox10_1550*, and *soldier_vox10_0690*.

Evaluation Metric

We measure the quality of the reconstructed point cloud with the point-to-point (D1) PSNR [73,74], calculated using the MPEG PCC DMetrics software [75]. The bit-rate is given in bit-per-point (bpp), calculated by dividing the number of bits over the total point number in the original point cloud.

Baseline Methods

The first baseline method is MPEG standard Geometry Point Cloud Compression (G-PCC) [26]. We follow the common test conditions (CTC) [52] to generate the baseline G-PCC Rate-distortion (R-D) curve. We use the TMC13 [61] and enable the G-PCC octree codec to code the dense point cloud. In order to generate the R-D points for variable bit rates, we set the G-PCC to code the octree level by level and truncate at different levels.

The second baseline method is VoxelContext-Net [27]. In this work, the author did both training and testing on the same ScanNet dataset, which may lead

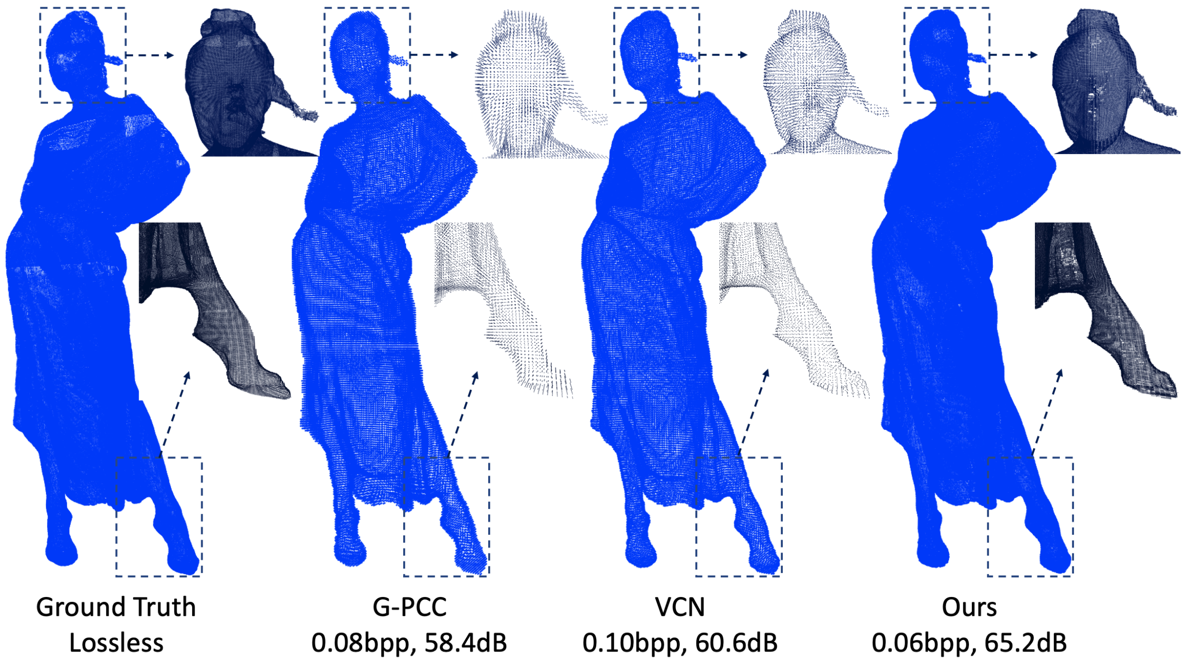

Figure 4.3: Visualization of the ground truth, G-PCC, VCN, and ours.

to model overfitting to the specific dataset and its performance on the MPEG G-PCC standard dataset is unclear. To fairly compare the performance, we train this baseline method on the same subset of the ShapeNetCore dataset and test the it on 8iVSLF dataset. Since there is no publicly available source code of VoxelContext-Net, the training code is reproduced by ourselves.

