國立清華大學電機資訊學院資訊工程研究所
博士論文
Department of Computer Science
College of Electrical Engineering and Computer Science
National Tsing Hua University
PhD Thesis

最佳化沈浸式影片串流至頭戴式顯示器
Optimizing Immersive Video Streaming to Head-Mounted Virtual Reality

樊慶玲
Ching-Ling Fan

學號：103062555
Student ID:103062555

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 109 年 9 月
September, 2020

國立清華大學
資訊工程研究所

博士論文

最佳化沈浸式影片串流至頭戴式顯示器

樊慶玲 撰

109
9

Acknowledgments

I would like to thank all the people who helped me in my PhD career. Among them, the most important one is my advisor, Dr. Cheng-Hsin Hsu. He always gives me guidance and suggestions that inspired and helped me on my research. He also has offered many opportunities for us to work with international researchers. Besides, he trained us to have outstanding communication, technical writing, and presentation skills. Moreover, he encouraged and helped us to apply domestic and foreign scholarships. Under his guidance, I step out of my comfortable zone and improve myself in various aspects. Besides, I also thank professor Chun-Ying Huang from National Chiao Tung University and Dr. Kuan-Ta Chen from Academia Sinica. Thanks for their suggestions and contributions to have the great joint projects and papers. Next, I would like to thank the friends I met in Networking and Multimedia Systems Lab. For example, thanks Dr. Hua-Jun Hong for always sharing his life experience with me, and giving me practical suggestions. I also have many great collaborations with Pin-Chun Wang, Wen-Chih Lo, Shun-Huai Yao, and Shou-Cheng Yen. Thanks all my colleagues in the past and the present. In the lab, we always encourage each other and enjoy the achievement together. It is my pleasure to meet and work with them all. Finally, I would like to thank my boyfriend, best friends, roommates, and my family. They always support and help me so that I can keep on and finally complete my work perfectly.

致謝

我想要感謝在碩士、逕讀博士班期間幫助過我的所有人。首先，最需感謝我的指導教授徐正炘教授，總是時時引導、啓發我們對研究的想法，嚴格培養、訓練我們嚴謹的研究態度，積極給予我們多元的國際學術交流機會，用心磨練我們的溝通、寫作與報告技巧，並且鼓勵、協助我們申請國內外獎學金。徐老師的引領與敦促，也讓我多次嘗試跳出舒適圈，進而突破自我的設限，收穫意想不到的成長。除此之外，我也想感謝所有合作過的教授與學者們，如來自交通大學的黃俊穎教授與當時臺灣中央研究院的陳昇瑋博士，有各合作者的建議、交流及貢獻才能完成許多有助於科學發展的專案與論文。接著，我想感謝在實驗室各個時期遇到的朋友們，尤其感謝走在前頭給予我許多人生經驗與建議的洪華駿、曾經一同熬夜合作投稿的王品淳、羅文志、姚舜懷和嚴守成，實驗室所有過去與現在的同仁們。我們互相幫忙、支持並勉勵彼此，在實驗室裡一起努力而成長，也一起享受辛苦換來的成果。與你們相遇、合作與共事，是不可多得的緣分。最後，我要感謝我的男朋友、摯友們、室友們以及最親愛的家人們，在我最煎熬的時刻總是不吝給予關心與幫助，有你們的支持與體諒，我才能堅持下去，順利完成博士學位。

中文摘要

隨著科技日新月異，人們不再滿足於僅僅使用平面顯示器觀看高清(Full High Definition)、超清(Ultra-High Definition)串流影片，而開始追求沈浸式(immersive)的觀看體驗。因此，能提供使用者沈浸式體驗的360度影片蔚為潮流，例如，知名影音串流平台如YouTube及Facebook皆已支援360度影片串流。此外使用頭戴式顯示器(HMD)觀看360度影片，更能讓使用者得到身歷其境的體驗，因為使用者能透過轉頭自然地改變觀看角度，猶如親身處在影片的虛擬環境中。然而，串流360度影片至頭戴式顯示器並非易事。首先360度影片為提供使用者頭戴式顯示器中擬真的畫面，需要極高的解析度而造成相當可觀得檔案大小，這將使頻寬不堪負荷而造成額外的延遲及差强人意的使用者體驗。此外，由於360度影片需投影到二維影片後才能進行壓縮，所造成的變形使現存的影片品質指標，如峰值信噪比(Peak Signal-to-Noise Ratio, PSNR)及結構相似性(Structural SIMilarity Index, SSIM)皆難以準確衡量360度影片的觀看品質，更遑論考慮人類複雜的視覺系統及使用者多元的觀看行為。這些困難阻礙了以使用者體驗為導向的360度影片串流最佳化發展。為了解決上述的挑戰，本論文解決了360度影片串流至頭戴式顯示器的三個核心問題，這三個問題分別處於串流的三個階段：串流傳輸、壓縮與包裝，以及顯示與觀看。首先，我們設計並開發了一個神經網路，運用感測資料及影片分析進行訓練，以預測使用者未來視野。我們所提出的預測網路有效地減少360度影片傳輸所需頻寬，但仍維持相當好的影片品質。接下來，我們利用影片模型、觀看機率及客戶端頻寬分佈，來計算最佳化編碼階梯(Encoding Ladder)，以決定應儲存哪些影片版本在有儲存空間限制的伺服器上，藉此最佳化異質客戶端的觀看品質。最後，我們設計並進行使用者研究與分析，調查並量化各式影響使用者體驗的因子，最終參考這些因子來建立觀看360度影片的使用者體驗模型。我們所研究的這三個核心問題可以有效地最佳化360度影片串流系統，這些開發的技術以及經驗，也將成為未來如虛擬實境、擴增實境、混合實境、以及延伸實境，這些創新應用的基石。

Abstract

Immersive videos, a.k.a. 360° videos, have become increasingly more popular. 360° deliver more immersive viewing experience to end users because of the freedom of changing viewports. Streaming immersive videos to Head-Mounted Displays (HMDs) offer even more immersive experience by allowing users to arbitrary rotate their heads to change the viewports as if they are physically in virtual worlds. However, streaming high-quality 360° videos to HMDs is quite challenging. First, 360° videos contain much more information than conventional videos, and thus are much larger in resolutions and size. This may introduce additional delay and degraded user experience due to insufficient network bandwidth. Second, existing quality metrics are less applicable to 360° videos, which is due to the complex human visual systems and diverse viewing behaviors. This inhibits the development of QoE-orientated optimization for 360° videos. To address these challenges, we study three core problems to optimize the: (i) delivery, (ii) production, and (iii) consumption of immersive video content in the emerging streaming systems to HMDs. First, we design a neural network that leverages sensor and content features to predict the future viewports of HMD viewers watching immersive tiled videos. Our proposed prediction network effectively reduces the bandwidth consumption while offering comparable video quality. Second, we develop a divide-and-conquer approach to optimize the encoding ladder of immersive tiled videos considering the video models, viewing probabilities, and client distribution. Our proposed algorithm aims to maximize the overall viewing quality of clients under the limits of server storage and heterogeneous client bandwidths. Last, we design and conduct a user study to investigate and quantify the impacts of various QoE factors. We then use these factors to build QoE models for the immersive videos. The outcomes of these three studies result in better optimized immersive video streaming systems to HMDs. Our developed technologies and accumulated experience will be the cornerstone of the upcoming Virtual Reality (VR), Mixed Reality (MR), and Augmented Reality (AR), collectively referred to as Extended Reality (XR), applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advances in technologies and services, users are no longer satisfied with watching videos on conventional Full High Definition (FHD) or Ultra High Definition (UHD) 2D displays. Novel types of displays have been developed, such as light-field displays, volumetric displays, Head-Mounted Displays (HMDs), and Optical Head-Mounted Displays (OHMDs). In particular, Commodity HMDs, e.g., Oculus Rift [155], HTC Vive [75], and Samsung Gear VR [178], have become increasingly widespread. A market report [130] claims that while the global HMD market is valued at US$6,744 in 2020, a Compound Annual Growth Rate (CAGR) of 22% is forecast from 2020 to 2026. These advanced technologies are making immersive applications more and more popular, including Virtual Reality (VR), Augmented Reality (AR), Mixed Reality (MR), and Extended Reality (XR).

Among the immersive content, 360° videos have gradually entered into our daily life. 360° videos allow viewers to dynamically change their orientations during video playback. Since 2015, online content providers, such as YouTube and Facebook, have started streaming 360° videos over the Internet, and these videos are becoming very popular. For example, Hong Kong Airlines' 360° video advertisement has attracted 35 times more viewers than their regular one [1]. A user study also found that: (i) 360° videos attract 8 times more web clicks, and (ii) 360° video viewers watch 29% longer on average, compared to conventional videos [128]. Moreover, the market of 360° cameras is predicted to have a CAGR of 25% during 2020-2025 [176]. The momentum of the increasing popularity of watching 360° videos shows no indication of slowing down in the coming years.

While 360° videos are becoming increasingly popular, offering high quality 360° videos is no easy task. In particular, 360° videos provide a much wider view than conventional videos and thus must be encoded at extremely high resolutions for good viewing experience. This is mainly because each viewer only sees a small viewport of about $100^\circ \times 100^\circ$ within his/her 360° video at any moment. To ensure that every viewport has at least 1080p (1920×1080) resolution, the whole 360° video needs to be encoded at a resolution close to 8k (7680×4320), following some back-of-envelope calculations. Streaming complete 360° videos in 8k resolution to HMD viewers consumes a staggering amount of resources, including encoding/decoding, networking, and storage, and is therefore vulnerable to degraded playback quality.

To address this challenge, tiled encoding has been proposed [134]. It divides every 360° video into equal-size rectangles, called *tiles*. The tiles are then independently encoded and decoded, so that they can be selectively streamed to HMD viewers based on their likelihood to be viewed (i.e., falling in HMD viewers' viewports). To further deal with the diverse and dynamic network environments, the Dynamic Adaptive Streaming over HTTP (DASH) [83, 188] *segment* in the time domain is adopted to combine with the tiling concept in the spatial domain, which encodes 360° videos into *tiled segments*. Each tiled segment represents a spatial region for a time duration of a few seconds. Adopting DASH requires the predetermination of a set of encoding configurations, called *an encoding ladder*, to generate a set of representations stored on the streaming server. These representations are adaptively requested by clients according to their network conditions during streaming sessions.

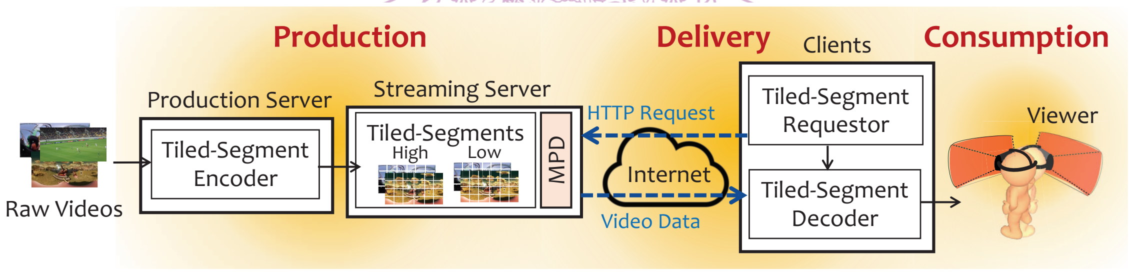


Figure 1.1: Overview of a typical tiled 360° video streaming system.

Fig. 1.1 illustrates an overview of a typical tiled 360° video DASH streaming system offered by a content distributor. Content distributors receive production-quality (raw) videos from *content providers*, prepare multiple representations of segments, and distribute the videos over the Internet to clients. The streaming service contains three entities: (i) *production server*, which produces the encoded tiled-segments using the tiled-segment encoder, (ii) *streaming server*, which stores the encoded tiled-segments and sends the video streams, and (iii) *clients*, which request, decode, and render the tiled-segments to viewers. These three entities compose three crucial phases of 360° video streaming: (i) production, which is triggered only when new videos are added to the streaming server, (ii) delivery, which is triggered at individual streaming sessions when the clients request tiled segments from the streaming server, and (iii) consumption, which is triggered when the viewers watch the videos rendered with the received and decoded tiled segments. Exploiting tiled segments enables wider room to maximize the Quality of Experience (QoE) of 360° video streaming to HMDs, which dictates careful optimization of the above three phases. In this thesis, we solve the problem for these three phases to achieve a QoE-optimized 360° video streaming system.

1.1 Delivery Optimization: Fixation Prediction

In delivery, we study the problem of predicting the viewing probability of different parts of 360° videos when streaming to HMDs. We propose a fixation prediction network based on a Recurrent Neural Network (RNN), which leverages sensor and content features. The content features are derived by Computer Vision (CV) algorithms, which may suffer from inferior performance due to various types of distortion caused by diverse 360° video projection models. We propose a unified approach with overlapping virtual viewports to eliminate such negative effects, and we evaluate our proposed solution using several CV algorithms, such as saliency detection, face detection, and object detection. We find that overlapping virtual viewports increase the performance of these existing CV algorithms that were not trained for 360° videos. We next fine-tune our fixation prediction network with diverse design options, including: (i) with or without overlapping virtual viewports, (ii) with or without future content features, and (iii) different feature sampling rates. We empirically choose the best fixation prediction network and use it in a 360° video streaming system. We conduct extensive trace-driven simulations with a large-scale dataset to quantify the performance of the 360° video streaming system with different fixation prediction algorithms. The results show that our proposed fixation prediction network outperforms other algorithms in several aspects, such as: (i) achieving comparable video quality (average gaps between -0.05 and 0.92 dB), (ii) consuming much less bandwidth (average bandwidth reduction by up to 8 Mbps), (iii) reducing the rebuffering time (on average 40 sec in bandwidth-limited 4G cellular networks), and (iv) running in real-time (at most 124 ms).

1.2 Production Optimization: Optimal Laddering

In production, we solve the optimal laddering problem that determines the optimal *encoding ladder* to maximize the client viewing quality. In particular, we consider video models, viewing probability, and client distribution to formulate the mathematical problem. We use the divide-and-conquer approach to decompose the problem into two subproblems: (i) per-class optimization for clients with different bandwidths and (ii) global optimization to maximize the overall viewing quality under the storage limit of the streaming server. We propose two algorithms for each of the per-class optimization and global optimization problems. Analytical analysis and real experiments are conducted to evaluate the performance of our proposed algorithms, compared to other state-of-the-art algorithms. Based on the results, we recommend a combination of the proposed algorithms to solve the optimal laddering problem. The evaluation results show the merits of our recommended algorithms, which: (i) outperform the state-of-the-art algorithms by up to 52.17 and 26.35 in Viewport Video Multi-Method Assessment Fusion (V-VMAF) in per-class optimization, (ii) outperform the state-of-the-art algorithms by up to 43.14 in V-VMAF for optimal laddering in global optimization, (iii) achieve good scalability under different storage limits and number of bandwidth classes, and (iv) run faster than the state-of-the-art algorithms.

1.3 Consumption Optimization: QoE Modeling

Conducting user studies to quantify the QoE of watching the increasingly more popular 360° videos in HMDs is time-consuming, tedious, and expensive. Deriving QoE models, however, is very challenging because of the diverse viewing behaviors and complex QoE features and factors. In consumption, we compile a wide spectrum of QoE features and factors that may contribute to the overall QoE. We design and conduct a user study to build a dataset of the overall QoE, QoE features, and QoE factors. Using the dataset, we derive the QoE models for both Mean Opinion Score (MOS) and Individual Score (IS), where MOS captures the aggregated QoE across all subjects, while IS captures the QoE of individual subjects. Our derived overall QoE models achieve 0.98 and 0.91 in Pearson's Linear Correlation Coefficient (PLCC) for MOS and IS, respectively. Besides, our analysis of the user study results leads to new observations as follows: (i) content factors dominate the overall QoE across all factor categories, (ii) VMAF is the dominating factor among content factors, and (iii) the perceived cybersickness is affected by human factors more than others. Our proposed user study design is useful for QoE modeling (in specific) and subjective evaluations (in general) of emerging 360° tiled video streaming to HMDs.

1.4 Contributions

The contributions of this thesis are listed below:

- **Delivery optimization: fixation prediction** [50, 52]

  - We are the first to jointly take both sensor and content features as the inputs to a neural network to predict the viewer fixation.
  - We propose a unified approach based on overlapping virtual viewports to turn CV algorithms designed and trained for 2D images/videos applicable to 360° videos.
  - Our proposed fixation prediction network outperforms a state-of-the-art algorithm in the literature in terms of prediction accuracy.
  - We employ the fixation prediction network and virtual viewport approach to optimize our 360° video streaming system. The evaluation results show the superior performance of our solution compared to the baseline approaches, i.e., our solution (i) achieves comparable video quality, (ii) consumes much less bandwidth, (iii) reduces the rebuffering time, and (iv) runs in real-time.

- **Production optimization: optimal laddering** [53]
  - We formulate the optimal laddering problem for 360° videos into a mathematical optimization problem considering video models, viewing probability, and client distribution.
  - We solve the problem using the divide-and-conquer approach with a diverse suite of mathematical tools. We also analytically analyze the performance of our proposed algorithms.
  - We conduct extensive experiments to show the performance and practicality of our proposed algorithms. Our evaluation results reveal that our algorithms: (i) result in both higher objective and subjective video quality, (ii) scale well under different storage limits and different numbers of bandwidth classes, and (iii) run faster than the state-of-the-art algorithms.

- **Consumption optimization: QoE modeling**
  - We compile a suite of 5 QoE features and 30 QoE factors, which are either inspired by the literature or from the tiled streaming nature.
  - We design and conduct a user study for watching tiled 360° videos in HMDs. We also build MOS and IS models of watching tiled 360° videos in HMDs.
  - Our evaluation results are fairly promising: e.g., our derived models for the overall QoE achieve 0.988 and 0.915 in PLCC in terms of MOS and IS, respectively.

– We find that a reduced set of QoE factors could also lead to good performance. For example, for the overall QoE, with a single QoE factor, we achieve >97% of the performance compared to comprehensive models with 30 QoE factors.

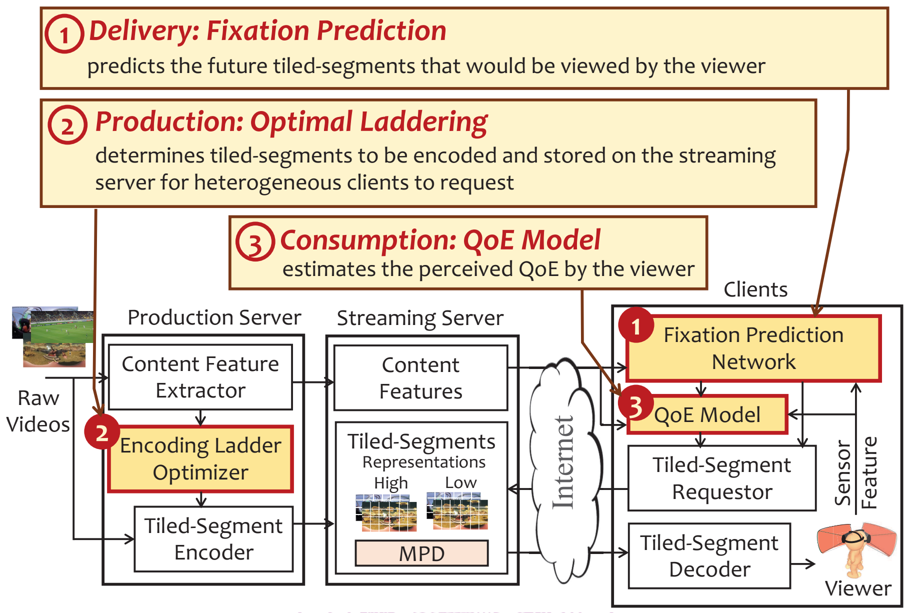

1 Delivery: Fixation Prediction
predicts the future tiled-segments that would be viewed by the viewer
2 Production: Optimal Laddering
determines tiled-segments to be encoded and stored on the streaming server for heterogeneous clients to request
3 Consumption: QoE Model
estimates the perceived QoE by the viewer
Raw Videos
Production Server
Content Feature Extractor
2 Encoding Ladder Optimizer
Tiled-Segment Encoder
Streaming Server
Content Features
Tiled-Segments Representations
High
Low
MPD
Internet
Clients
1 Fixation Prediction Network
3 QoE Model
Tiled-Segment Requestor
Tiled-Segment Decoder
Sensor Feature
Viewer

Figure 1.2: Overview of our proposed optimized 360° video streaming system.

With the above contributions, we believe that we can achieve a QoE-optimized 360° video streaming system, as illustrated in Fig. 1.2. In delivery optimization, we develop the fixation prediction neural network leveraging both sensor and content features to predict the viewer's future viewed tiled segments. This avoids wasting resources on streaming unwatched parts. In production optimization, we develop the encoding ladder optimizer to determine the optimal encoding ladder considering content features (video models), viewing probability, and bandwidth distribution. In consumption optimization, we conduct a comprehensive user study to investigate diverse factors that capture user behaviors and affect user experience of watching tiled 360° videos. The collected dataset from the user study is used to build *QoE models* in various aspects, such as the overall QoE, immersion level, and cybersickness level. Our encoding ladder optimizer ensures the optimal resource allocation on the streaming sever, while the fixation prediction network and QoE models ensure the optimal resource allocation during streaming.

1.5 Thesis Organization

The organization of this thesis is given below. We report the related work for our proposed three core problems in Chapter 2. We solve the delivery problem in Chapter 3, the production problem in Chapter 4, and the consumption problem in Chapter 5. Finally, we conclude this thesis and discuss the future directions in Chapter 6.

Chapter 2

Background and Related Work

In this chapter, we first introduce the background of tiled 360° video streaming to HMDs [51]. Then, we zoom into the related work of the three core problems studied in this thesis.

2.1 Background

2.1.1 Off-the-Shelf Hardware

Table 2.1: The State-of-the-art 360° Cameras

Name	Release	Lenses	Video Res.	Image Res.	OS Support	MSRP ($)
Insta360 Evo	2018	200° (f/2.2) x2	5760 x 2880/30 fps 3008 x 1504/100 fps	6080 x 3040	iOS	420
Insta360 One X	2018	200° (f/2.0) x2	5760 x 2880/30 fps	6080 x 3040	iOS	515
Yi 360 VR	2018	220° (f/2.0) x2	5760 x 2880/30 fps	5760 x 2880	iOS, Android	200
Insta360 Nano S	2018	210°(f/2.2) x2	3840 x 1920/30 fps	6272 x 3136	iOS	239
Insta360 One	2017	230°(f/2.2) x2	3840 x 1920/30 fps 2048 x 512/120 fps	6912 x 3456	iOS, Android	299
Insta360 Air	2017	210°(f/2.0) x2	2560 x 1280 /30 fps	3008 x 1504	Android	119
SAMSUNG Gear 360	2017	360°(f/2.2) x2	4096 x 2048/24 fps 1920 x 1080/60 fps	5472 x 2736	iOS, Android, Windows, OS X	130
LG 360 cam	2016	206°(f/1.8) x2	2560 x 1280/30 fps	5660 x 2830	iOS, Android	90
RICOH THETA V	2017	- (f/2.0) x2	3840 x 1920 /30 fps	5376 x 2688	iOS, Android	397
RICOH THETA S	2015	- (f/2.0) x2	1920 x 1080 /30 fps	5376 x 2688	iOS, Android	299
RICOH THETA SC	2016	- (f/2.0) x2	1920 x 1080 /30 fps	5376 x 2688	iOS, Android	189
GARMIN VIRB 360	2017	201°(f/2.0) x2	3840 x 2160 /30 fps	5640 x 2816	Windows, OS X	800
Rylo 360-degree camera	2017	208°(f/2.8) x2	4K/30 fps	6K	iOS, Andriod	499
KODAK ORBIT360 4K	2017	235°(f/2.4) x2	1920 x 960/30 fps	7360x3680	iOS, Andriod	399
KODAK SP360 4K	2016	235°(f/2.8) x2	2880 x 2880/30 fps	2880 x 2880	iOS, Android	449
KODAK SP360	2014	214°(f/2.8) x2	1920 x 1080/30 fps	3264 x 3264	iOS, Android	174
GoPro Fusion	2017	180°(f/2.0) x2	4992 x 2496/30 fps	5760 x 2880	iOS, Android	699
Nikon KEYMISSION 360	2016	180°(f/2.0) x2	1440 x 960p/30 fps	7744 x 3872	iOS, Android	497
Insta360 Pro 2	2018	200°(f/2.4) x6	7680 x 7680/30 fps (3D) 3840 x 3840/120 fps	7680 x 7680 (3D) 7680 x 3840 (2D)	iOS, Android, Windows, OS X	4,499
Insta360 Pro	2017	200°(f/2.4) x6	3840 x 1920/30 fps	7680 x 3840	iOS, Android, Windows, OS X	3,499
VideoStitch Orah 4i	2017	170°(f/2.0) x4	4096 x 2048/30 fps	1920 x 1440	iOS, Android	3,595

One crucial driving force of the increasing popularity of 360° video streaming is the availability of consumer-grade hardware components, especially the 360° cameras and HMDs. We summarize the state-of-the-art hardware in this section.

Table 2.2: The State-of-the-art HMDs

Name	Year	Screen	Resolution	FoV	Ref. Rate (Hz)	MSRP ($)
FOVE 0	2017	OLED	2560 x 1440	100°	70	599
Oculus Go	2018	LCD	2560 x 1440	110°	90	209
Oculus Rift	2016	OLED	2160 x 1200	110°	90	421
Samsung Gear VR	2017	AMOLED	2560 x 1440	101°	60	102
Samsung Odyssey	2017	AMOLED	2880 x 1600	110°	90	420
Sony PlayStation VR	2016	OLED	1920 x 1080	100°	120	215
HTC Vive FOCUS	2018	AMOLED	2880 x 1600	110°	75	599
HTC Vive Pro	2018	AMOLED	2880 x 1600	110°	90	799
HTC Vive	2016	OLED	2160 x 1200	110°	90	399
OSVR HDK2	2016	OLED	2160 x 1200	110°	90	300
OSVR HDK1	2015	OLED	1920 x 1080	100°	60	-
Google DayDream View VR	2017	Using the inserted cellphone				87
Google Cardboard	2014	Using the inserted cellphone				15

360° cameras employ multiple lenses to capture 360° images and videos. The first consumer-grade 360° cameras hit the market in 2014, but were extremely expensive until 2016. Ever since, a growing number of 360° cameras, especially those for iOS and Android, have been released at affordable prices. Table 2.1 lists representative 360° cameras on the market. Several characteristics, such as the maximal resolution and release year, are reported in this table. We note that the MSRPs (Manufacturer's Suggested Retail Prices) are retrieved from Amazon at the time of writing. We roughly classify the 360° cameras into two groups: *professional* and *consumer-grade*. The professional ones are $>$ USD 3000, and typically have more than two lenses. Besides, they have larger form-factors and are heavier. Most professional 360° cameras are standalone rather than clipped onto smartphones. The consumer-grade cameras focus more on user interfaces and can be attached to smartphones or other mobile devices running iOS or Android. Furthermore, most of them have no more than two lenses in order to reduce the cost.

HMDs have been used in a wide range of scenarios, including military, medicine, video gaming, and sports. In Table 2.2, we present some of the most influential HMDs on the market. We highlight critical specifications, such as the resolution, FoV, and refresh rate. Such information may be useful for researchers, developers, and users to choose the HMD that is most appropriate for their usage scenarios.

2.1.2 Existing Systems

Multiple studies developed 360° video systems for various purposes, such as demonstrating the practicality or evaluating the optimization ideas. Table 2.3 summarizes and compares the prototype systems proposed in the literature. Some 360° video systems render videos stored in local storage spaces. For example, Petry and Huber [166] presented a 360° display system that allows viewers to interact with the video through an HMD and a mounted gesture recognizer. In particular, the viewers are able to play, pause, forward, and rewind the video by performing different mid-air hand gestures. Alface et al. [4] proposed a system that is able to handle 16K videos by only processing the required pixels in the viewer's viewport. In particular, they composed a moving 4K canvas that always keeps the viewport at the center of the canvas. In this way, the processed video is always in 4K resolution, which reduces the demands for processing power. Anderson et al. [7] developed a more comprehensive 360° video system that comprises capturing, stitching, and rendering components. They first mounted cameras on a rig, so that all light rays from the viewer's eyes are recorded as 360° stereo videos. All 16 videos were stitched using their proposed optical flow and composition algorithms. Ferworn et al. [58] developed a special-purpose 360° video system for dogs to perform urban search and rescue. In their proposed system, a dog wears multiple cameras, which keep capturing video frames. These captured video frames are uploaded to a Personal Computer (PC) after the mission is over. Next, the PC analyzes videos, performs tracking, and stabilizes frames for better-quality 360° videos.

There were more systems developed with Internet streaming supports. Gaemperle et al. [63] adopted a multi-camera system with image blending algorithms to capture 360° videos. The video was then distributed through the server to the client, and the HMD viewport was reconstructed by the client. Several studies tried building low-cost streaming systems for 360° videos. Canessa and Tenze [26] developed a 360° video Real-time Transport Protocol (RTP) streaming system on Raspberry Pi with a fish-eye camera module and several open-source packages, such as FFmpeg, OpenCV, and MPlayer. However, the resolution of their system is only about 360p. Choi and Jun [32] further considered multiple camera-equipped Raspberry Pis. They sent the images over a network to a PC for stitching. Several optimization tools have been employed to improve the system performance, including simplified algorithm and multi-threading. They tried different blending methods proposed in the literature and quantified their performance. Jiang et al. [92] built a 360° video streaming system for power consumption measurement. They compared 360° video streaming systems with conventional 2D ones. The measurement results showed that viewport generations consume the most power due to the high computation overhead for viewports. The network transmission consumed the second most, followed

Table 2.3: Representative 360° Video Streaming Prototype Systems

Networked 360° Video Streaming					
System	**Maximum Resolution**	**Codec**	**Streaming Protocol**	**Adaptive Streaming**	**Player**
Gaemperle et al. [63]	1024x768	-	TCP	-	Homebrew
Canessa and Tenze [26]	352x288	-	RTP	-	MPlayer [137]
Choi and Jun [32]	$<$ 3840x720	H.264	RTSP	-	-
Ochi et al. [153]	2560x1280	H.264	HTTP	-	Oculus Player [155]
Ochi et al. [154]	1920x960	H.264	RTMP	-	Oculus Player [155]
Qian et al. [168]	1080p	AVC	HTTP/1.1	DASH	YouTube Player
Xie and Zhang [219]	4K	VP8	RTP	WebRTC	-
Schafer et al. [179]	10000x1920	H.265	HTTP/1.0	DASH	-
Lim et al. [118]	-	H.264	HTTP/1.0	DASH/SRD	Homebrew
Feuvre et al. [111]	4K	HEVC	HTTP/1.1	DASH/SRD	GPAC/MP4Client [195]
Ozcinar et al. [161]	8K	AVC	HTTP/1.1	DASH/SRD	WebVR Player [211]
Lo et al. [124]	4K	HEVC	HTTP/1.1	DASH/SRD	GPAC/MP4Client [195]
Graf et al. [66]	4K	HEVC	HTTP/1.1	DASH/SRD	WebVR Player [211]
Kim et al. [102]	8K (4K/4K)	HEVC	HTTP/1.1	DASH/SRD	-
Nasrabadi et al. [139]	4K	HEVC/SHVC	HTTP/1.0	-	Homebrew
Petrangeli et al. [165]	4K	HEVC	HTTP/2	DASH/SRD	Google/Exoplayer [65]
Niamut et al. [149]	8K	-	HTTP/1.0	DASH	Homebrew
Gaddam et al. [62]	4K	H.264	HTTP/1.0	-	-
Local 360° Video Streaming (Files)					
Inoue et al. [82]	6400x1280	H.264/MVC	-	-	-
Kimata et al. [104]	8000x1000	H.264/MVC	-	-	-
Kimata et al. [105]	5000x1000	H.264/MVC	-	-	-
Alface et al. [4]	16K	H.264	-	-	GearVR Player [178]
Anderson et al. [7]	8192x8192	-	-	-	-
Ferworn et al. [58]	1080p	-	-	-	-

by the screen displays and video codecs. Based on their observations, they proposed several power-saving approaches, e.g., viewport-based streaming and edge-based rendering.

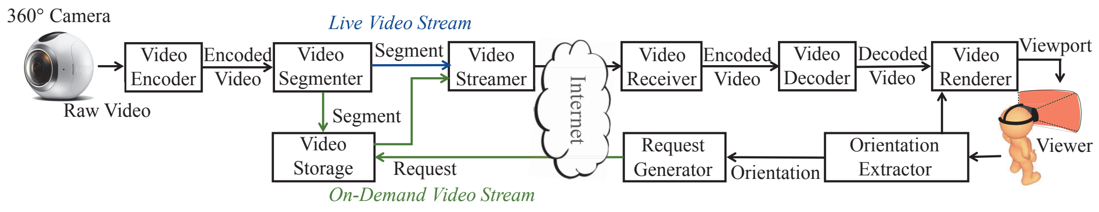
Several other systems further considered tiling for improving the performance of encoding or streaming 360° videos. Ochi et al. [153, 154] built a 360° video streaming system that streamed high-bitrate tiles to the viewer's viewports and low-bitrate tiles to other parts, for reduced bandwidth consumption. However, the latency of their proposed streaming system still has room for improvement. Qian et al. [168] presented a streaming system over cellular networks, which only streams the tiles in the viewport based on head movement predictions. Xie and Zhang [219] presented an interactive 360° streaming system over cellular networks. They developed a conservative compression strategy to keep the quality of the viewport more stable, even when the variation of viewports is high. Besides, their proposed system monitors the buffer occupancy of the uplink for congestion detection. Once the congestion is detected, the encoding bitrate is adjusted to maintain the video quality. Schafer et al. [179] developed a 360° video capturing and streaming system. Their system split videos into several sub-videos and performed stitching in the compressed domain so that the client only needed to decode one stream.

Different from the above systems, several studies split the videos into equal-size tiles and stored the tile information, such as resolution and position in metadata files [111,118]. Feuvre et al. [111] applied tiling to general 4K videos with unequal tile quality levels to achieve viewport-aware adaptive streaming. Ozcinar et al. [161] also developed an algorithm to select the representation of each tile to achieve viewport-aware adaptive streaming in their proposed system. Lo et al. [124] built a streaming system and compared the performance of transmitting all tiles versus visible tiles only. They further studied the impact of tiling, e.g., coding efficiency and tiling overhead. Graf et al. [66] built a tiled-streaming system considering different streaming strategies. Kim et al. [102] developed a streaming system for 360° video in virtual spaces. In particular, they adopted virtual cameras in Unity and Unreal engines. Scenes from 12 virtual cameras were captured and stitched to generate the 360° videos. The videos were then tiled and segmented for adaptive streaming through CDN (Content Delivery Network). Nasrabadi et al. [139] exploited Scalable High-efficiency Video Coding (SHVC) to further adaptively stream 360° videos with multiple layers. The base-layer of all tiles is prefetched to avoid video stalls, while the enhancement-layer tiles in viewports are transmitted with the residue bandwidth. Combined with state-of-the-art network technologies, Petrangeli et al. [165] leveraged HTTP/2.0 and OpenFlow to reduce latency and avoid network congestion, respectively.

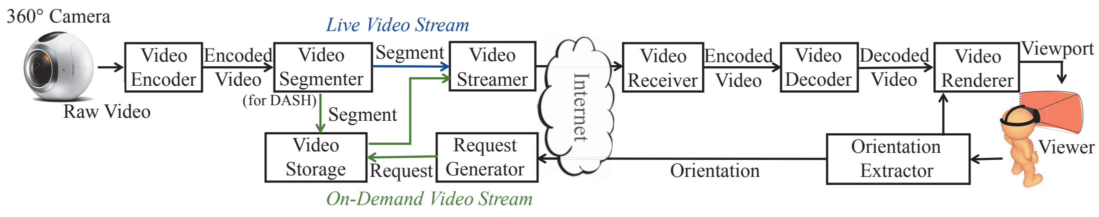
Similar to 360° videos, live broadcast events, such as soccer or basketball games, are often streamed as panorama videos. These videos contain wider horizontal viewing angles (which may be less than 360°). Since most live events have a single Region-of-Interest (RoI), tiles are also used in panorama videos to support zooming [62,82,104,105,149]. Inoue et al. [82] developed a tiled-streaming system based on Multi-View Coding (MVC) to support tiles. A rate-quality mapping table was used for determining the transmitted viewports to maximize the visual quality under restricted bandwidth. Kimata et al. [104] proposed an interactive panorama video streaming system allowing viewers to control their viewports for high-quality videos and sound. Redundant viewports are also adaptively streamed to guarantee smooth and fast switches of viewports. Some extensions of this system were proposed in Kimata et al. [105]. They extended the system to support mobile and multiple devices. In particular, the viewers are allowed to interact, e.g., zoom in/out, with their hand-held mobile devices while watching high-resolution panorama videos on larger screens. Niamut et al. [149] proposed an end-to-end panorama streaming system with acquisition, transmission, and display components. In their system, the scenes are captured with multiple representations, i.e., resolutions and frame rates. Some analysis, such as saliency detection and person tracking, is performed for automatic camera selection for serving a larger number of viewers. The videos are encoded with multiple representations and are streamed to the clients. Their system supports several gesture interactions to control the viewports and the playbacks. For generating virtual views for individual clients, Gaddam et al. [62] proposed to leverage tiling for quality allocation and GPUs for rendering acceleration.