4.4.2 Experiment Results

For lossless compression, we calculate the bpp for G-PCC, VoxContext-net and the proposed method, on the test point clouds, the results are shown in Table 4.1. For the proposed method, we show the results using both Model A, shown in Fig. 4.1(a), and Model B, shown in Fig. 4.1(b). The context size is $k = 5$. Note that the upsampling model is not used for the lossless compression. As shown, Model A and Model B both outperform the baseline methods. Compared to G-PCC, Model A and Model B reduce the number of bits by 32.1% and 27.0%, respectively, on average. For the VoxContext-net model, we observe that it requires more bits than G-PCC to compress the dense 8iVSLF point cloud. This differs from the performance gain over G-PCC reported in [27]. This could be caused by several reasons: Firstly, we train the model on a subset of the ShapeNet dataset and test on the 8iVSLF dataset for a fair comparison, whereas the original paper [27] reports the testing results on the ShapeNet dataset; Secondly, we used only a subset of the ShapeNet dataset for training whereas the work in [27] used the entire training set of the ShapeNet; Finally, the handcrafted G-PCC explores the previously coded neighbors in the current coding level while VoxContext-net does not, and such information could be especially useful for the dense point cloud data.

The rate-distortion (RD) curves of lossy compression by these methods are shown in Fig. 4.4. Compared with G-PCC and VoxContextNet, both our Model A and Model B save around 80% of bit rate. Model A uses the fully convolutional network in Fig. 4.1(a) for probability prediction, and the fully convolutional network in Fig. 4.2(a) for upsampling. Model B uses the architectures in Fig. 4.1(b) and Fig. 4.2(b), for probability prediction and upsampling, respectively. Visualization results of *longdress_vox10_1300* for our proposed method and baseline methods are shown in Fig. 4.3.

4.4.3 Ablation Study

Network Architecture

For both the probability estimation and point cloud upsampling tasks, the architecture with fully-connected layers (Model B) is slightly less efficient in terms of rate-distortion trade-off than the fully convolutional one (Model A), as shown