The abovementioned 360° video streaming systems comprise different subsets of computation and networking components, from playing local 360° video files to streaming 360° videos over the Internet. Insights gathered in these studies are beneficial to engineers who plan to build similar systems, and to researchers who plan to evaluate their proposed solutions using real testbeds.

2.1.3 General 360 Video Streaming Framework



(a)



(b)

Figure 2.1: General 360° video streaming systems: (a) pull-based (like HTTP/1.1) and (b) push-based (like RTP).

Fig. 2.1 shows two 360° video streaming frameworks: (i) pull-based (client-driven) (Fig. 2.1(a)), which puts the intelligent components at the client side to determine the requested video segments, and (ii) push-based (server-driven) (Fig. 2.1(b)), which, in contrast, puts them at the server side to decide which segments to push to the client. We describe the functions of the individual components in Fig. 2.1 below:

- **Video encoder** encodes the captured 360° videos. It may further support tiling for partially streaming and rendering to save bandwidth consumption.
- **Video segmenter** splits the encoded video into segments, where each segment contains a few consecutive video frames and lasts for a few seconds.
- **Video storage** stores the encoded video on the server, which is used for on-demand video streaming. It may store several versions of the encoded video at different quality levels to support adaptive video streaming.
- **Video streamer** is responsible for sending the encoded video to the client.
- **Orientation extractor** computes viewer orientations using inputs from HMD sensors or other devices.
- **Request generator** generates requests of videos or video segments. It is placed at the client side in pull-based systems and at the server side in push-based systems. It is usually the core component that makes decisions for optimizing the streaming system. For example, a bitrate allocation algorithm can be implemented in this component for generating video segment requests based on network conditions.
- **Video receiver** receives the video streamed from the server.
- **Video decoder** is responsible for decoding the encoded videos.
- **Video renderer** computes the viewports according to the viewer's orientation. It may also convert among 360° video projection models [49], e.g., equirectangular, equal-area, and cube.

The interactions among the pull-based components in Fig. 2.1(a) are as follows. At the server side, the 360° camera sends a 360° video to the video encoder. It compresses this video with or without tiling support and sends the encoded video to the video segmenter. The video segmenter splits the received video into temporal segments and either: (i) directly sends the segments to the video streamer for live video streaming or (ii) stores them in the video storage for on-demand video streaming. At the client side, the orientation extractor keeps recording the viewer's orientation, which is computed from the sensor readings of the HMD[1]. The request generator considers the current network condition and may take the viewer's orientation as input to generate the requests. For example, the adaptive streaming system can request video segments or frames at different quality levels, and the tiled-video streaming system may further skip some tiles based on the viewer orientations. The video streamer at the server side then streams the encoded videos from the video segmenter (live video) or from the video storage (on-demand video) to the client. The video receiver passes the received video to the video decoder. The video decoder then decodes the encoded video. The video renderer renders the decoded video according to the viewer's orientation to the viewer. The push-based components in Fig. 2.1(b) are similar, except that the request generator is at the server side.

The pull- and push-based systems have diverse pros and cons. The pull-based streaming systems often adopt HTTP/1.1 protocol. This protocol supports dynamic adaptive streaming using short segments, which are typically a few seconds long and independently decodable. This approach is better known as DASH (Dynamic Adaptive Streaming over HTTP). The pull-based streaming systems are not affected by the Network Address Translation (NAT) traversal problems. Besides, it is convenient to reuse the WWW infrastructure, including servers, caches, and CDNs. In DASH streaming, the media content is encoded into various versions and stored on the server beforehand to adapt to varying and dynamic networks. This consumes a large amount of storage space. In contrast, push-based streaming systems, which often adopt RTP as the transmission protocol, stream the media content to the clients without waiting for requests. This leads to lower latency compared to pull-based streaming. However, it requires the streaming server to keep track of the states of a potentially large number of clients. Without a reliable protocol like TCP, the push-based streaming systems may result in inferior video quality due to network impairments, such as insufficient bandwidth, packet loss, and NAT traversal issues. Based on the pros and cons described above, we believe that the pull-based streaming systems are more applicable for *presentational* video streaming, such as YouTube and Netflix, where a few seconds of one-way delay is acceptable [2]. In contrast, the push-based systems are more suitable for *conversational* video streaming, such as video conferencing, where ultra-low latency is required for high interactivity. Requiring both high video quality and ultra-low latency, streaming 360° videos over these two types of systems needs additional optimization tools, such as viewport prediction or resource allocation.

We have introduced the general framework of pull- and push-based 360° video streaming systems. Researchers and practitioners may start from the general frameworks and add specialized components to meet their needs.

[1]Other input devices are possible when displays other than HMDs are used.

2.1.4 Tiled 360° Videos

High Efficiency Video Coding (HEVC) [156] supports Motion-Constrained Tile Set (MCTS), where tiles are disjoint rectangular regions that can be independently decoded. Tiles allow: (i) parallel decoding for a decoder speedup to cope with high resolutions and (ii) random decoding of dynamic viewports. Tiles, however, impose constraints on the encoding process, which needs to be carefully considered. In particular, MCTS motion dictates constraint among tiles, which reduces the coding efficiency because motion vectors do not go across the tile boundaries. Therefore, there exists a critical tradeoff between the coding efficiency and the tiled streaming flexibility, which can be controlled by the number of tiles. More details on the HEVC standard are given in Sullivan et al. [191], while the details of the MCTS supports in HEVC can be found in Misra et al. [134]. It is reported that: (i) HEVC results in a 50% rate cut at similar visual quality [191] compared with AVC and (ii) tiles achieve up to a 5.5% luminance bitrate reduction [134] compared with regular slices. Due to their superior coding efficiency, HEVC and its tiling support are widely used in 360° video systems. HEVC standard does not specify the precise optimization algorithms used at the encoder side. Among existing open-source HEVC codecs, Kvazaar [204] is developed in C language and provides an option to be optimized in Assembly. Kvazaar implements various coding tools defined in HEVC, which enable parallelization on multi-core CPUs and hardware acceleration. It supports three parallel processing approaches including tiled encoding, and thus can be leveraged by 360° video testbeds.

The MPEG DASH standard includes an amendment on Spatial Representation Description (SRD), which is extended from MPD to provide x-axis, y-axis, width, and height as attributes to DASH clients. Concolato et al. [35] discuss the latest HEVC and ISO Base Media File Format (ISOBMFF) standards, which are used for encoding and encapsulating tiled videos. They demonstrate that a client may merge several tiles into a video stream, and decode it with a single decoder by combining SRD, HEV, and ISOBMF. The MPEG group developed the Omnidirectional MediA Format (OMAF) standard for the delivery and storage of 360° videos. Skupin et al. [187] presented the application requirements, projection formats, video/audio codec, and DASH integration of the OMAF standard.

Several papers [43, 66, 111] share their experience of developing standard-based tiled 360° video streaming systems. In particular, D'Acunto et al. [43] used SRD to realize navigable video streaming. They summarized the design choices of the SRD-enabled DASH player, such as: (i) definitions of representations, and (ii) seamless switches among representations. Feuvre and Concolato [111] realized tiled-based adaptive streaming using several open-source projects, including Kvazaar [204], MP4Box [194], and MP4Client [195]. They demonstrated how interactive navigation and bitrate adaptation can be achieved using DASH and SRD. Furthermore, they presented their implementation supporting diverse adaptation policies based on the open-source MP4Client. Graf et al. [66] implemented various evaluation tools to quantify the pros and cons of different encoding and streaming strategies on tiled-based 360° video streaming systems. They also discussed various options to enable the bandwidth-efficient adaptive streaming of 360° videos. In their evaluation, 6×4 tiles provide the best tradeoff between tiling overhead and bandwidth consumption, which confirms a bitrate saving of 40% compared to the baseline solutions. The aforementioned studies do not consider the diverse viewing probabilities of individual tiles.

2.2 Delivery: Fixation Prediction

In terms of the fixation prediction problem, we survey the literature in three directions: (i) 2D image/video saliency, (ii) 360° image/video saliency, and (iii) fixation/head movement prediction in HMDs.

2.2.1 2D Image/Video Saliency

Conventional fixation prediction is built on salient object detection, which has been done on different content types, such as still images [20]. Image saliency can be derived from low-level features, e.g., contrast, textures, and edges [121, 127]. Several studies have proposed to detect image saliency based on region-based models, which leverages graph-based segmentations [54] or linear iterative clustering [3]. Wang et al. [209] analyzed the image saliency by combining color and contrast with spatial priors. For example, the saliency maps are aligned to image edges or correlated with color distributions. Given training samples with ground truth, the learning-based methods are becoming popular because of their higher accuracy. Liu et al. [122] trained learning-based models using a binary labeled dataset with low-level features. In addition, deep learning has become the most popular method to perform saliency detection. For example, the Convolution Neural Network (CNN) is able to learn from low-level features parallelly, which is suitable for vision processing. Li and Yu [115] adopted a pre-trained CNN for extracting features in different scales, and performed regression on the inputs to produce saliency maps. These studies, however, were designed for 2D conventional images.

In terms of 2D video saliency, Mavlankar and Girod [132] predicted the future viewing trajectory based on extrapolation and enhanced the performance by analyzing the characteristics of video content, such as optical flow and motion vectors. A growing number of supervised learning methods are being adopted for fixation detection [6, 27, 148] to achieve better feature extraction and prediction accuracy. In particular, Chaabount et al. [27] predicted video saliency by developing a CNN with residual motion as its main features. Furthermore, they adopted transfer learning to cope with the lack of large video datasets. Their results show the positive effectiveness of the transfer learning. Alshawi et al. [6] analyzed the correlation between the saliency of pixels and their spatial/temporal neighbors, where the correlation is much affected by the video characteristics. Nguyen et al. [148] noted a close relationship between image (static) and video (dynamic) saliency. Based on their observation, they adopted both the information of image saliency (static) and camera motion to predict video saliency (dynamic).

2.2.2 360° Image/Video Saliency

The saliency detection algorithms specifically designed for 360° videos have been proposed. Assens et al. [11] developed a deep CNN to predict the scan-paths of 360° images. They trained the prediction network on a public 360° image dataset [171], which consists of 60 360° images and 63 participants with the trajectories of both their head and eye movements. In particular, they performed transfer learning by initializing the network weights from several 2D image datasets. They further analyzed the prediction under different sampling strategies, and their results revealed that limiting the distance between fixations can improve the prediction accuracy. Monroy et al. [136] first mapped the 360° images to six faces of a cubic projection, which reduced the distortion close to poles compared to the equirectangular projection. Each face is detected by a conventional saliency detection network with spherical coordinates for locating the face on the sphere. Finally, six detected saliency faces were combined into a single saliency map for the 360° image. Similarly, Cheng et al. [31] also mapped the 360° videos into cubic projection for eliminating the distortion. Some tricks, e.g., wider angle for each face and temporal model development, were introduced to improve the saliency prediction accuracy. Zhang et al. [230] developed a spherical CNN with spherical Mean Squared Error (MSE) loss function, which took the angle to the center of the sphere into consideration. Besides, the starting position for the viewer to watch the 360° video was also considered as an important feature in their model. These studies [11, 31, 136, 230] developed and trained neural networks to predict the saliency of the 360° videos. We note that the resulting prediction algorithms are *locked in* with projection models, compared to our unified approach. Nevertheless, as a future task, we may integrate their proposed solutions with our proposed fixation prediction network for a given projection model.

Table 2.4: Key References on Saliency and Fixation Prediction Algorithms

Literature	**Approach**	**Considered Features**	**Output**
Fan et al. [50, 52]	LSTM	Historical sensor data, saliency maps, and motion maps of frames	Future tile viewing probabilities
Nguyen et al. [145]	LSTM	Saliency maps and historical orientation maps of frames	Future saliency maps
Bai et al. [13]	Neural Network	Historical orientation	Future orientation
Xu et al. [221]	LSTM	Historical orientation	Future orientation
Qian et al. [167]	Regressor	Historical orientation	Future orientation
Xu et al. [223]	Regressor	Historical orientation	Future orientation
Zhang et al. [230]	Spherical CNN	Spherical video frames	Future saliency maps
Xu et al. [222]	CNN+LSTM	Historical viewer fixation trajectories, video frames	Future gaze trajectory
Hou et al. [73]	LSTM	Historical orientation	Future orientation
Hou et al. [71]	LSTM	Historical viewed tiles	Future viewed tiles
Wu et al. [214]	Spherical CNN	Video frames, viewport, and motion	Future viewport
Chen et al. [30]	CNN+LSTM	Video frames and historical orientation	Future orientation
Feng et al. [55]	CNN+LSTM	Video segment and historical orientation	Future orientation
Vielhaben et al. [203]	Regressor	Historical orientation	Future orientation
Cheng et al. [31]	CNN+Convolutional LSTM	Faces of cubic frames	Future saliency maps
Xu et al. [220]	Reinforcement Learning	Historical viewer orientation and video frames	Future head-moving directions
Feng et al. [56]	Bayes prediction	Viewer orientation and video frames	Future tile viewing probabilities
Nasrabadi et al. [140]	Extrapolation	Historical and other's orientation	Future orientation
Ban et al. [12]	KNN	Historical and other's orientation	Future tile viewing probabilities
Xie et al. [217]	SVM	Historical orientation	Viewing behavior class

2.2.3 Fixation/Head Movement Prediction in HMDs

Our work goes beyond saliency detection to predict viewer fixation for 360° video streaming to HMDs. There are a few studies that share a similar goal to ours. Ban et al. [12]

Table 2.5: Key References on Saliency and Fixation Prediction Algorithms

Literature	Approach	Classification	Considered Features	Output
Fan et al. [50, 52]	LSTM	No	Historical sensor data, saliency maps, and motion maps of frames	Future tile viewing probabilities
Nguyen et al. [145]	LSTM	No	Saliency maps and historical orientation maps of frames	Future saliency maps
Bai et al. [13]	Neural Network	No	Historical orientation	Future orientation
Xu et al. [221]	LSTM	No	Historical orientation	Future orientation
Qian et al. [167]	Regressor	No	Historical orientation	Future orientation
Xu et al. [223]	Regressor	No	Historical orientation	Future orientation
Zhang et al. [230]	Spherical CNN	No	Spherical video frames	Future saliency maps
Xu et al. [222]	CNN+LSTM	No	Historical viewer fixation trajectories, video frames	Future gaze trajectory
Hou et al. [73]	LSTM	No	Historical orientation	Future orientation
Hou et al. [71]	LSTM	No	Historical viewed tiles	Future viewed tiles
Wu et al. [214]	Spherical CNN	No	Video frames, viewport, and motion	Future viewport
Chen et al. [30]	CNN+LSTM	No	Video frames and historical orientation	Future orientation
Feng et al. [55]	CNN+LSTM	No	Video segment and historical orientation	Future orientation
Vielhaben et al. [203]	Regressor	No	Historical orientation	Future orientation
Cheng et al. [31]	CNN+Convolutional LSTM	No	Faces of cubic frames	Future saliency maps
Xu et al. [220]	Reinforcement Learning	No	Historical viewer orientation and video frames	Future head-moving directions
Feng et al. [56]	Bayes prediction	Clustered by video content and viewer behavior	Viewer orientation and video frames	Future tile viewing probabilities
Nasrabadi et al. [140]	Extrapolation	Clustered by viewer behavior	Historical and other's orientation	Future orientation
Ban et al. [12]	KNN	Per video	Historical and other's orientation	Future tile viewing probabilities
Xie et al. [217]	SVM	Per video	Historical orientation	Viewing behavior class

proposed to predict the viewer's head movements in three steps. First, an initial prediction was performed based on the viewer's previous viewing position using linear regression. Second, K-Nearest-Neighbors (KNN) were calculated to find the nearest K view points among all other viewers. This is based on the assumption that that most viewers would be interested in the same objects/areas in 360° videos. Last, the viewport region of the K nearest view points is calculated to finalize the predicted viewing probability of each tile. This study is considered as one of our baselines for comparison. Not only consider the previous viewing positions, Xu et al. [220] further took video content into account. In particular, they developed the head movement prediction network based on Reinforcement Learning (RL) considering the previous viewer orientation and frame content. Their network aimed to predict which direction among the eight directions (top, left, bottom, right, and the direction between each two of the above) the current viewer will move to. However, their study only predicted the future head moving direction, which may be insufficient to be applied to tiled streaming systems. Nguyen et al. [145] employed the same fixation prediction network architecture as Fan et al. [50] that considered both the orientation and the detected saliency of frames. They made three major changes: (i) they developed their own image saliency network trained on their own 360° video viewing dataset, (ii) they did not consider the motion map in our work, and (iii) they represented the orientation data as orientation maps instead of the raw sensor values of *yaw*, *roll*, and *pitch* used in our work. Several studies [30,55,56,71,73,140,167,203,214,217,221–223] focusing on the same goals as ours have been published after our work. Table 2.5 summarizes the approach, considered features, and the prediction outputs of each related work. Our work [50] is the first to leverage both sensor and content features as the inputs to the neural network.

2.3 Production: Optimal Laddering

In terms of optimal laddering, we survey the literature in the following directions: (i) viewport-adaptive tiled streaming, (ii) ABR algorithms, and (iii) bitrate allocation and optimal laddering algorithms

2.3.1 Viewport-Adaptive Tiled Streaming

Several studies proposed to stream the tiles in the viewports at a higher quality, and other tiles at a lower quality, in order to reduce the bandwidth consumption. For example, Zare et al. [229] adopted HEVC tiled-streaming and proposed three heuristic schemes for 360° video streaming to HMDs. Their experiment results confirmed that their solution utilized common patterns of head movements to achieve better coding efficiency. Ju et al. [94] streamed low-resolution tiles for the whole 360° videos along with high-resolution tiles for the viewports. They also proposed to consider the heatmap of viewers' attentions and stream tiles with higher viewing probability using broadcast. Corbillon et al. [38] took diverse projection models into consideration, and varied both bitrates and viewports when encoding 360° videos into multiple representations. HMD viewers then requested the proper presentations via DASH. Duanmu et al. [47] encoded each 360° video into a base and multiple enhancement representations. They adopted separate buffers for different representations, and gave the highest priority to the base representation for smooth playback. The residual bandwidth was then used to stream enhancement representations. The aforementioned studies only differentiated the quality of tiles in an ad-hoc way without intelligently allocating resources among tiles.

2.3.2 Adaptive BitRate (ABR) Algorithms

The ABR algorithm [15, 44] is a key client-side component in the delivery phase, which adaptively selects the representations that minimize the video distortion without congesting the networks. ABR algorithms have been studied for conventional videos [108]. Several studies [5,147,161,164,167,216,226] designed ABR algorithms for tiled 360° videos. Ozcinar et al. [161] considered unequal-size tiles. In their algorithm, higher quality level is selected for the tiles in the viewport. The quality levels of the remaining tiles are gradually reduced as the distance between them and the viewport increases. Xie et al. [216] formulated the ABR for 360° videos into an ILP problem considering the viewing probability of each tile. Their objective was to minimize the expected video distortion in terms of MSE and spatial quality variance. Besides, a buffer-based rate control mechanism was proposed for smooth playback. Alface et al. [5] proposed a greedy algorithm to select the quality levels according to the ratio between their considered utility function and the tile size, where the utility function is the estimated quality times the viewing probability. Their considered viewing probability is predicted from the previous viewport and filtered by a Gaussian filter, where the standard deviation σ is proportional to the delay.

A few studies [147, 164, 167] have grouped tiles into a number of classes and assigned the same quality level to the tiles in the same class. Petrangeli et al. [164] grouped the tiles into three classes: (i) viewport, (ii) extended area, and (iii) background. Their algorithm estimated the available bandwidth and selected the highest affordable representations for viewport, extended area, and background tiles in that order. Similarly, Nguyen et al. [147] also grouped the tiles into three classes. The size of their extended area is proportional to the estimated viewport prediction error. Their algorithm then searches all possible quality levels for the maximal estimated viewing quality under the constraint of the available bandwidth. Qian et al. [167] grouped the tiles into four classes. They searched for all the possible quality levels for each class and selected the one with the highest considered utility, which aimed for higher viewing quality and fewer stalls. A measurement study quantitatively compared the above ABR algorithms under various conditions [146].

2.3.3 Bitrate Allocation and Optimal Laddering Algorithms

In addition to the ABR algorithms that selected the representations stored on the streaming server, some studies [28, 36] have proposed algorithms for bitrate allocation that encoded and streamed tiles at different bitrates to the clients under the constraint of a given bandwidth. Corbillon et al. [36] proposed an algorithm to allocate bitrates within a 360° video [215]. Their proposed algorithm can be extended for tiled 360° videos. In particular, they classified the tiles into two categories based on the viewport, and assigned an even bitrate to the tiles in each category. A bitrate gap was introduced to restrict the quality differences between the two categories to avoid sudden quality changes. Their work only adopted two quality levels without considering the tile characteristics. Chakareski et al. [28] studied an RDO problem for streaming tiled 360° videos. They took viewing probability into account and employed convex optimization to solve the bitrate allocation problem. They did not present the detail of convex optimization nor practical concerns, such as discrete and limited QP values. In contrast, we give detailed proofs and propose a rounding algorithm. These studies [28, 36] focused on live bitrate allocation that only considers the constraint of a target bandwidth. Ozcinar et al. [160] shared a similar goal to ours. That is, they studied the optimal laddering problem for tiled 360° videos. They formulated the problem into ILP and decided the number of representations for each bandwidth class according to its fraction of clients. However, they did not develop efficient algorithms to solve the problem, which incurs an extremely long running time. Moreover, they ignored the characteristics of each video tile, such as the complexity level and viewing probability, and did not vary the quality across tiles. In contrast, we propose efficient algorithms and take the diverse characteristics of the tiles into consideration. These studies [28, 36, 160] are considered as the state-of-the-art algorithms in our evaluations.

2.4 Consumption: QoE Modeling

In terms of QoE modeling, we survey the literature in the following three categories: (i) QoE measurements, (ii) QoE factors, and (iii) QoE modeling.

2.4.1 QoE Measurements

QoE measurements on videos have been done in the past decades [96], while QoE measurements on 360° videos are attracting increasingly more attention. Upenik et al. [200] proposed a subjective evaluation testbed for viewing 360° images and videos. The testbed adopted mobile devices for rendering videos, which can be turned into HMDs, like Google Cardboard. Regal et al. [174] developed the VR player using Unity and integrated a virtual QoE questionnaire with the VR player. Singla et al. [184] conducted a user study on 360° videos with various bitrates. FHD and UHD videos were compared in their study. Bessa et al. [19] made the QoE comparisons on viewing 2D and 3D (stereoscopic) 360° videos. Their results revealed that watching 3D 360° videos delivered similar QoE with 2D ones, in terms of both immersion and cybersickness. Singla et al. [186] studied the impact of resolution, bandwidth, and delay on the perceived quality and cybersickness. Upenik et al. [201] recruited 45 subjects to rate 360° images. The correlation between the collected QoE scores and existing objective metrics, such as Spherical Peak Signal-to-Noise Ratio (S-PSNR) and Viewport PSNR (V-PSNR), was analyzed. Their results showed that these metrics have a low correlation with the subjective scores. Orduna et al. [158] focused on evaluating the correlation between VMAF and the subjective quality of watching 360° videos. Their results indicated that VMAF has quite high correlation with the subjective quality compared to other objective quality metrics, including S-PSNR and V-PSNR.

2.4.2 QoE Factors

Several studies [48, 57, 185, 205] have investigated the key factors, such as bitrate, resolution, and video characteristics, that influenced the user experience. Fernandes and Feiner [57] implemented a testbed that dynamically varies the viewport size in the HMD during the subjective test. The subjects were asked to traverse through a set of way-points and to periodically graded the cybersickness level. Their results show that restricting the viewport size reduced the cybersickness level, unless the viewport size was extremely small. Singla et al. [185] studied the cybersickness level using different HMDs: Oculus DK2 and HTC Vive. They recruited 28 subjects, each of whom rated 24 videos with 6 different types of contents, 2 resolutions, and 2 bitrates. Their results showed that HTC Vive leads to slightly better subjective visual quality compared to Oculus DK2. On the other hand, both content and resolution imposed higher impacts on user experience than the HMD model. Bio sensors can also be used to predict the subjective quality in VR applications. Egan et al. [48] captured the heart rate and electrodermal activity of the subjects when they were watching 360° videos. The correlations of these two metrics and the subjective scores were analyzed and discussed. Their results revealed that HMDs offered better subjective quality levels than 2D displays.

Some other studies have investigated the key factors of particular VR applications. Schatz et al. [180] considered VR-based training applications. They studied how the rendering styles and scene types affect the subjective scores and subject performance. Vlahovic et al. [205] studied the impact of locomotion, such as first-person, teleportation, and gesture-based, in VR applications. They found that the controller-based locomotion resulted in higher cybersickness levels. In contrast, the teleportation one resulted in the lowest average cybersickness level and the highest overall QoE. They also pointed out that reducing the cybersickness level improves the subjective visual quality. Singla et al. [186] conducted a user study on tiled 360° video streaming. They studied the impacts from various factors, such as bandwidth, delay, and resolution. They have also measured the perceived quality and cybersickness level under different latency types, such as the tile switching delay and network delay. Their results showed that 47 ms is the acceptable network delay that does not degrade the quality ratings. These studies carried out experiments to identify the factors of subjective quality metrics on 360° videos [48, 57, 185, 186, 205].

2.4.3 QoE Modeling

Several studies have developed QoE models for conventional videos. Song et al. [189] used piece-wise linear regression models for predicting the subjective scores of viewing H.264, H.265, and VP9 encoded videos. Learning-based modeling has also been studied [10]. Some QoE modeling studies for 360° images or videos have been carried out. Huang et al. [79] built a QoE model for 360° images. They used the resolution and compression quality extracted from the Joint Photographic Experts Group (JPEG) to predict the overall QoE. Hsu et al. [74] developed QoE models of watching 360° videos using a 2D monitor with foveated rendering support, which encoded the foveal region at a higher resolution. They pre-defined (fixed) the viewing path of the 360° videos based on the viewers' gaze paths in the pilot tests. Xie et al. [218] built a model for the perceptual impact of the speed of a viewport's quality updates upon a sudden orientation change. Yao et al. [225] developed a 360° video player supporting various 360° video projection schemes. They used the testbed to collect the MOS scores and built a QoE model with diverse Quantization Parameter (QP), projection schemes, and video characteristics. Croci et al. [41] split each 360° video into multiple patches, which have evenly distributed pixels on the sphere. The objective quality metrics, e.g., PSNR, Structural Similarity Index (SSIM) [29], or VMAF [142], have been calculated for these patches, which were then averaged across all patches. The resulting quality metrics are referred to as VI-PSNR, VI-SSIM, and VI-VMAF, respectively. Their user study showed that VI-VMAF had the highest correlation with the MOS scores of overall QoE. They further extended their metrics into weighted variants [42], where the weights were Visual Attention (VA) maps [93] and the resulting metrics were referred to as VI-VA-metrics, such as VI-VA-PSNR and VI-VA-VMAF. Their results suggested that VI-VA-VMAF and VI-VA-MS-SSIM are the two quality metrics with the highest correlation with the overall QoE. Anwar et al. [8] leveraged Logistic regression to model the overall QoE with content and human factors. Different from most other studies (including this thesis), they formulated the QoE modeling problem as a classification problem with only two classes: *satisfied* and *unsatisfied*, which may be too coarse-grained. Therefore, their work cannot be compared with ours and other continuous QoE models [41, 42, 79, 218, 225].

In addition, several studies have built QoE models for cybersickness [101, 103, 112, 162, 169]. For example, Kim et al. [103] used 360° scenes generated by a local Unity engine to study how different factors affect cybersickness. Several other studies [101, 112, 162] have leveraged neural networks for modeling the perceived cybersickness on 360° videos. Furthermore, Raake et al. [169] modeled the perceived cybersickness of each viewing session considering the elapsed time.

Table 2.6: A Comparisons among Ours and Existing Models

QoE Models	**Overall QoE**	**QoE Feature**				**QoE Factor**			**Model Type**	
		IQ	**FG**	**IM**	**CS**	**Content**	**Human**	**Context**	**MOS**	**IS**
This Work	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Yao et al. [225]	✓					✓			✓	
Croci et al. [41]	✓					✓			✓	
Croci et al. [42]	✓					✓		✓	✓	

In this thesis, we build complete QoE models for overall QoE and QoE features of watching 360° videos, and further investigate the dominating factor categories and factors. Moreover, we build the models for both MOS and IS. In contrast, most existing modeling studies focused on 360° still images [79], cybersickness [101, 103, 112, 162, 169], binary QoE classification [8], and QoE impacts of viewport adaptation [218], which cannot be compared with our models. Yao et al. [225], Croci et al. [41], and Croci et al. [42] are probably the most relevant studies to ours. Table 2.6 summarizes the comparisons among all the QoE models. In summary, our work has a much wider scope in at least three dimensions: (i) 30 QoE factors from 3 categories, (ii) overall QoE and 5 QoE features,and (iii) MOS versus IS models. Other relevant studies [41, 42, 225], however, only consider a few factors in the first dimension and largely ignore the second and third dimensions. We use these three QoE models as the baselines when evaluating our overall QoE model in Sec. 5.3.

Chapter 3

Delivery Optimization: Fixation Prediction

In 360° videos, HMD viewers only get to see a small viewable region, called *the viewport*, of each 360° video at any moment. Therefore, streaming whole 360° videos wastes precious bandwidth on many unwatched regions. A better solution is to predict *viewer fixation*, which can be quantified by the viewing probability of different regions, and only transmit the regions with high viewing probability. In particular, we propose to use a RNN that considers both sensor and content features to predict the viewer fixation, where the sensor features are extracted from the HMDs and the content features are detected from the video content via CV algorithms.

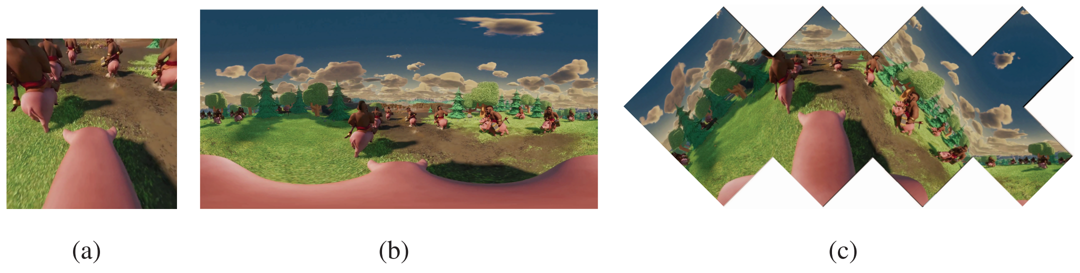


Figure 3.1: A sample image: (a) seen in HMDs; distorted images due to: (b) equirectangular projection and (c) rhombic dodecahedron projection.

However, the content features, such as saliency maps [20] and motion maps [98], are outputs of CV algorithms, they are vulnerable to distortion attributed to projection models. Such distortion can be further classified into: (i) shape distortion and (ii) poor segmentation, which are illustrated in Fig. 3.1. Compared to the image seen in an HMD (Fig. 3.1(a)), the image from equirectangular projection (Fig. 3.1(b)) suffers from shape distortion, which is especially severe for objects close to the north and south poles. On the other hand, the image from rhombic dodecahedron projection (Fig. 3.1(c)) suffers from poor segmentation, where an object is cut into smaller pieces in different parts of the projected image. One solution approach is to adopt CV algorithms specifically designed for 360° videos in a given (say equirectangular) projection model. However, such CV algorithms would not work for 360° videos in other projection models. Moreover, compared to CV algorithms designed for 2D images/videos, there are only very few CV algorithms proposed and trained with 360° videos [11, 107, 119, 136]. While the number of these CV algorithms may increase over time, they will still be outnumbered by the CV algorithms for 2D images/videos. To overcome such limitations, we systematically generate *virtual viewports* ahead of the streaming time. Virtual viewports are *carefully* chosen *simulated* viewports in HMDs, which are projected back to the *sphere*. Therefore virtual viewports are not vulnerable to distortion caused by projection models. By sending virtual viewports to existing CV algorithms, we improve the quality of their outputs, as well as that of the subsequent fixation network.

3.1 Overview

We present an overview of the 360° video streaming systems, which is optimized in the rest of this chapter.

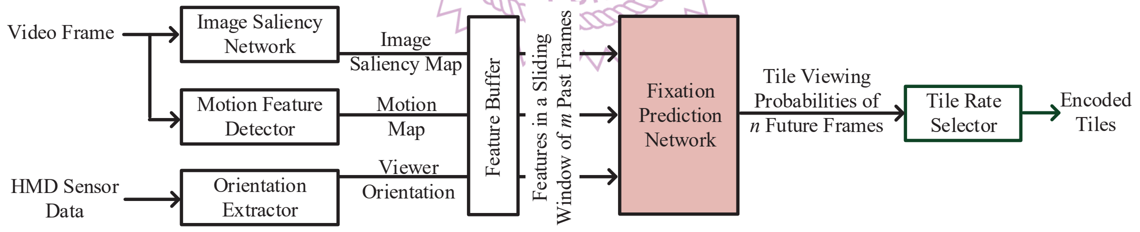


Figure 3.2: Architecture of the proposed 360° video streaming server. A tile-based streaming example is shown.

3.1.1 360° Video Streaming Systems

Fig. 3.2 presents our proposed architecture of a 360° streaming server [50], in which we focus on the software components related to *fixation prediction*. We have identified two content features: image saliency map [20] and motion map [98] from 360° videos; and a sensor feature: orientation from HMDs. We describe the software components in the following:

- **Image saliency network** is a deep neural network trained to derive the *image saliency map*, which shows the parts of the image that attract viewers the most.
- **Motion feature detector** analyzes the Lucas-Kanade optical flow [126] of consecutive frames, because viewers may be attracted by moving objects.
- **Orientation extractor** derives the viewer orientation data including yaw, pitch, and roll, from HMD sensors.
- **Feature buffer** stores the features, including the saliency map, motion map, and viewer orientation in a sliding window, which are used for fixation prediction.
- **Fixation prediction network** uses content features (image saliency maps and motion maps) and sensor features (viewer orientation) as inputs to predict the viewing probability of different regions of the next n frames.
- **Tile rate selector** performs rate allocation among video *tiles*, which are rectangular and independently decodable regions of a video frame. 360° video streaming systems may be classified into two classes: tile-based [43, 111, 144, 207] and transcoder-based [9, 206]. In tile-based systems, the server encodes 360° videos into tiles, while the client dynamically requests tiles at specific bitrates for adaptation. In transcoder-based systems, the server dynamically transcodes the viewport of each viewer on-the-fly. For the sake of brevity, we assume that tile-based systems are used, although our solution is also applicable to transcoder-based systems.

The interactions among these components are as follows. The video frames are sent to the image saliency network and motion feature detector for generating the image saliency map and the motion map, respectively. Generating these two maps is potentially resource demanding, and we assume that they are created offline for pre-recorded videos. The HMD sensor data are transmitted to the orientation extractor to derive the viewer orientation. The feature buffer maintains a sliding window that stores the latest image saliency maps, motion maps, and viewer orientations as the inputs of the fixation prediction network. The fixation prediction network predicts the future viewing probability of each tile. The tile rate selector optimally selects the rates of the encoded video tiles.

3.1.2 Viewport and Modeling

Different HMDs may have different viewport sizes, which need to be systematically derived. We conduct experiments of playing a 360° video with artificial *grids* to viewers, and collect questionnaires to understand how to model the viewport of commodity HMDs. We find that the viewport of existing HMDs, including Oculus Rift, HTC Vive, and Samsung Gear can all be modeled as a circle on sphere. Viewports could be in different shapes on 2D projected planes. For example, a viewport appears as an ellipse on an equirectangular

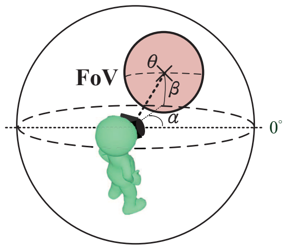


Figure 3.3: The model of HMD viewport.

projected plane. Fig. 3.3 presents the viewport model of HMDs. The viewer stands at the center of the sphere. Let α and β be the yaw and pitch of the HMD viewport center, which are reported from the sensors equipped by HMDs. Furthermore, we let θ be the diameter of a viewport in degrees. Therefore, we describe the viewport in the spherical space as $f_s = (\alpha, \beta, \theta)$. The measured θ values are about 100° (Oculus Rift), 67° (HTC Vive), and 67° (Samsung Gear). We use 100° in our experiments if not otherwise specified. We note that the parameters of other HMDs may be derived using our experiment design.