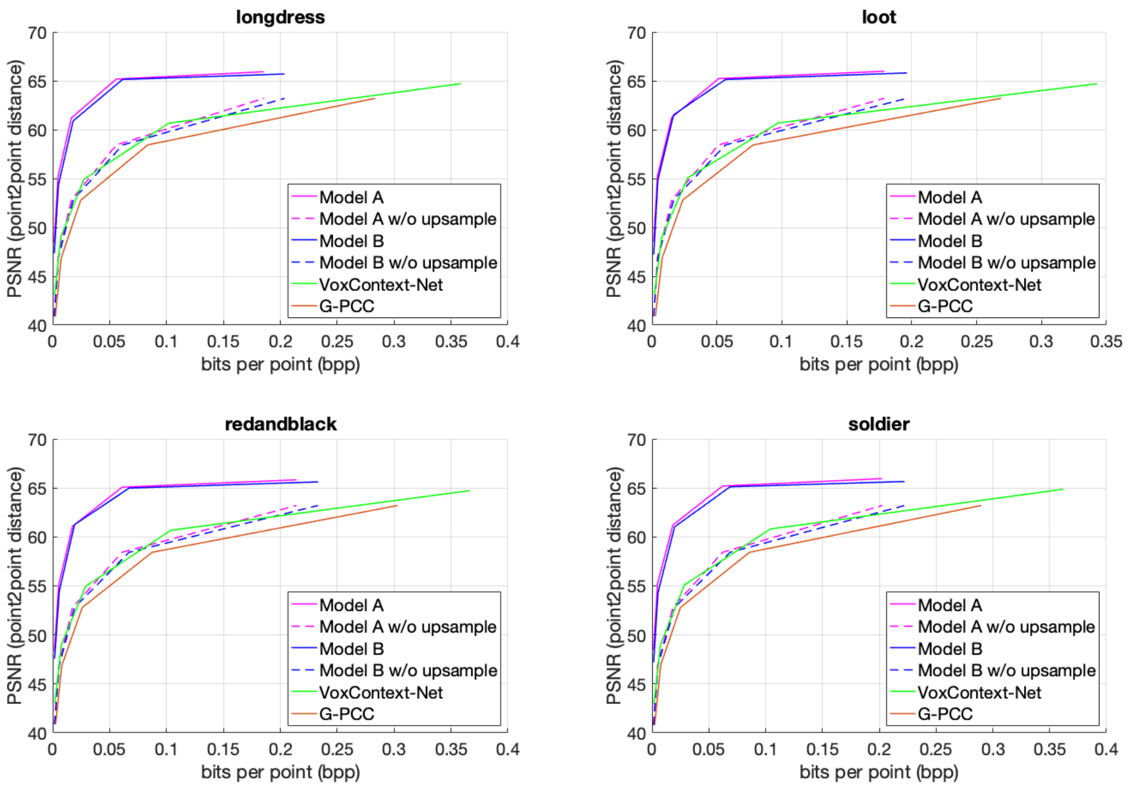

Figure 4.4: Model A (fully convolutional network) vs. Model B (convolutional+fully connected network).

Table 4.2: Number of floating point operations (KFLOPS per node) for the probability estimation (E) and the post-processing (P) procedures, respectively, with different methods and at different voxel context sizes for the Longdress sequence. B(3) means the proposed Model B using context size with $k = 3$.

	VCN(5)	A(5)	B(3)	B(5)	B(9)
E	93.2	559.7	30.1	30.4	206.9
P	149.4	404.9	33.2	60.4	73.0

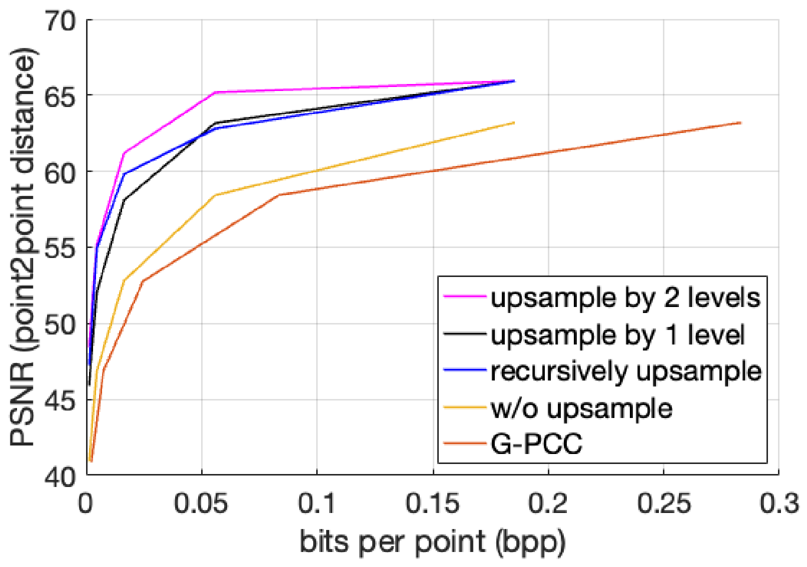

Figure 4.5: Comparison between different postprocessing (upsampling) strategies. For example, for the upsampling by 2 levels curve, the highest rate point is obtained by coding the octree to level 8 and upsample to level 10; while the next highest rate point is obtained by coding the octree to level 7 and upsample to level 9.

in Fig. 4.4. However, Model B significantly reduces the computational complexity. Table 4.2 compares the number of floating point operations (FLOPS) of these two models. As shown, inference using Model B takes only about 15% of the FLOPS compared to Model A. Given that the coding efficiency of using Model B does not drop significantly compared to Model A, Model B may be preferred for practical applications.