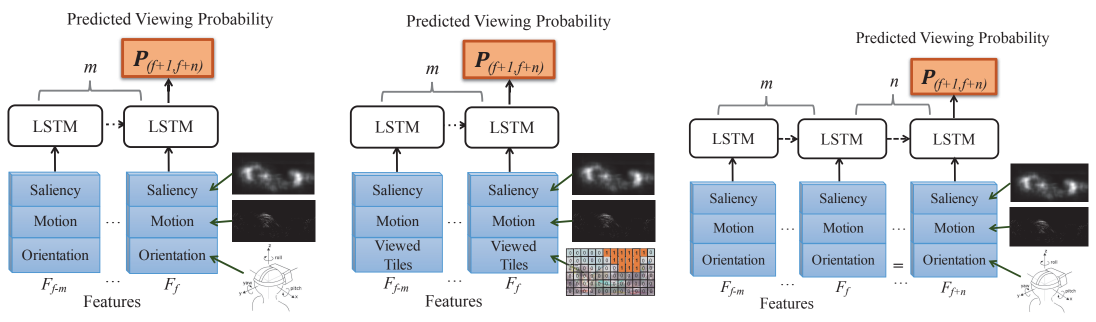
3.2 Fixation Prediction Networks

The core component of our proposed 360° streaming server is the fixation prediction network, which is detailed in this section.

The fixation prediction network is based on an RNN, which is suitable to learn useful information from a time series of video frames. However, basic RNNs suffer from the problem of gradient vanishing during back-propagation [133]. This prevents the RNN from learning long-term dependencies effectively. Hence, we chose to use the LSTM (Long Short Term Memory) network [70]. LSTM solves the problem by using gates in its neurons, and learns more long-term dependencies among video frames.

In this chapter, we propose three neural networks: (i) orientation-based, (ii) tile-based, and (iii) future-aware. The orientation-based network takes the orientation values of the past frames, which are read from HMD sensors, as the sensor features. The tile-based network considers the viewing probabilities of tiles, which have already been projected from the raw orientation values, of the past frames as the sensor features. Both networks take the saliency and motion maps of the past frames as the content features. Our preliminary study [50] demonstrates the higher prediction accuracy and efficiency of the orientation-based network compared to the tile-based network. Therefore, we extend the orientation-based network into the future-aware network in this chapter. The future-aware network considers the content features of not only the past frames but also future frames. This is feasible because all the video frames are pre-stored on the server; thus, the content features can be extracted and saved beforehand.

3.2.1 Overview



(a) (b) (c)

Figure 3.4: Our proposed fixation prediction networks: (a) orientation-based network, (b) tile-based network, and (c) future-aware network.

In addition to the future-aware network, we enhance the networks presented in our previous study [50] in two ways. First, we reduce the feature sampling rate to 1 frame-per-second (fps). The intuition behind this decision is that the changes of video content and viewer orientation are typically small over a short time period. Therefore, although lower sampling rates may impose small negative impacts on prediction accuracy, they significantly reduce the resource consumption. Second, the proposed networks predict the viewing probability of each tile within a number of frames instead of just a single frame. The rationale is that in most practical streaming systems, each client asks for a few consecutive frames in a request. For example, a DASH client asks for a *segment* of video frames in each request. Our pilot experiments show that the two enhancements lead to: (i) at least three times of training time reduction and (ii) on average 1.4% accuracy boost, compared to our original networks [50]. We present the three resulting networks below, while more performance results are given in Sec. 4.6.

3.2.2 Orientation-Based Network

Fig. 3.4(a) presents the orientation-based network. Let F_f be the features of frame f, which include the image saliency map, motion map, and viewer orientation. The saliency maps and motion maps are downsampled to 64x64 to avoid excessive computation loads. The viewer orientation is the sensor data, which consists of *x, y, z, yaw, roll*, and *pitch*, read from HMD sensors. These features are concatenated and fed into the network. Let m and n be the number of past frame samples that contribute the features to the network and the number of the predicted frames, respectively. We let $P^t_{f+1,f+n}$ denote the predicted viewing probability of tile t within frame samples $f+1$ to $f+n$. That is, if the tile is predicted to be viewed for x times within these frames, the predicted viewing probability of this tile is $\frac{x}{n}$. We collectively write the probabilities of all tiles within frame samples $f+1$ to $f+n$ as $\mathbf{P}_{f+1,f+n}$.

3.2.3 Tile-Based Network

The tile-based network is presented in Fig. 3.4(b). Compared to the orientation-based network, the tile-based network replaces the viewer orientation with the viewing probability of each tile. More specifically, the probabilities of tiles that are viewed by the viewer are 1's and the others are 0's. Similar to the orientation-based network, the saliency maps, motion maps, and the viewing probability of tiles from past m frames are concatenated as features and fed into the network to predict the viewing probability of tiles within the next n frames.

3.2.4 Future-Aware Network

Fig. 3.4(c) presents the proposed future-aware network. It is extended from the orientation-based network due to its better performance compared to the tile-based network [50]. In particular, it also takes the future content features into account. That is, the future-aware network takes the features from F_{f-m} to F_{f+n} as inputs to predict the viewing probabilities $\mathbf{P_{f+1,f+n}}$. The unknown future viewer orientation for F_{f+1} to F_{f+n} is approximated with the last received viewer orientation, if not otherwise specified, while more sophisticated extrapolation [168] can also be used.

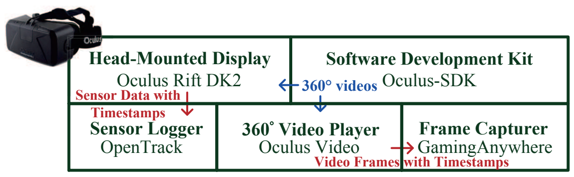
3.3 Datasets and Network Implementations

We collected a 360° video dataset, which contains 50 viewers, each of whom watched ten 360° videos [123]. In this section, we first summarize the dataset. We next compare the performance of the proposed fixation prediction networks using the collected dataset.

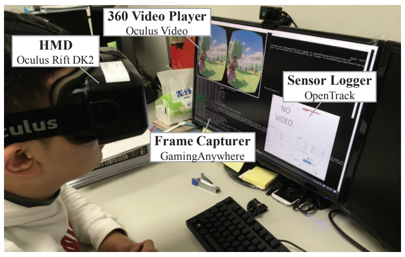
3.3.1 Dataset

Fig. 3.5(a) presents the design of our testbed [50], which consists of: (i) an Oculus Rift HMD, (ii) the Oculus Software Development Kit (SDK), (iii) the 360° video player (rendering 360° videos in HMD and on a mirrored screen), (iv) the sensor logger based on OpenTrack, and (v) the frame capturer based on GamingAnywhere [78].

When a viewer watches a 360° video as shown in Fig. 3.5(b), the rendered video is captured by the frame capturer and stored to the disk. The viewer's head movements,



(a)



(b)

Figure 3.5: Our testbed for collecting our dataset: (a) testbed architecture and (b) a photo of a subject performing the experiments.

including position and orientation, are recorded by the sensor logger. Both of them are timestamped on the same computer. By aligning sensor data and 360° videos, we know where the viewer is watching at any moment.

We downloaded ten 360° videos from YouTube, which are in 4K resolution with a frame rate of 30 fps. The videos have diverse characteristics, e.g., computer-generated versus natural images, and slow- versus fast-paced. We recruited 50 viewers for dataset collection. We played all ten videos to each viewer, which resulted in 500 *traces* in our dataset. By trace, we refer to a combination of a viewer and a video, in the rest of this chapter. For more details about the compositions of the viewers and the format of the datasets, readers are referred to Lo et al. [123].

3.3.2 Network Implementations

We consider the fixation prediction problem on tiles as a multi-label classification problem and have implemented the neural networks using Scikit-Learn and Keras. The ground truth of the fixation prediction networks for each tile is the fraction of frames containing the viewed tiles. This fraction represents the *importance* of each tile. Using the datasets, we sample the points within the viewport by projecting the orientation on the sphere to the equirectangular model. Then, the viewed tiles are those that contain some projected samples. For a single video frame, each tile is either watched or not, i.e., it has a boolean viewing probability.

We use the traces from 50 viewers to train the proposed three networks. We randomly divide the 500 traces [123] into two subsets: 80% for training and 20% for testing. We reserve 20% of the training set for validation purpose. The networks are trained to minimize the *logarithmic loss*, also known as *cross-entropy loss*, using Stochastic Gradient Descent [21] with a learning rate of 10^{-1}. An early-stop mechanism, which stops the training once the logarithmic loss is smaller than a given value, is adopted to speed up the

Table 3.1: The Performance of the Proposed Models with 1-sec Sliding Window

The Orientation-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*F*	86.93%	0.703	85.79%	0.678
512	*2*	*T*	88.41%	0.741	87.03%	0.711
1024	***2***	***T***	**89.09%**	**0.760**	**87.05%**	**0.732**
2048	*2*	*T*	88.11%	0.733	86.67%	0.702

The Tile-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*T*	84.80%	0.636	83.65%	0.610
512	*2*	*F*	84.68%	0.632	0.147	83.43%
1024	*2*	*F*	84.96%	0.636	83.75%	0.608
2048	***2***	***T***	**85.15%**	**0.643**	**83.90%**	**0.614**

The Future-Aware Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*T*	88.09%	0.733	86.77%	0.706
512	*2*	*T*	88.99%	0.759	87.62%	0.732
1024	***2***	***T***	**89.27%**	**0.767**	**87.77%**	**0.737**
2048	*2*	*F*	85.65%	0.663	84.43%	0.635

network training and avoid over-fitting. We consider the sliding window size of 1 and 4 secs to predict the frames in the upcoming second. To obtain the optimal parameters, we consider the number of neurons in {256, 512, 1024, 2048}, the number of LSTM layers in {1, 2, 3}, and the dropout in {True, False}, where the dropout rate is 0.2.

We note that the predicted probability is a real number between 0 and 1, and we use a *threshold* ρ to round it to a boolean decision. We refer to $p_f^t \geq \rho$ as *predicted tiles*, and the actually viewed tiles as *viewed tiles*. We let $\rho = 0.5$ if not otherwise specified. To select the optimal parameters of the three neural networks, we consider two metrics: (i) *accuracy*, which is the ratio of correctly classified tiles to the union of predicted and viewed tiles and (ii) *F-score*, which is the harmonic mean of the precision and recall, where the precision and recall are the ratios of correctly predicted tiles to the predicted and viewed tiles, respectively.

We find that the networks with two LSTM layers generally give better performance, and thus we report sample 2-layer results with 1- and 4-sec sliding windows in Tables 3.1 and 3.2, respectively. The optimal parameters and results for each network are in bold

Table 3.2: The Performance of the Proposed Models with 4-sec Sliding Window

The Orientation-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*F*	86.37%	0.615	83.27%	0.588
512	***2***	***F***	**87.91%**	**0.729**	**86.43%**	**0.699**
1024	*2*	*F*	86.90%	0.696	85.27%	0.658
2048	*2*	*F*	84.73%	0.634	83.61%	0.606

The Tile-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*T*	84.62%	0.629	83.49%	0.604
512	*2*	*F*	84.69%	0.624	83.44%	0.593
1024	***2***	***T***	**85.00%**	**0.634**	**83.73%**	**0.603**
2048	*2*	*F*	84.57%	0.623	83.38%	0.594

The Future-Aware Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*F*	84.44%	0.612	83.45%	0.589
512	***2***	***T***	**88.15%**	**0.733**	**86.70%**	**0.701**
1024	*2*	*F*	85.27%	0.648	84.18%	0.624
2048	*2*	*F*	84.80%	0.636	83.71%	0.610

fonts. These tables show that the future-aware network has higher accuracy and F-score for both sliding window sizes. Besides, with the future-aware network, the 1-sec sliding window performs slightly better than the 4-sec one (accuracy 87.77% > 86.70%), yet runs 6x faster (69 versus 398 minutes). *Hence, we adopt the future-aware network with the 1-sec sliding window as our fixation prediction network in the rest of the chapter.*

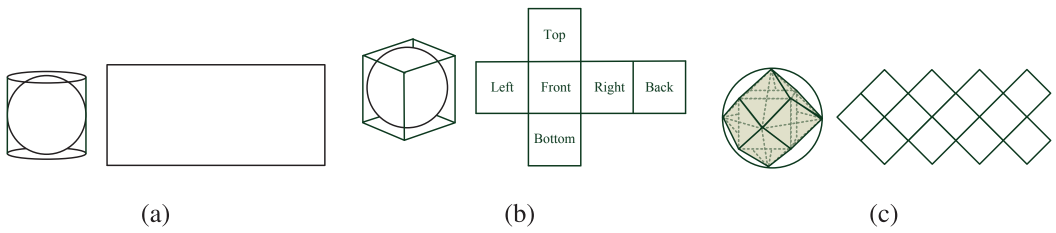


Figure 3.6: Several projection models can be used for 360° videos, such as: (a) equirectangular, (b) cubic, and (c) rhombic dodecahedron.

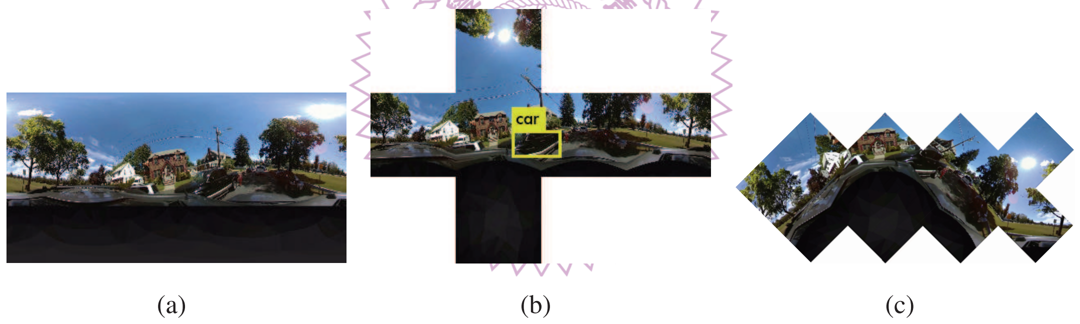


Figure 3.7: Object detection on the sample image with different projection models: (a) equirectangular, (b) cubic, and (c) rhombic dodecahedron. Only one object is detected.

3.4 Overlapping Virtual Viewports

In this section, we first introduce the projection models and discuss how they negatively affect the performance of CV algorithms designed and trained for 2D images/videos. We then propose the overlapping virtual viewport (OVV). We apply OVVs on three existing CV algorithms and report their performance boosts. We then apply OVV to our proposed fixation prediction network. Last, to validate the generality of our solution, we evaluate it using additional videos and viewers.

3.4.1 Projection Models

There are many projection models proposed for 360° videos [37,49] in the literature, and the commonly seen ones[1] are: (i) equirectangular, (ii) cubic, and (iii) rhombic dodecahedron. We present the properties of individual projection models in the following.

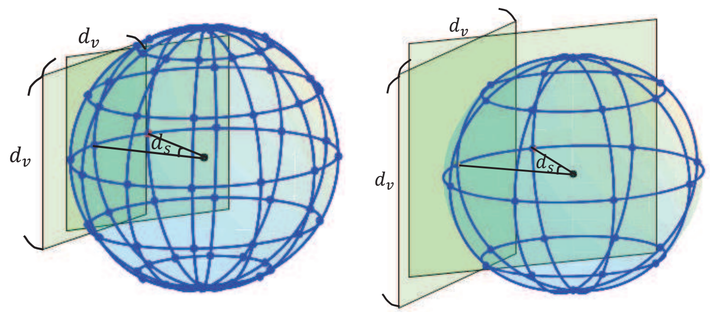
Equirectangular. As shown in Fig. 3.6(a), the equirectangular projection projects the sphere to a *cylinder*. It introduces large shape distortion at the areas close to poles (see Fig. 3.1(b)), which may result in redundant data transmission and inferior performance (for example, accuracy) of the existing CV algorithms.

Cubic. The cubic projection model projects a sphere to a circumscribed cube with six square faces as shown in Fig. 3.6(b). For each point on the sphere, we first find the closest face as its corresponding face. Each face adopts 90° rectilinear projection, which maps the sphere surface to a tangent plane, where the points are projected along with the line from the sphere center to the plane. In contrast to equirectangular, the cubic projection model preserves the straight lines on each face. Therefore, the cubic projection model results in no pole distortion, and reduces about 25% of the data size [109]. However, for lines or objects that span multiple faces, they are unnaturally, or poorly, segmented at the face boundaries.

Rhombic dodecahedron. The rhombic dodecahedron projection model adopts 12 equal-size spherical rhombuses. Fig. 3.6(c) shows the construction of the rhombic dodecahedron, which is an octahedron with a cube embedded in it. Two of the four corners of each rhombus are from the cube, while the other two are from the octahedron. We can project the rhombic dodecahedron to a sphere surface using gnomonic projection, which is a superset of rectilinear projection that does not limit the degree to 90°. One way to project pixels to the rhombic dodecahedron is to used the great circle subdivisions [61]. While the rhombic dodecahedron projection model does not suffer from noticeable distortion, it incurs higher computational overhead.

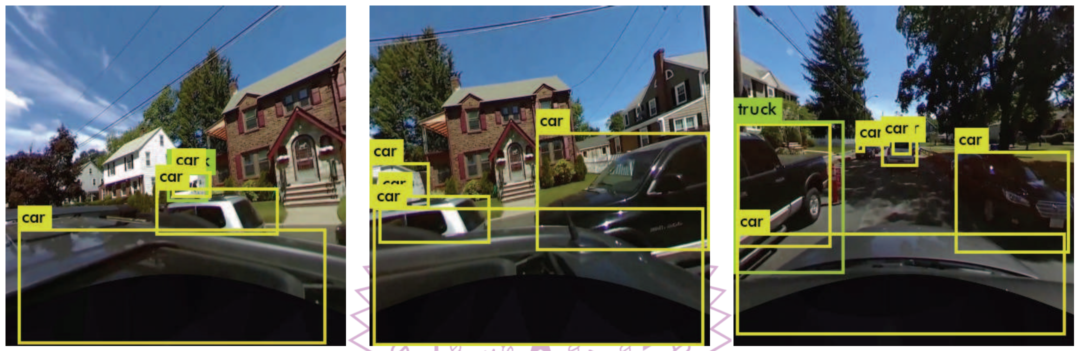
For the sake of understanding the distortion level, we use YOLO9000 [173] to perform object detection on different projection models. Fig. 3.7 shows sample object detection results with: (i) equirectangular, (ii) cubic, and (iii) rhombic dodecahedron projection models. This figure shows that only a single object is detected. This can be attributed to shape distortion and poor segmentation.

[1]Note that researchers and companies continue proposing new projection models for better coding efficiency. Most projection models suffer from some shape distortion and/or ill-segmentation, similar to the representative models presented here.



(a) (b)

Figure 3.8: Examples of the OVV: (a) $(d_s, d_v) = (30°, 60°)$ and (b) $(d_s, d_v) = (45°, 90°)$.