Upsampling Strategies

In this ablation study, we compare the proposed scheme with two alternative upsampling strategies: 1) upsample by one level only; 2) recursively upsample by one level until the final level (different models are used at different levels). Fig. 4.6 compares their R-D performances . The experiments are conducted on *longdress_vox10_1300*. Compared with upsampling by one level, the model that directly upsamples by two levels has consistent and significant gain in reconstruction quality. However, recursively upsampling by one level sometimes leads to worser quality. This is because the upsampling error at an earlier level propagates, and negatively affects the upsampling for the following levels. Therefore, we adopt the two-level upsampling strategy.

Context Cube Size

The dimensionality of the context cube affects the probability estimation accuracy and upsampling accuracy. To determine the best size of the context, we train Model B using different context sizes, and compare their R-D performances on the *longdress_vox10_1300* point cloud in Fig. 4.6. The performance gain of using a larger context cube diminishes when the context size k exceeds 5 (corresponding to a context cube of $10 \times 10 \times 10$ for probability estimation, and $5 \times 5 \times 5$ for upsampling). The FLOPS of the models with different context sizes are given in Table 4.2. To balance the performance and the complexity, $k = 5$ is the preferred choice and is used in the results shown in Fig. 4.4, Fig. 4.6 and Table 4.2 . For complexity-sensitive applications, a context size of $k = 3$ could also be used, which only suffer from slight degradation in rate-distortion performance, as shown in Fig. 4.6.

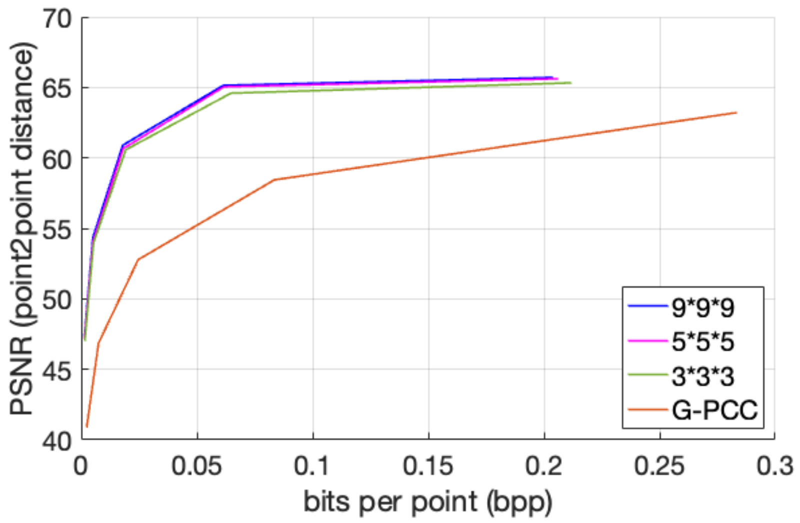

Figure 4.6: R-D performances corresponding to different context sizes. The experiments are conducted with Model B.

Chapter 5

Conclusion

In this thesis, we first explore the coding and streaming system design of 360° video for interactive applications in Chapter 2 and Chapter 3. Then, we explore an octree-based scalable point cloud geometry coding with learned entropy model and resolution enhancement in Chapter 4.

For interactive 360° video streaming, we addressed the challenges of integrating temporal predictive coding into low-latency, FoV-adaptive coding and streaming of interactive 360° video. Through accurate quality-rate modeling that explicitly considers the reduced coding efficiency due to the prolonged temporal prediction time lapse, the proposed system can achieve accurate rate control at both the segment and frame levels and optimize rate allocation to maximize the rendering quality. By introducing rotating intra-regions, the system can periodically stop error propagation due to frame losses as well as quality degradation in un-coded regions, without causing bit rate spikes that increase frame delay. Together with push-based frame delivery and target rate adaptation at both segment and frame levels, the proposed system has been shown to be capable of reducing the mean end-to-end delay to below 100ms under challenging bandwidth traces. Compared to benchmark systems, the proposed system improves the average WS-PSNR by 2 to 10 dB, and also reduces the frame delay by up to 20ms, which should correspond to substantial improvement in the overall user quality of experience.

For point cloud compression, we propose an octree-based point cloud geometry compression method using machine learning models. Our first contribution considers context-based entropy coding of octree nodes. We form a “noisy” context using the occupancy information at the currently coded octree level, and use 3D convolution-based “denoising" networks to predict the probability that an octree node is non-empty. The second contribution considers decoder post-processing and proposes convolutional networks to upsample a low-resolution point cloud (corresponding to a low bit rate resulting from coding to a low level of the octree) to a higher resolution (corresponding to a higher octree level). The combination of the probability estimation and the upsampling approaches significantly improves the rate-distortion performance of octree-based geometry coding over the current MPEG standard G-PCC as well as several prior works leveraging machine learning. We further compare different network structures for both the probability estimation task and the upsampling task in terms of both the rate-distortion performance and computational complexity. Being an octree-based geometry coding solution, our method naturally leads to a scalable bit stream and has strong potential to be adopted in future point cloud streaming platforms. For the future work, this approach can also be extended to the color of the point cloud to develop a completely scalable point cloud compression solution.

In summary, our interactive 360° video streaming system shows significantly improved rendered video quality, while achieving very low end-to-end delay and low frame-freeze probability. Our learning-based entropy model and resolution enhancement model for point cloud geometry coding utilize the advantage of octree's scalability, while significantly improving the efficiency of coding the geometry of dense point cloud. Overall, our coding and streaming system design for interactive 360-degree video applications and scalable point cloud geometry coding will benefit the continued development of lifelike virtual reality experiences.

Bibliography

[1] HUAWEI TECHNOLOGIES CO., "Whitepaper on the vr-oriented bearer network requirement (2016)," .

[2] Jiarun Song, Fuzheng Yang, Wei Zhang, Wenjie Zou, Yuqun Fan, and Peiyun Di, "A fast fov-switching dash system based on tiling mechanism for practical omnidirectional video services," *IEEE Transactions on Multimedia*, vol. 22, no. 9, pp. 2366–2381, 2020.

[3] Jinyu Chen, Xianzhuo Luo, Miao Hu, Di Wu, and Yipeng Zhou, "Sparkle: User-aware viewport prediction in 360-degree video streaming," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[4] Fanyi Duanmu, Eymen Kurdoglu, Yong Liu, and Yao Wang, "View direction and bandwidth adaptive 360 degree video streaming using a two-tier system," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[5] Fanyi Duanmu, Yixiang Mao, Shuai Liu, Sumanth Srinivasan, and Yao Wang, "A subjective study of viewer navigation behaviors when watching 360-degree videos on computers," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[6] Pantelis Maniotis, Eirina Bourtsoulatze, and Nikolaos Thomos, "Tile-based joint caching and delivery of 360° videos in heterogeneous networks," *IEEE Transactions on Multimedia*, vol. 22, no. 9, pp. 2382–2395, 2020.

[7] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *2017 IEEE international conference on communications (ICC)*. IEEE, 2017, pp. 1–7.

[8] Xavier Corbillon, Alisa Devlic, Gwendal Simon, and Jacob Chakareski, "Optimal set of 360-degree videos for viewport-adaptive streaming," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 943–951.

[9] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai, "A two-tier system for on-demand streaming of 360 degree video over dynamic networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 43–57, 2019.

[10] Duc V Nguyen, Huyen TT Tran, Anh T Pham, and Truong Cong Thang, "An optimal tile-based approach for viewport-adaptive 360-degree video streaming," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 29–42, 2019.

[11] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, 2016, pp. 1–6.

[12] Xueshi Hou, Sujit Dey, Jianzhong Zhang, and Madhukar Budagavi, "Predictive adaptive streaming to enable mobile 360-degree and vr experiences," *IEEE Transactions on Multimedia*, vol. 23, pp. 716–731, 2020.