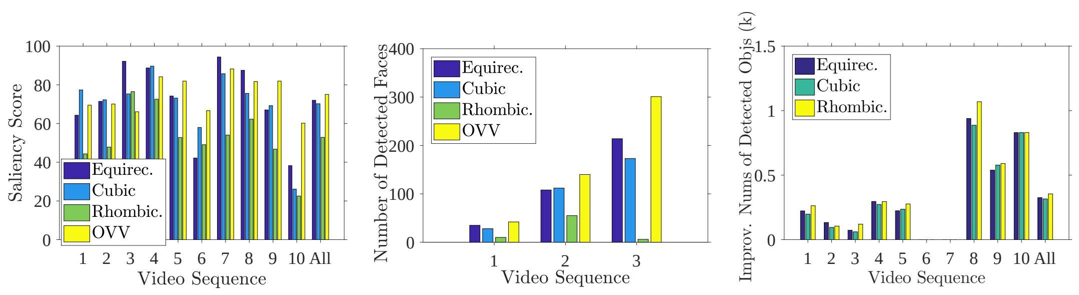
(a) (b) (c)

Figure 3.9: Object detection on sample virtual viewports of OVV: (a) (yaw, pitch) = (315°, 90°), (b) (yaw, pitch) = (0°, 90°), and (c) (yaw, pitch) = (90°, 90°). More objects are detected with OVV, compared to Fig. 3.7.

3.4.2 Overlapping Virtual Viewport (OVV)

We propose to leverage OVV to cover the whole sphere space so as to turn CV algorithms designed and trained for 2D images/videos applicable to 360° videos. A virtual viewport is a square tangent to a point on the sphere surface. OVV is defined by d_v and d_s, where d_v represents the viewable angle at the equator of each virtual viewport, and d_s is the sampling angle of virtual viewports. Both d_v (size) and d_s (density) are in the unit of degrees. Fig. 3.8 illustrates example OVVs with $(d_s, d_v) = (30°, 60°)$ and $(d_s, d_v) = (45°, 90°)$. In the figure, we only plot two sample virtual viewports for brevity. However, each intersection point on the sphere surface is the center of a virtual viewport, which is tangential to the sphere at that point. Therefore, each 360° sphere has $\frac{2\pi}{d_s}\frac{\pi}{d_s}$ virtual viewports.

OVV eliminates the shape distortion and poor segmentation by: (i) extracting virtual



(a) (b) (c)

Figure 3.10: Our proposed OVV improves the performance of the pre-trained CV algorithms: (i) saliency detection, (ii) face detection, and (iii) object detection.

viewports, which are the actual views seen in HMDs, and (ii) oversampling (overlapping) virtual viewports to increase the chance for CV algorithms to identify, for example, objects. Fig. 3.9 shows the sample results of detecting objects with OVV, which gives more recognized objects than the original approach reported in Fig. 3.7.

3.4.3 Validations with Real Computer Vision Algorithms

We consider three representative CV algorithms: (i) saliency detection, (ii) face detection, and (iii) object detection. In the following, we describe these algorithms and report the benefits of applying OVV with (d_s,d_v)=(45°,90°) on them. The ten videos used for validation are from the dataset mentioned above [123], and the ground truth is tagged by our group.

Saliency detection [20] calculates the attraction level of each image pixel. This can be done based on analyzing image contrast, detecting objects, and locating outstanding and meaningful parts of images. We adopt a pre-trained deep multi-level network [40] that takes low- to high-level features to perform saliency detection with diverse projection models. To compare the performance of the saliency detecting algorithm on different projection models, we use *true* viewports from our traces to quantify the quality of the resulting saliency maps. For each projection model, we normalize the detected saliency map so that the sum of the saliency values is 1. Then we sum all the detected saliency values within the *actual* viewport at each frame of each projection model, and refer to it as the *saliency scores*. For OVV, we perform saliency detection on individual virtual viewports and normalize the overlapped regions among virtual viewports for fair comparisons. Fig. 3.10(a) plots the total saliency scores with 95% confidence intervals under different projection models. This figure shows that OVV generally has higher saliency scores than other projection models. Besides, the p-value from the one-way analysis of variance (ANOVA) test is only 0.012 (≤ 0.05), which shows the statistic significance of the performance difference.

Face detection [69] overlays rectangles that contain human faces on images. This can be done by first extracting features and then performing cascades detection, which is to check the features with classifiers stage-by-stage. Once the checking process fails at a stage, it terminates. A face is detected only if all stages are passed. We implement face detection based on Haar Cascades classifier using OpenCV [24] with a scaleFactor of 1.3 and a minNeighbors of 5. We perform face detection on 3 (out of 10) videos that contain people. We record the number of correctly detected faces of each projection model and plot the total number of each video in Fig. 3.10(b). This figure shows that OVV detects up to 300 more faces compared to other projection models.

Object detection [64] highlights the regions that contain real-world objects, such as dogs, bicycles, and chairs. We adopt the YOLO9000 network [173], which is trained by ImageNet [46] and COCO [120] datasets, and is able to recognize more than 9000 objects. We apply YOLO9000 on the ten videos from the dataset, and generate the annotated videos with different projection models and OVV. We report the correctly detected results in Fig. 3.10(c). This figure clearly demonstrates that OVV significantly increases the number of correctly detected objects. Note that all projection models detect no objects in videos 6 and 7 since these two videos is of landscapes and 2D games, which have almost no real-world objects.

In sum, the performance of the existing CV algorithms designed and trained for 2D images/videos are improved by our proposed OVV, which is a unified approach to apply them on 360° videos. In the next section, we integrate OVV into our proposed fixation prediction network.

3.4.4 Fixation Prediction with OVV

We perform saliency detection on virtual viewports, and stitch the saliency maps of virtual viewports into one image for each frame. The stitched saliency maps then replace the original equirectangular saliency maps as the inputs of the fixation prediction networks. We consider two different configurations of OVV: (i) (d_s, d_v)=(30°, 60°) and (ii) (d_s, d_v)=(45°, 90°). We then train the proposed neural network using these two OVV setups, and report the sample results in Table 3.3. The optimal parameters and results are in bold font. This table shows that the performance of the future-aware network with OVV is better when (d_s,d_v)=(30°,60°). Moreover, it outperforms the equirectangular projection model in terms of accuracy and F-score reported in Table 3.1. Thus, we adopt OVV with (d_s,d_v)=(30°,60°) as the fixation prediction network in the rest of this chapter.

Table 3.3: The Performance of the Future-Aware Network with OVV

$(d_s, d_v) = (30°, 60°)$						
Model Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*F*	87.31%	0.714	86.03%	0.686
512	*2*	*F*	82.57%	0.602	81.54%	0.582
1024	*2*	*T*	89.22%	0.764	87.71%	0.733
2048	***2***	***T***	**89.72%**	**0.778**	**87.93%**	**0.742**

$(d_s, d_v) = (45°, 90°)$						
Model Parameters		*Training Set*		*Testing Set*		
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	*2*	*F*	86.71%	0.702	85.35%	0.674
512	*2*	*T*	84.83%	0.647	83.59%	0.620
1024	*2*	*T*	88.36%	0.745	86.94%	0.716
2048	***2***	***T***	**88.63%**	**0.753**	**87.09%**	**0.722**

3.4.5 Validation with Additional Videos/Viewers

To understand the generality of our proposed prediction model, we validate our trained model with additional videos and viewers.

Setup. We perform prediction on five new 360° videos downloaded from YouTube at 3840x1920 resolution and 30 fps. We cut them into the same length of 30 seconds. Table 3.4 lists the considered videos. We recruit 30 viewers between 19 and 28 years old. Among them, about 2/3 are males. All viewers are asked to freely watch the five videos in random order. The viewer orientation is logged when they are watching the videos. We consider Ban et al. [12], which is introduced in Sec. 2.4, as our baseline, and call it CUB360[2]. Because CUB360 employs KNN for prediction, we perform 3-fold validations on CUB360, where 10 viewers are used as the testing set, and 20 viewers are used as the KNN inputs. For our prediction network, we use the model derived in the previous section to predict the viewing probabilities of each tile on all the traces from the 30 viewers viewing the five videos. We note that the comparisons give CUB360 a slight edge of peeking into the new videos/viewers.

Results. Table 3.5 reports the performance in accuracy and F-score for our proposed prediction network and CUB360 under different K-nearest viewpoints selection. To be conservative, we grid-search on the thresholds to round the probabilities to Boolean decisions and report the absolutely optimal solution of CUB360 for each K value. This

[2]While we also want to consider Nguyen et al. [145] as another baseline, we couldn't do so because some of their parameters had not yet been made public at the time of writing.

Table 3.4: Validation Videos

Video	Time Interval	Link
Snow Boarding	01:30–02:00	https://goo.gl/2H4RxM
Elephants	00:50–01:20	https://goo.gl/aejCVM
Sharks	00:05–00:35	https://goo.gl/BUEBG9
CERN	01:50–02:20	https://goo.gl/68gDAZ
London	00:40–01:10	https://goo.gl/YRN1kN

Table 3.5: The Performance of Different Prediction Algorithms

Prediction Algorithm		Accuracy	F-Score
Our		81.8%	63.1%
CUB360	K**=0**	73.1%	31.0%
	K**=2**	73.0%	53.4%
	K**=5**	73.0%	54.3%
	K**=10**	72.2%	54.6%

table shows that although CUB360 references the K-nearest viewpoints from other viewers on the same video, our prediction network still achieves at least 8.7% higher accuracy without peeking into these traces.

The results demonstrate the generality of our proposed fixation prediction network. In terms of CUB360, increasing K does not always improve the performance. This is because other viewers' fixation may be misleading, since different viewers may have quite diverse viewing behavior. In contrast, our proposed fixation prediction network is trained with other 360° videos and also takes the current viewer's past orientation into account. This makes our prediction algorithm more robust. In addition to CUB360, Nguyen et al. [145] also propose a head movement prediction network using LSTM, which employs the same network architecture as our preliminary study [50], while introducing three improvements as we detail in Sec. 2.4. While we are not able to compare with their work, applying their enhancements in our systems would likely boost our performance further.

3.5 Evaluations

In this section, we evaluate the performance of our 360° video streaming system with the future-aware network and OVV. We conduct the evaluations through simulations, because: (i) the viewer behavior can be repeated for fair comparisons among different algorithms, and (ii) more traces/subjects can be leveraged at a relatively lower cost.

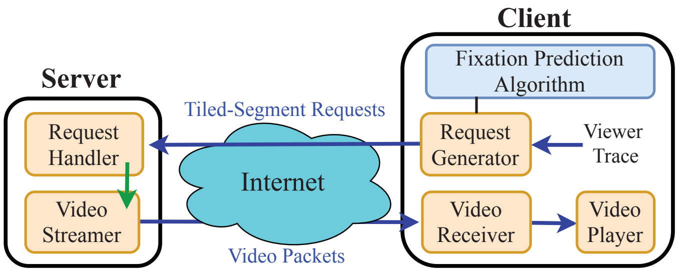


Figure 3.11: The streaming server and client in our simulator.

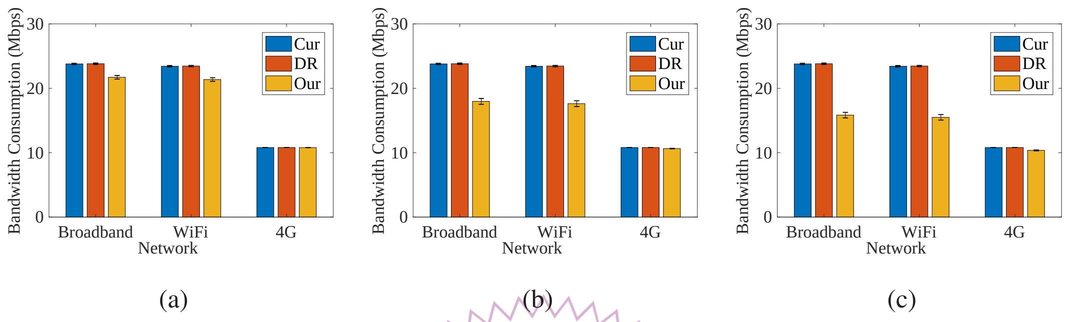


Figure 3.12: The bandwidth consumption in different networks with different target missing ratio: (a) $\tau = 1\%$. (b) $\tau = 5\%$, and (c) $\tau = 10\%$.

3.5.1 Implementations

We have implemented two other prediction algorithms using Python: (i) Cur, which uses the current orientation as the prediction in the next segment and (ii) Dead Reckoning (DR)[3] [168], which computes a weighted moving average of the viewer orientation velocity for prediction. We have implemented a simulator using C++ based on NS-3 [151] and the DASH simulator [152]. We modify the simulator to read the real traces of viewing 360° videos, which contains sizes of the transmitted tiles. We implement our proposed fixation prediction network and the other two baseline algorithms in the fixation prediction algorithm as shown in Fig. 3.11. In addition, there are five more components: (i) the request generator, (ii) the request handler, (iii) the video streamer, (iv) the video receiver, and (v) the video player. The request generator reads the viewer traces and invokes one of the prediction algorithms to generate requests. The request handler parses the received requests. The video streamer encapsulates the tiles into packets and sends them to the video receiver. After the video player reaches an initial buffering time and receives a sufficient number of segments, the video is played until there is no segment available in the

[3]There are two variants of DR prediction: based on the past velocity or on both past velocity and acceleration. We implement the DR algorithm based on the past velocity in our simulations, following a related work [168].

receiving buffer. If a segment is not received in time, a rebuffering event is logged and the player pauses the video until the next segment is received.

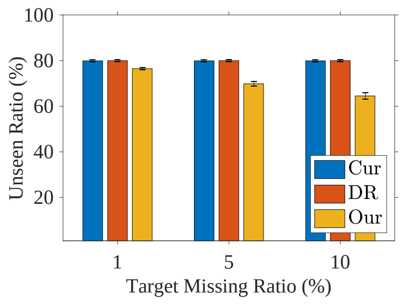


Figure 3.13: The unseen ratio under different target missing ratios.

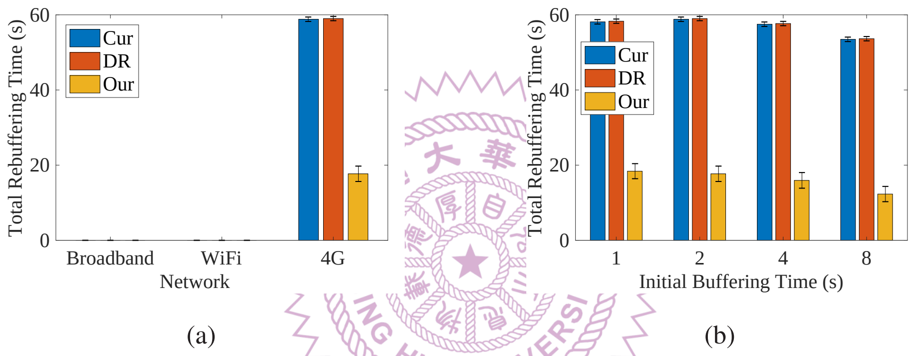


Figure 3.14: The total rebuffering time under: (a) different networks and (ii) different initial buffering time.

3.5.2 Setup

We use all the traces from the testing set (see Sec. 3.3), 98 traces in total, to drive our simulations. We encode these videos into 20x10 tiles, where each tile has 192x192 pixels, with a QP of 28 using Kvazaar [204], and divide them into 1-sec segments using MP4Box. We assume that the network bottleneck is at the client side; therefore, we repeat the simulations with three access networks: (i) (fixed) Broadband, (ii) WiFi, and (iii) 4G cellular network. We consider the average bandwidth (latency) of the above networks in our simulator as 43.2 (3 ms), 37.1 (10 ms), and 12.7 (40 ms) Mbps, respectively, following white papers [34, 157]. To accommodate the latency caused by networks and protocols, we run fixation prediction algorithms a couple of seconds[4] ahead of the current playout time.

We consider the following performance metrics:

[4]Increasing it to 4 seconds or reducing it to 1 second results in similar results.

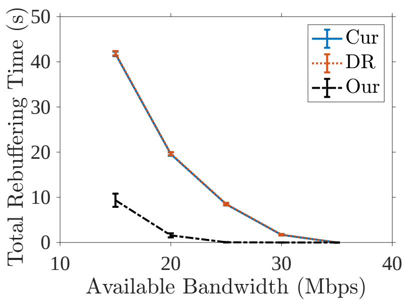


Figure 3.15: The total rebuffering time under different available bandwidths. The curves from Cur and DR overlap.

- **Missing ratio.** The fraction of unavailable tiles at the client over all tiles that are watched by the viewer. A higher missing ratio leads to more *holes* (missing tiles) in the 360° videos.

- **Unseen ratio.** The fraction of the tiles at the client that are not watched by the viewer over all transmitted tiles. Higher unseen ratio indicates more wasted network resources.

- **Bandwidth consumption.** The consumed bandwidth used to stream the predicted tiles.

- **Peak bandwidth.** The peak bandwidth consumption due to streaming the predicted tiles.

- **Video quality.** We employ the objective quality metric V-PSNR, which is proposed for 360° videos [227] and adopted by JVET [84]. V-PSNR is essentially the PSNR value of a viewer's viewport. We use the ground truth of viewports in the datasets to calculate V-PSNR values.

- **Total rebuffering time.** The total rebuffering time throughout each 1-min playout.

Some pilot simulations reveal that the missing ratio is non-trivial for our and the baseline solutions: more than 15% missing ratio is observed. To be practical, we augment our solution to ensure sub-τ missing ratio by adjusting ρ, where *target missing ratio*: $\tau \in \{1\%, 5\%, 10\%\}$. The default τ is 10% if not otherwise specified. For Cur and DR, we iteratively add new tiles at the edge of predicted tiles for δ_{Cur} and δ_{DR} times to accommodate the inferior missing ratio, respectively. Besides, the missed tiles are assumed to be concealed by replaying the last received tiles during video playback. In the next section, we report the simulation results with 95% confidence intervals whenever applicable.