[13] Fanyi Duanmu, Eymen Kurdoglu, S. Amir Hosseini, Yong Liu, and Yao Wang, "Prioritized buffer control in two-tier 360 video streaming," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, New York, NY, USA, 2017, VR/AR Network '17, pp. 13–18, ACM.

[14] Yuwen He, Xiaoyu Xiu, Philippe Hanhart, Yan Ye, Fanyi Duanmu, and Yao Wang, "Content-adaptive 360-degree video coding using hybrid cubemap projection," in *2018 Picture Coding Symposium (PCS)*, 2018, pp. 313–317.

[15] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai, "A two-tier system for on-demand streaming of 360 degree video over dynamic networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 43–57, 2019.

[16] Omar Eltobgy, Omar Arafa, and Mohamed Hefeeda, "Mobile streaming of live 360-degree videos," *IEEE Transactions on Multimedia*, vol. 22, no. 12, pp. 3139–3152, 2020.

[17] Ridvan Aksu, Jacob Chakareski, and Viswanathan Swaminathan, "Viewport-driven rate-distortion optimized scalable live 360° video network multicast," in *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 2018, pp. 1–6.

[18] Carsten Griwodz, Mattis Jeppsson, Håvard Espeland, Tomas Kupka, Ragnar Langseth, Andreas Petlund, Peng Qiaoqiao, Chuansong Xue, Konstantin Pogorelov, Micheal Riegler, et al., "Efficient live and on-demand tiled hevc 360 vr video streaming," in *2018 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2018, pp. 81–88.

[19] Xing Liu, Bo Han, Feng Qian, and Matteo Varvello, "Lime: understanding commercial 360° live video streaming services," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 154–164.

[20] Liyang Sun, Yixiang Mao, Tongyu Zong, Yong Liu, and Yao Wang, "Flocking-based live streaming of 360-degree video," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 26–37.

[21] Cong Zhang, Qiyun He, Jiangchuan Liu, and Zhi Wang, "Exploring viewer gazing patterns for touch-based mobile gamecasting," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2333–2344, 2017.

[22] Marko Viitanen, Jarno Vanne, Timo D Hämäläinen, and Ari Kulmala, "Low latency edge rendering scheme for interactive 360 degree virtual reality gaming," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1557–1560.

[23] Liyang Zhang, Syed Obaid Amin, and Cedric Westphal, "Vr video conferencing over named data networks," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, 2017, pp. 7–12.

[24] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, 2015, pp. 151–165.

[25] Donald Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[26] "MPEG Point Cloud Compression," `https://mpeg-pcc.org/index.php/public-contributions/g-pcc-codec-description`.

[27] Zizheng Que, Guo Lu, and Dong Xu, "Voxelcontext-net: An octree based framework for point cloud compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6042–6051.

[28] Mikko Pitkänen, Marko Viitanen, Alexandre Mercat, and Jarno Vanne, "Remote vr gaming on mobile devices," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2191–2193.

[29] Vamsidhar Reddy Gaddam, Michael Riegler, Ragnhild Eg, Carsten Griwodz, and Pål Halvorsen, "Tiling in interactive panoramic video: Approaches and evaluation," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1819–1831, 2016.

[30] Kaixuan Long, Ying Cui, Chencheng Ye, and Zhi Liu, "Optimal wireless streaming of multi-quality 360 vr video by exploiting natural, relative smoothness-enabled and transcoding-enabled multicast opportunities," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[31] Mengbai Xiao, Chao Zhou, Yao Liu, and Songqing Chen, "Optile: Toward optimal tiling in 360-degree video streaming," in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017, pp. 708–716.

[32] Jill Boyce, Elena Alshina, Adeel Abbas, and Yan Ye, "Jvet common test conditions and evaluation procedures for 360 video," *Joint Video Exploration Team of ITU-T SG*, vol. 16, 2017.

[33] IK Kim, K McCann, K Sugimoto, B Bross, WJ Han, and G Sullivan, "High efficiency video coding (hevc) test model 14 (hm 14) encoder description. document: Jctvc-p1002," *JCT-VC, Jan*, 2014.

[34] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang, "A dataset for exploring user behaviors in vr spherical video streaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 2017, pp. 193–198.

[35] Chenge Li, Weixi Zhang, Yong Liu, and Yao Wang, "Very long term field of view prediction for 360-degree video streaming," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 297–302.

[36] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1161–1170.

[37] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *Proceedings of the 27th Workshop on Netwbibliographyork and Operating Systems Support for Digital Audio and Video*. ACM, 2017, pp. 67–72.

[38] Ching-Ling Fan, Shou-Cheng Yen, Chun-Ying Huang, and Cheng-Hsin Hsu, "Optimizing fixation prediction using recurrent neural networks for 360° video streaming in head-mounted virtual reality," *IEEE Transactions on Multimedia*, vol. 22, no. 3, pp. 744–759, 2020.

[39] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao, "Gaze prediction in dynamic 360 immersive

videos," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5333–5342.

[40] Junchen Jiang, Vyas Sekar, and Hui Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 97–108.

[41] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu, "Real-time bandwidth prediction and rate adaptation for video calls over cellular networks," in *Proceedings of the 7th International Conference on Multimedia Systems*, 2016, pp. 1–11.

[42] Chaoqun Yue, Ruofan Jin, Kyoungwon Suh, Yanyuan Qin, Bing Wang, and Wei Wei, "Linkforecast: cellular link bandwidth prediction in lte networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1582–1594, 2017.

[43] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 272–285.

[44] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifan Han, Feng Li, and Jin Li, "Realtime mobile bandwidth prediction using lstm neural network and bayesian fusion," *Computer Networks*, vol. 182, pp. 107515, 2020.

[45] Jinsung Lee, Sungyong Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Hyoyoung Lim, Jihoon Lee, Xiaoqing Zhu, Sangeeta Ramakrishnan, Dirk Grunwald, Kyunghan Lee, et al., "Perceive: deep learning-based

cellular uplink prediction using real-time scheduling patterns," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, 2020, pp. 377–390.

[46] Abdelhak Bentaleb, Ali C. Begen, Saad Harous, and Roger Zimmermann, "Data-driven bandwidth prediction models and automated model selection for low latency," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[47] Peter L Dordal, "An Introduction to Computer Networks," `http://intronetworks.cs.luc.edu/1/html/packets.html`.

[48] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang, "A dataset for exploring user behaviors in vr spherical video streaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 193–198.

[49] Yixiang Mao, Liyang Sun, Yong Liu, and Yao Wang, "Low-latency fov-adaptive coding and streaming for interactive 360° video streaming," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 3696–3704.

[50] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al., "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[51] Maja Krivokuca, Philip A Chou, and Patrick Savill, "8i voxelized surface light field (8ivslf) dataset," *ISO/IEC JTC1/SC29/WG11 MPEG, input document m42914*, 2018.

[52] D Graziosi, O Nakagami, S Kuma, A Zaghetto, T Suzuki, and A Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020.

[53] "Information technology–coded representation of immersive media – part 5: Visual volumetric video-based coding (v3c) and video-based point cloud compression (v-pcc)," *ISO/IEC*, pp. 23090–5, 2021.

[54] Olivier Devillers and P-M Gandoin, "Geometric compression for interactive transmission," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*. IEEE, 2000, pp. 319–326.

[55] Jingliang Peng and CC Jay Kuo, "Octree-based progressive geometry encoder," in *Internet Multimedia Management Systems IV*. SPIE, 2003, vol. 5242, pp. 301–311.