Table 3.6: Video Quality under Different Prediction Algorithms (V-PSNR in dB)

Missing Ratio	1%			5%			10%		
Prediction Algorithm	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
Cur	38.62	42.03	47.58	38.63	42.03	47.58	38.62	42.03	47.58
DR	38.70	42.05	47.64	38.70	42.05	47.64	38.70	42.05	47.64
Our	38.39	42.08	47.89	37.24	41.63	47.27	36.11	41.11	46.66

Table 3.7: Consumed Bandwidth under Different Prediction Algorithms (Mbps)

Missing Ratio	1%			5%			10%		
Prediction Algorithm	Min.	Avg.	Max.	Min.	Avg.	Max.	Min.	Avg.	Max.
Cur	21.11	23.99	24.42	21.11	23.99	24.42	21.11	23.99	24.42
DR	21.13	24.02	24.23	21.13	24.02	24.23	21.13	24.02	24.23
Our	17.13	21.88	24.10	14.08	18.04	22.23	12.26	15.85	20.48

3.5.3 Results

Our fixation prediction network consumes less network bandwidth at any target missing ratio. To meet $\tau \in \{1\%, 5\%, 10\%\}$, δ_{Cur} and δ_{DR} are set to 1, which leads to $\rho = \{0.008, 0.027, 0.053\}$, respectively. We plot the bandwidth consumption under different networks with different τ values in Fig. 3.12. In this figure, Cur and DR show high bandwidth consumption. On the other hand, with properly selected ρ, our fixation prediction network can reduce about 2, 6, and 8 Mbps in bandwidth consumption compared to Cur and DR in Broadband and WiFi networks. Note that the available bandwidth of the 4G cellular network is too limited, and is used up no matter which algorithm is adopted.

Fig. 3.13 plots the unseen ratio under different target missing ratios τ. This figure shows that our fixation prediction network reduces the unseen ratio by 5%. The reduction becomes 10% and 15% when $\tau = 5\%$ and $\tau = 10\%$, respectively. In summary, Figs. 3.12 and 3.13 show that our fixation prediction network consumes less bandwidth due to fewer unseen tiles.

Our fixation prediction network leads to shorter rebuffering time. We plot the total rebuffering time under different networks in Fig. 3.14(a). This figure shows that all considered prediction algorithms lead to no rebuffering event in the Broadband and WiFi networks. However, the rebuffering events occur in all the considered prediction algorithms under bandwidth-limited 4G cellular networks. It is worth noting that our fixation prediction network has shorter rebuffering time than other algorithms in 4G cellular networks: up to 40 second reduction. Fig. 3.14(b) presents the total rebuffering time under different initial buffering time in 4G networks. This figure shows that the rebuffering time is reduced as the initial buffering time increases. However, the total rebuffering time

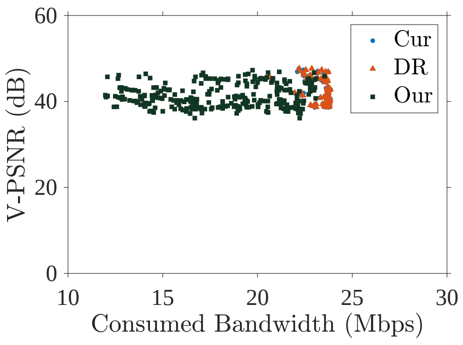


Figure 3.16: The relation between video quality and bandwidth consumption observed under individual traces in WiFi networks.

is still non-negligible to the 60-sec streaming session. This indicates the importance of *rate adaptation* for streaming systems. In our future work, we may choose lower bitrates for the selected tiles that have lower predicted viewing probability to further reduce the bandwidth consumption.

Our fixation prediction network requires less available bandwidth to avoid rebuffering events. To understand the minimum available bandwidth required to avoid rebuffering events without rate adaptation, we conduct simulations under different available bandwidths and plot the results in Fig. 3.15. This figure shows that our fixation prediction network gets rid of rebuffering events while the available bandwidth is higher than 25 Mbps, which is approximately 10 Mbps less than other algorithms. We note that a survey [157] reports that the mean available bandwidth of 4G cellular networks is less than 15 Mbps in North America. Hence, we do not consider 4G cellular networks in the rest of this chapter. The limitation of the current 4G cellular networks are likely to be lifted in the future, as 4G/5G cellular networks continue to advance.

Our fixation prediction network achieves comparable video quality at lower bandwidth consumption. We dig a bit deeper and report the minimum, average, and maximum of video quality of the considered prediction algorithms in Table 3.6. This table shows that our fixation prediction network has comparable video quality compared to other algorithms. The minimum, average, and maximum of the consumed bandwidth are given in Table 3.7, which shows that our algorithm consumes less bandwidth. Combining these two tables, we observe that, with $\tau = 1\%$, our prediction network saves about 9% of the bandwidth, while achieving similar video quality. In addition, it reduces about 8 Mbps in bandwidth consumption while only sacrificing less than 1 dB video quality on average when $\tau = 10\%$. Fig. 3.16 plots the video quality of individual traces from all viewers in WiFi networks. This figure shows that our solution leads to the points located at the left to the points from other solutions, which confirms the above observations.

Table 3.8: The Training Time in Minutes

No. Neurons		**256**		**512**		**1024**		**2048**	
Dropout		**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**
No. Layers	**1**	105	109	78	108	120	125	67	103
	2	47	50	95	59	50	71	89	83
	3	83	105	47	77	119	65	123	69

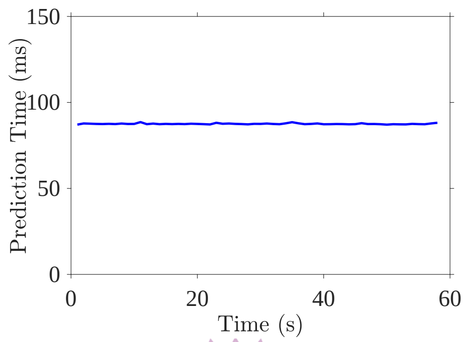


Figure 3.17: The prediction time from a sample trace.

Our fixation prediction network has short training and prediction time. We run the training and prediction process on an Intel 32-core Xeon server with a Nvidia 1080Ti GPU. We report the training time of each parameter setting in Table 3.8. We note that some training sessions take less than 1 hour. This may be attributed to the early stop (see Sec. 3.3) adopted by the training process. Fig. 3.17 plots the prediction time from a sample trace. This figure shows that the prediction time of the next segment is always less than 90 ms for this trace. The average and maximum running time for each prediction across all testing traces are 88.74 ms and 124 ms, respectively, which are relatively short compared to, for example, 2-sec segments.

Table 3.9: The MOS and Consumed Bandwidth from Three Sample Traces

Trace	**MOS**			**Bandwidth (Mbps)**		
	Cur	**DR**	**Our**	**Cur**	**DR**	**Our**
Roller Coaster	3.14	2.86	2.86	24.35	24.33	15.32
Hog Rider	3.43	3.43	3.43	24.18	24.21	13.32
SFR Sport	3.14	3.00	3.29	24.19	24.25	13.71
Average	*3.24*	*3.10*	*3.20*	*24.24*	*24.26*	*14.12*

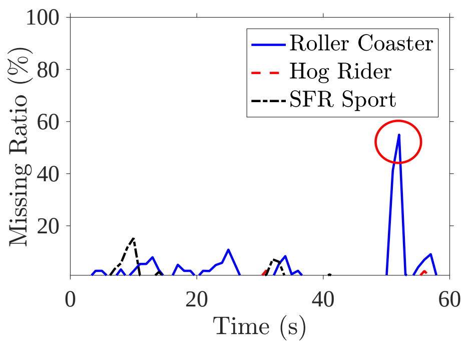


Figure 3.18: The missing ratio of our prediction algorithm on the considered random traces. Roller Coaster, in general, suffers from higher missing ratio.

3.5.4 A Small-Scale User Study

We conduct a user study to understand the correlation between V-PSNR and user experience. We randomly select three sample user traces (one for each video category) from the testing set (see Sec. 3.3). The considered videos are: (i) Roller Coaster, (ii) Hog Rider, and (iii) SFR Sport. We use our prediction algorithm and baseline algorithms to predict the fixations with a missing ratio of $\tau < 10\%$. No error concealment is performed. For each prediction algorithm, we generate viewport videos from the predicted tiles according to the sensor data (yaw, roll, and pitch) from the trace, where the viewports are in 1066×1066 resolution (equivalent to $100° \times 100°$). In total, nine (three traces with three prediction algorithms) viewport videos are generated. We play the viewport videos to seven subjects, who provide user experience scores in the 1 (worst) –– 5 (best) scale. We disable the inertial sensors on the HMDs, so that all subjects watch exactly the same viewport videos following the head movements of the sample user trace. For each pair of the trace and prediction algorithm, we compute the MOS across the user experience scores from all subjects.

We report the MOS and consumed bandwidth in Table 3.9. This table shows that, compared to the baseline algorithms, our solution achieves very similar average MOS for all sample videos, yet saves about 41% bandwidth on average. This table also reveals that our prediction algorithm suffers from an inferior MOS score, with Roller Coaster, than the Cur algorithm. We plot the missing ratio of our algorithm on the considered three traces over time in Fig. 3.18. This figure shows that our algorithm leads to higher missing ratio on Roller Coaster compared to the other two traces. A deeper investigation shows that the higher missing ratio is due to the higher maximum head rotation speed in Roller Coaster trace. More precisely, it is about 35.37 degree/s in yaw direction, which is higher than that of other two traces by up to 14 degree/s. In particular, the maximum rotation speed occurs at 50 s, which is inline with the peak (highlighted with the circle) in Fig. 3.18. We cross check the V-PSNR values of Our and Cur algorithms with Roller Coaster: which are 34.37 and 38.35 dB, respectively. The V-PSNR values are consistent with our MOS results.

In summary, our preliminary user study results (about 40% bandwidth saving) are inline with our earlier experiments using V-PSNR. Our user study also reveals that improving the fixation prediction accuracy for even lower missing ratio is important. It is, however, possible to cope with the imperfect fixation prediction using some innovative networking tools. For example, some studies [224, 226] leverage emerging network protocols, such as HTTP/2 and QUIC [110], to stream critical tiles over high-priority concurrent streams, in order to avoid missing tiles (holes).

3.6 Conclusion

360° video streaming has become increasingly popular. However, the extremely large file sizes impose high loads on networks. In this chapter, we propose to leverage both sensor and content features to predict the viewing probability of each tile in future frames, in order to reduce the network loads and improve the video quality. Several novel enhancements are proposed to improve the prediction performance, including generating virtual viewports, considering future content, reducing the feature sampling rate, and training with larger datasets. We conduct extensive simulations using real traces to quantify the performance of our proposed solution. The evaluation results show that, compared to other algorithms, our proposed fixation prediction network achieves comparable video quality while: (i) saving about 8 Mbps in bandwidth consumption and (ii) cutting the rebuffering time by 40-sec. Besides, our proposed prediction network estimates tile viewing probability in almost real-time.

There are some limitations in this work:

- The predicted viewing probabilities are at the tile level. Therefore, the prediction results are only compatible with fewer tiles (larger resolution). This may be solved by training the network with the smallest resolution of tile that is generally suitable for tiled 360° video streaming.

- The prediction accuracy degrades as the length of the prediction window increases, which is faced by all fixation prediction works. This may be solved by adaptively adjusting the threshold according to the length of the prediction window.

Chapter 4

Production Optimization: Optimal Laddering

Although our fixation prediction network in Chapter 3 successfully predicts the tiled segments that will be viewed by the viewer in the future, the client can only request the representations of the tiled segments that are generated at the production phase. Hence, the *optimal laddering* problem is crucial to determine the encoding ladder that generates the representations for maximizing the overall viewing quality of clients without exceeding the storage limit on the server. Although the optimal laddering problem has been studied for conventional videos [143, 177], solving the same problem for tiled[1] 360° videos [51] to HMDs is much more challenging for a couple of reasons. First, the extremely high resolution of 360° videos consumes much more storage space than conventional videos. Second, the optimal ladders may be different among tiles of the same video. This is because each tile may have different complexity levels and viewing probabilities. For example, the tiles that are frequently viewed (e.g., the main foreground objects) may need to have higher quality levels than those tiles that are rarely viewed (e.g., straight up/down).

In this chapter, we study the optimal laddering problem for tiled 360° videos to HMDs in the production phase. This is different from the majority of the prior work on designing ABR algorithms for tiled 360° videos [5, 147, 161, 164, 167, 216, 226], where the viewing quality of each client in the delivery phase is maximized under a *given* encoding ladder. To the best of our knowledge, Ozcinar et al. [160] is the only work that also computes the encoding ladder for 360° videos. However, their solution allocates an even bitrate to all tiles and assumes that all tiles have the same complexity level and viewing probability. In contrast, we consider a more general setup, which consists of the following:

[1]The modern HEVC codecs [134] spatially divide a video into smaller rectangular subvideos, referred to as tiles, which can be independently streamed and decoded. Tiles are essential for 360° video streaming, in which HMD viewers only watch small portions of the whole 360° videos at any moment.

- *Video model*, which maps the encoding configurations (i.e., QP in this chapter) to expected video distortion levels and bitrates. The video model captures the complexity levels of individual tiles of each video.

- *Viewing probability*, which quantifies the chance of each tile being viewed by HMD viewers. The viewing probability can be estimated from historical data of other HMD viewers or computed using fixation prediction algorithms [50, 52]. We take the former approach in this chapter if not otherwise specified.

- *Client distribution*, which represents the fraction of clients with different amounts of available bandwidth. The clients with the same available bandwidth are considered in the same bandwidth *class* in our problem.

Fig. 4.1 shows an illustrative example of our optimal laddering problem, in which we allocate more resources to the tiles with higher complexity levels and viewing probabilities (e.g., with smaller QP values) and to the representations being requested by more clients (e.g., with larger storage space). The eventual goal is to maximize the overall viewing quality in the delivery phase.

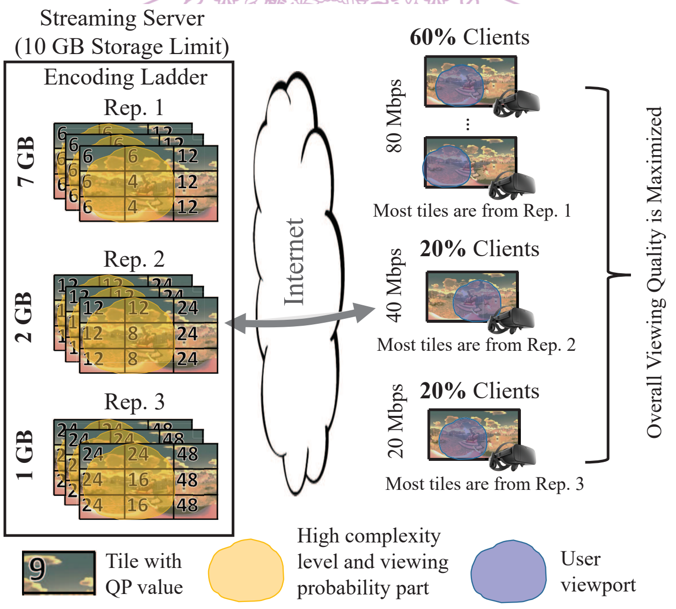


Figure 4.1: An illustrative example of the optimal laddering problem.

To solve the optimal laddering problem, we leverage the divide-and-conquer approach to decompose the problem into two subproblems: (i) per-class optimization and (ii) global optimization. The per-class optimization problem focuses on the optimization for each class with a given available bandwidth. We formulate this problem into a convex optimization problem [23] considering video models and viewing probability. We solve it using a suite of mathematics tools, such as Lagrangian multiplier and rounding, in the Rate-Distortion Optimization (RDO) fashion [192]. However, the complex video models result in non-negligible computation overhead. Thus, we also propose a more efficient greedy algorithm that iteratively sets the encoding configurations of individual tiles. For the global optimization, the goal is to adjust the solution from the per-class optimization to meet the overall storage limit. In this problem, the client bandwidth distribution is considered when optimizing the overall viewing quality across all clients at the given storage limit. We have also proposed two algorithms for global optimization. One of them runs more efficiently and the other one offers better video quality. We have conducted experiments on a real testbed to evaluate our proposed algorithms, compared to the state-of-the-art algorithms. We then recommend a combination of our proposed algorithms to efficiently solve the optimal laddering problem while achieving high viewing quality.

4.1 System Overview

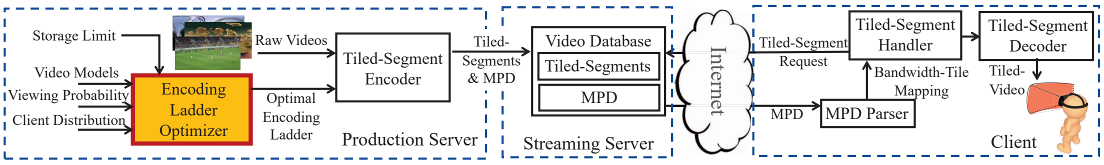


Figure 4.2: System overview of our considered 360° video streaming system.

Fig. 4.2 details our considered streaming system for tiled 360° videos. The components are introduced below.

- **Encoding ladder optimizer** determines the optimal encoding ladder under the storage limit. The resulting ladder is then used for encoding the tiled-segments.
- **Tiled-segment encoder** on the production server compresses and splits the videos into tiled-segments. Each tiled-segment is a tile across multiple consecutive video frames. Tiled-segments are the basic streaming units that can be independently encoded and decoded.
- **Video database** on the streaming server stores the encoded tiled-segments following the encoding ladder. The video database also stores the MPD (Media Presentation Description) files, which provide the meta-data of the representations to the clients.
- **MPD parser** on the client parses the MPD files from the server to get the mapping between representations and URLs.

- **Tiled-segment handler** on the client sends the requests and receives the incoming tiled-segments. It also implements ABR algorithms for selecting the tiled-segments.
- **Tiled-segment decoder** on the client decodes the received tiled-segments for the HMD viewers.

The interactions among these components are as follows. At the production server, raw 360° videos are encoded and segmented into tiled-segments according to the encoding ladder computed by the encoding ladder optimizer. These tiled-segments and the MPD files are stored in the video database on the streaming server. In each streaming session, the MPD parser parses the MPD file for the meta-data of the tiled-segments. The tiled-segment handler then adaptively requests videos from the streaming server. The streamed tiled-segments are decoded by the tiled-segment decoder for the HMD viewers. *Among the above components, the encoding ladder optimizer (shaded block in Fig. 4.2) is the core component studied in this chapter, which will be detailed in the remaining sections.*

4.2 Optimal Laddering Problem

In this section, we first describe our research problem, which is followed by the system models and problem formulation. Table 4.1 summarizes the symbols used in this chapter.

4.2.1 Problem Statement

Our research problem can be described as follows. Given a 360° video server with a storage limit of S, each 360° video is divided into T segments, where each segment is further divided into N tiles. We classify the clients into C classes based on their available bandwidth b_c. A class c client has a probability of $f_{v,c}$ to watch video v. In particular, $f_{v,c} = w_v^v \times w_c^b$. w_v^v denotes the video popularity and w_c^b denotes the fraction of clients at bandwidth class c, where $\sum_{v=1}^{V} w_v^v = 1$ and $\sum_{c=1}^{C} w_c^b = 1$. Let n denote the tile number and q denote the encoding QP[2]. The goal of the encoding ladder optimizer is to make two sets of decisions to minimize the overall viewing distortion. First, the tiles and their representations stored on the streaming server need to be determined. These are captured by the Boolean variables $y_{v,t,n,q} \in \{0, 1\}$, where v denotes the video, t denotes the segment number, n denotes the tile number, and q denotes the encoding QP. $y_{v,t,n,q} = 1$ if and only if video v's tiled-segment (t, n) has a representation with QP q stored on the streaming server. Second, the tiles and their representations that are planned

[2]We focus on controlling QP for rate control, while other parameters may be used as well. For example, studies in the literature propose to employ the Lagrangian multiplier λ [113, 114, 210] instead of QP to control the video quality for higher rate control accuracy.