[56] Diogo C Garcia and Ricardo L de Queiroz, "Intra-frame context-based octree coding for point-cloud geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1807–1811.

[57] Yan Huang, Jingliang Peng, C-C Jay Kuo, and M Gopi, "A generic scheme for progressive point cloud coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440–453, 2008.

[58] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 778–785.

[59] Ruwen Schnabel and Reinhard Klein, "Octree-based point-cloud compression.," in *PBG@ SIGGRAPH*, 2006, pp. 111–120.

[60] "Information technology — coded representation of immersive media — part 9: Geometry-based point cloud compression," *ISO/IEC*, pp. 23090–9, Under development.

[61] "MPEG G-PCC TMC13," `https://github.com/MPEGGroup/mpeg-pcc-tmc13`.

[62] "Google Draco," `https://github.com/google/draco`.

[63] Daniel Maturana and Sebastian Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[64] Tianxin Huang and Yong Liu, "3d point cloud geometry compression on deep learning," in *Proceedings of the 27th ACM international conference on multimedia*, 2019, pp. 890–898.

[65] Bin Yang, Wenjie Luo, and Raquel Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[66] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *2019 IEEE international conference on image processing (ICIP)*. IEEE, 2019, pp. 4320–4324.

[67] Yin Zhou and Oncel Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[68] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma, "Multiscale point cloud geometry compression," in *2021 Data Compression Conference (DCC)*. IEEE, 2021, pp. 73–82.

[69] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 415–430, 2020.

[70] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux, "Improved deep point cloud geometry compression," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–6.

[71] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun, "Octsqueeze: Octree-structured entropy model for lidar compression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1313–1323.

[72] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[73] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3460–3464.

[74] Rufael Mekuria, Sebastien Laserre, and Christian Tulvan, "Performance assessment of point cloud compression," in *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2017, pp. 1–4.

[75] "MPEG PCC DMetrics," `http://mpegx.int-evry.fr/software/MPEG/PCC/mpeg-pcc-dmetric.git`.

Publication List

- **Yixiang Mao**, Liyang Sun, Yong Liu, and Yao Wang, *"Low-latency fov-adaptive coding and streaming for interactive 360° video streaming".* In Proceedings of the 28th ACM International Conference on Multimedia, pages 3696–3704, 2020

- **Yixiang Mao**, Yueyu Hu, and Yao Wang.*"Learning to predict on octree for scalable point cloud geometry coding".* Submitted to 2022 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR).

- **Yixiang Mao**, Liyang Sun, Yong Liu, and Yao Wang, *"Interactive 360$\circ$ Video Streaming with Frame-Level FoV-Adaptive Coding Using Temporal Prediction".* Submitted to ACM Transactions on Multimedia Computing, Communications, and Applications.

- Fanyi Duanmu, **Yixiang Mao**, Shuai Liu, Sumanth Srinivasan, and Yao Wang, *"A subjective study of viewer navigation behaviors when watching 360-degree videos on computers".* In 2018 IEEE International Conference on Multimedia and Expo (ICME), pages 1–6. IEEE, 2018

- Liyang Sun, **Yixiang Mao**, Tongyu Zong, Yong Liu, and Yao Wang, *"Flocking-based live streaming of 360-degree video"*. In Proceedings of the 11th ACM Multimedia Systems Conference, pages 26–37, 2020.
- Liyang Sun, **Yixiang Mao**, Tongyu Zong, Yong Liu, and Yao Wang, *"Live 360 degree video delivery based on user collaboration in a streaming flock"*. IEEE Transactions on Multimedia, 2022
- Angelica M Guercio, **Yixiang Mao**, Victor ND Carvalho, Jiazhen Zhang, Changyuan Li, Zheng Ren, Winnie Zhao, Yao Wang, and Eric D Brenner, *"Plant tracer: A program to track and quantify plant movement from cell-phone captured time-lapse movies"*. Bioscene: Journal of College Biology Teaching, 45(3):14–21, 2019