Table 4.1: Symbol Table

Symbol	**Description**
V	Number of videos
C	Number of bandwidth classes
T	Number of segments
N	Number of tiles
Q	Maximum of available QP values
S	Storage limit on server
$f_{v,c}$	Probability of client in bandwidth class c watching video v (client distribution)
$p_{v,t,n} = p_\phi$	Viewing probability of tile n at segment t for video v
a_n	Area scaling factor of tile n
$x_{v,t,n,c,q} = x_{\phi,c,q}$	Whether tile n with QP q at segment t is selected to be transmitted in bandwidth class c watching video v
$y_{v,t,n,q} = y_{\phi,q}$	Whether tile n with QP q at segment t watching video v is selected to be stored on the server
$D_{v,t,n} = D_\phi$	Maximum distortion of tile n at segment t of video v
$d_{v,t,n}(q) = d_\phi(q)$	Distortion of tile n with QP q at segment t of video v (distortion model)
$r_{v,t,n}(q) = r_\phi(q)$	Bitrate of tile n with QP q at segment t of video v (bitrate model)
b_c	Available bandwidth of bandwidth class c

to be streamed to the clients need to be determined. These are captured by the Boolean variables $x_{v,t,n,c,q} \in \{0,1\}$, where $x_{v,t,n,c,q} = 1$ if and only if the representation with QP q of tiled-segment (t,n) is streamed to class c clients who watch video v. We use ϕ to represent (v,t,n) and $\Phi = \{(v,t,n)|v \in [1,V], t \in [1,T], n \in [1,N]\}$ to denote all possible 3-tuples in the remaining chapter for brevity. For example, we interchangeably write $y_{v,t,n,q}$ and $y_{\phi,q}$ as well as $x_{v,t,n,c,q}$ and $x_{\phi,c,q}$ if they do not cause any ambiguity.

Table 4.2: The Adj. R^2 of the Video Models for the Considered Videos

Video	**Mega Coaster**	**Roller Coaster**	**Shark Shipwreck**	**Hog Rider**	**Chariot Race**	**SFR Sport**
MSE	0.99	0.99	0.99	0.99	0.99	0.93
Bitrate	0.99	0.99	0.99	0.99	0.99	0.99

4.2.2 Video Models

We let $d_\phi(q)$ and $r_\phi(q)$ be the distortion and bitrate models, which are functions of QPs. The video models allow us to estimate the viewing quality and consumed bandwidth when solving the optimal laddering problem. To understand the properties of the models, we divide videos from a public dataset [123] into 6×4 tiles and encode them multiple times with different QPs in $\{1, 8, 14, 20, 26, 32, 38, 44, 51\}$ using Kvazaar [204]. The MSE[3] and bitrate of each tile under these QPs are measured. We then use the measured results to estimate the model parameters. Several possible functions can be adopted for the models, such as linear, power, and exponential functions. Our pilot tests indicate that the linear function has the worst modeling performance. In contrast, the distortion and bitrate functions can be well modeled by the power and exponential functions, respectively. Our findings are inline with other empirical models [28] with only minor differences. Specifically, we write these two models as:

$$d_\phi(q) = \alpha^d_\phi q^{\beta^d_\phi} + \gamma^d_\phi; \tag{4.1}$$

$$r_\phi(q) = \alpha^r_\phi \mathrm{e}^{\beta^r_\phi q}. \tag{4.2}$$

In the models, α^d_ϕ, β^d_ϕ, γ^d_ϕ, α^r_ϕ, and β^r_ϕ are model parameters. We fit the distortion and bitrate models for individual tiled-segments. We plot the MSE and bitrate of a sample tiled-segment from *Mega Coaster* in Figs. 4.3(a) and 4.3(b) under different QPs. These figures reveal that Eqs. (4.1) and (4.2) are reasonably accurate, as the curves closely follow the samples. We also encode the tiled-segment with several additional QPs to evaluate the accuracy of the resulting models. We mark these *additional samples* in the figures with circles, which are also close to the model curves. We notice that figures from video models of other tiled-segments are similar, and are left out due to the limited space. The average adjusted R^2 of the distortion and the bitrate models are reported in Table 4.2. This table confirms the accuracy of our video models. The power and exponential models are both convex functions. This property is utilized by our solution proposed later.

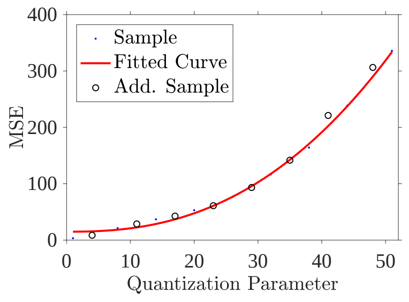
4.2.3 Problem Formulation

The optimal laddering problem is quite hard to solve. We give the proof of the following lemma in Appendix 4.8.

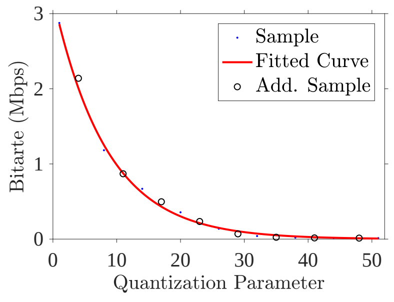
Lemma 1. *The optimal laddering problem is NP-hard.*

We formulate the optimal laddering problem into an Integer Linear Programming (ILP) problem. The decision variables of our formulation are $x_{\phi,c,q}$ and $y_{\phi,q}$. When projecting tiles of the 2D reconstructed videos to the 3D sphere for generating the viewports,

[3] Instead of 360° video quality metrics, we use the MSE of individual tiles to quantify the distortion of the video. This is because our problem is to determine the QP values of individual tiles, i.e., at the tile level, while 360° video quality metrics are at the *video level*. Nonetheless, we employ 360° specific metrics, e.g., V-VMAF, to quantify the overall video quality in our evaluations. We note that, in addition to MSE, other quality metrics at the tile-level may be adopted, such as PSNR or QoE models [226].



(a)



(b)

Figure 4.3: Sample video models: (a) MSE over QP ($d_\phi(q)$) and (b) bitrate over QP ($r_\phi(q)$).

different sphere areas are covered by each tile (mainly due to different latitudes). To account for this, we let a_n be the area scaling factor of tile n, which is defined as the ratio of the area of tile n to that of the whole 3D sphere. Concretely, we let A_n be the area of tile n on the sphere, where $1 \leq n \leq N$. We then write the scaling factor a_n as $\frac{A_n}{\sum_{i=1}^{N} A_i}$. A larger a_n value indicates that tile n affects the resulting viewports more. Note that without incurring ambiguity, we interchangeably write a_ϕ and a_n; we define $a_\phi = a_n$, where $\phi = (v, t, n)$ as a is different only when n changes. Similarly, we let $f_{\phi,c} = f_{v,c}$, where $\phi = (v, t, n)$. With the notations defined so far, we write our problem as:

$$\min \sum_{c=1}^{C} \sum_{\phi \in \mathbf{\Phi}} f_{\phi,c} p_\phi a_\phi \sum_{q=1}^{Q} d_\phi(q) x_{\phi,c,q} \tag{4.3a}$$

$$st: \sum_{n=1}^{N} \sum_{q=1}^{Q} r_{v,t,n}(q) x_{v,t,n,c,q} \leq b_c \quad c \in [1, C], v \in [1, V], t \in [1, T]; \tag{4.3b}$$

$$\sum_{\phi \in \mathbf{\Phi}} \sum_{q=1}^{Q} r_\phi(q) y_{\phi,q} \leq S; \tag{4.3c}$$

$$x_{\phi,c,q} \leq y_{\phi,q} \quad c \in [1, C], q \in [1, Q], \phi \in \mathbf{\Phi}; \tag{4.3d}$$

$$\sum_{q=1}^{Q} x_{\phi,c,q} = 1 \quad c \in [1, C], \phi \in \mathbf{\Phi}; \tag{4.3e}$$

$$x_{\phi,c,q} \in \{0, 1\} \quad c \in [1, C], q \in [1, Q], \phi \in \mathbf{\Phi}; \tag{4.3f}$$

$$y_{\phi,q} \in \{0, 1\} \quad q \in [1, Q], \phi \in \mathbf{\Phi}. \tag{4.3g}$$

The objective function in Eq. (4.3a) minimizes the expected overall distortion in weighted-MSE[4] [125], where the tile weights depend on: (i) viewing probability (p_ϕ) and area scaling factor (a_ϕ) across all tiles and (ii) the probability of clients in different classes watching different videos ($f_{\phi,c}$). Eq. (4.3b) makes sure that the total streamed bitrate to each client class c does not exceed the available bandwidth b_c. Eq. (4.3c) constrains the consumed storage space within the storage limit S on the server. Eq. (4.3d) indicates that clients only select the tiles offered by the server. Besides, each client only selects one

[4]The overall distortion in our formulation is a video-level quality metric, which is essentially a weighted sum of the quality of all the tiles.

representation for each tile as enforced by Eq. (4.3e).

4.3 Problem Decomposition

The optimal laddering problem in Eqs. (4.3a)–(4.3g) is fairly complicated because of the complex interplay between the bandwidth and storage constraints. More specifically, the best solution that satisfies all the bandwidth constraints may exceed the storage limit, while restricting the storage space for each class causes a huge number of permutations on storage space assignments across videos. Hence, we opt for the *divide-and-conquer* approach, as illustrated in Fig. 4.4. In particular, we decompose the optimal laddering problem into the following two subproblems.

- *Per-class optimization* problem optimizes the per-class solution under the bandwidth constraint of each class. It takes the video models and viewing probability of video v and bandwidth constraint of class c as the inputs. It then determines the best QP values for tiled-segment (t, n). The output for class c of video v contains the Boolean values $\{x^*_{v,t,n,c,q}(= x^*_{\phi,c,q})|\forall t, n, q\}$, which are collectively denoted as $\mathbf{X^*_{v,c}}$.

- *Global optimization* problem combines and adjusts all per-class solutions into a global solution under the storage limit. It takes the video models, viewing probability, client distribution, and the storage limit of *all* videos as the inputs. It then adjusts $\mathbf{X^*_{v,c}}$ of each class c to fit all tiled-segments into the storage limit. When the storage limit is loose, all $\mathbf{X}^*_{v,c}$ solutions from the per-class optimization problems may be directly accepted. The output of the global optimization problem contains the revised $\mathbf{X^*_{v,c}}$ for all v and c, which is collectively written as $\mathbf{X}^*$. Moreover, the output also specifies the optimal QP values for stored tiled-segment (t, n) of video v as $\{y^*_{v,t,n,q}(= y^*_{\phi,q})|\forall v, t, n, q\}$, which is collectively written as $\mathbf{Y}^*$. $\mathbf{Y}^*$ is essentially the *optimal encoding ladder*. It is not hard to see that $\mathbf{Y}^*$ is a function of $\mathbf{X}^*$; that is, $y^*_{v,t,n,q} = 1$ if and only if $\sum_{c=1}^{C} x^*_{v,t,n,c,q} \geq 1, \forall v, t, n, q$.

We solve the per-class optimization problem for each class with a bandwidth constraint in Sec. 4.4. We solve the global optimization problem with the storage limit in Sec. 4.5.

4.4 Per-Class Optimization

We first give the formulation, which is followed by two algorithms.

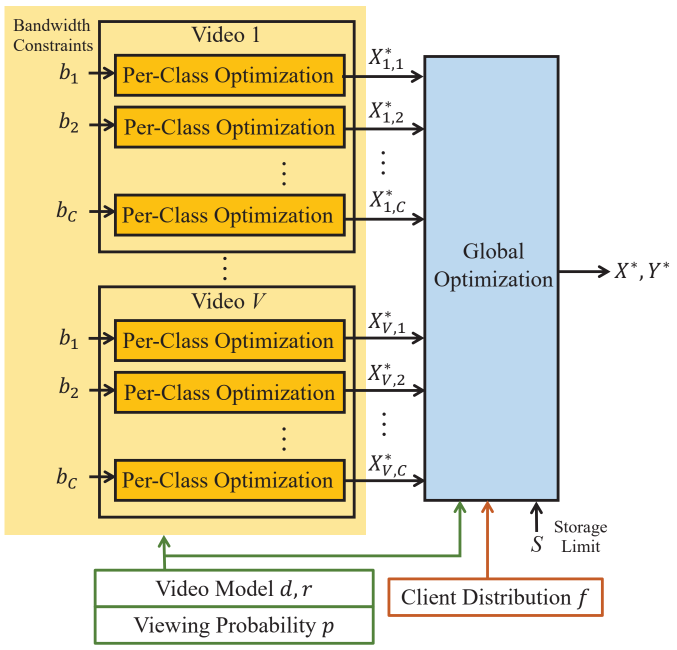


Figure 4.4: The overview of our divide-and-conquer approach.

4.4.1 Per-Class Formulation

Let $P(v, c)$ be the subproblem of bandwidth class c watching video v, where the storage limit is ignored. That is, this subproblem only considers the constraints in Eqs. (4.3a)–(4.3g) that are related to $x_{v,t,n,c,q}$. We formally formulate the problem as:

$$P(v,c) : \min \sum_{t=1}^{T} \sum_{n=1}^{N} p_{v,t,n} a_n \sum_{q=1}^{Q} d_{v,t,n}(q) x_{v,t,n,c,q} \tag{4.4a}$$

$$st : \sum_{n=1}^{N} \sum_{q=1}^{Q} r_{v,t,n}(q) x_{v,t,n,c,q} \leq b_c \qquad t \in [1, T]; \tag{4.4b}$$

$$\sum_{q=1}^{Q} x_{v,t,n,c,q} = 1 \quad t \in [1, T], n \in [1, N]; \tag{4.4c}$$

$$x_{v,t,n,c,q} = \{0, 1\} \quad t \in [1, T], n \in [1, N], q \in [1, Q]. \tag{4.4d}$$

Eq. (4.4a) minimizes the expected distortion for the clients in bandwidth class c who watch video v. Eq. (4.4b) constrains the consumed bitrate within the available bandwidth b_c. Eq. (4.4c) ensures that only one representation is selected.

We next propose two algorithms to solve the formulation in Eqs. (4.4a)–(4.4d): (i) Per-Class Lagrangian-Based Algorithm (PC-LBA), which leverages the convexity of video models to get the solution, and (ii) Per-Class Greedy-Based Algorithm (PC-GBA), which runs more efficiently. Their performance will be compared in Sec. 4.6.

4.4.2 Lagrangian-Based Algorithm: PC-LBA

PC-LBA consists of two steps: (i) a QP optimizer, which employs the Lagrangian multiplier [17] to find the optimal QPs as real numbers, and (ii) the optimal rounding algorithm, which solves an ILP formulation to optimally round the QPs to integers supported by the encoder.

QP optimizer. To adopt the Lagrangian approach, we *transform* the discrete decision variables $x_{v,t,n,c,q}$ into continuous decision variables $\kappa_{v,t,n,c}$. We let $\kappa_{v,t,n,c}$ represent the QP value of tiled-segment (t, n) of video v streamed to class c clients. $\kappa_{v,t,n,c}$ is a real number in the range of $[\kappa_{min}, \kappa_{max}]$, where κ_{min} and κ_{max} are the QP bounds from the video encoder. With $\kappa_{v,t,n,c}$, the transformed formulation has fewer decision variables and gets rid of Eq (4.4c). Moreover, we observe that the decisions on different segments are independent. Hence, we write each $P(v, c)$ into a series of transformed $P'(v, t, c)$ for all $t \in [1, T]$ as:

$$P'(v,t,c) = \min \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,t,n,c}) p_{v,t,n} a_n \tag{4.5a}$$

$$st: \sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c}) \le b_c; \tag{4.5b}$$

$$\kappa_{v,t,n,c} \in [\kappa_{min}, \kappa_{max}]. \tag{4.5c}$$

In this formulation, Eqs. (4.5a) and (4.5b) account for the expected distortion and consumed bitrate for clients in class c watching segment t of video v. The following two lemmas show that the transformed formulation in Eqs. (4.5a)–(4.5c) can be solved efficiently. The proofs are given in Appendix 4.8 to maintain the flow of the chapter.

Lemma 2. *When the power function in Eq. (4.1) is adopted as the distortion model, the objective function in Eq. (4.5a) is convex.*

Lemma 3. *When the exponential function in Eq. (4.2) is adopted as the bitrate model, the constraint in Eq. (4.5b) is convex.*

We write $\{\kappa_{v,t,1,c}, \kappa_{v,t,2,c}, \cdots, \kappa_{v,t,N,c}\}$ as $\mathbf{K_{v,t,c}}$ in the following for the sake of presentation. Combining Lemmas 2 and 3, we know that our optimization problem is a convex programming problem, which can be solved using the Lagrangian multiplier as follows. We first introduce a Lagrangian multiplier $\mu \in \mathbb{R}^+$, and rewrite our (constrained) convex programming problem into an unconstrained optimization problem:

$$\begin{aligned} \min\ L(\mathbf{K_{v,t,c}}, \mu) = & \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,t,n,c}) p_{v,t,n} a_n \\ & + \mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c}) - b_c) \end{aligned} \tag{4.6}$$

Consider Eq. (4.6) as the primal Lagrangian problem; the Lagrangian dual function g

minimizes the Lagrangian value over $\mathbf{K_{v,t,c}}$:

$$g(\mu) = \inf_{\mathbf{K_{v,t,c}}} (\mathbf{K_{v,t,c}}, \mu) = \inf_{\mathbf{K_{v,t,c}}} (\sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,t,n,c}) p_{v,t,n} a_n + \mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c}) - b_c)). \tag{4.7}$$

Lemma 4. *The Lagrangian dual function (Eq. (4.7)) constitutes a lower bound for the objective value of any feasible solution to the Lagrangian primal problem (Eq. (4.6)). In fact, because the strong duality holds here, the optimal solution of the Lagrangian dual problem is also the optimal solution of the original (primal) problem.*

To solve the Lagrangian dual problem, we first calculate the partial derivative w.r.t. each $\kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$:

$$\frac{\partial L}{\partial \kappa_{v,t,n,c}} = (\alpha^d_{v,t,n} \beta^d_{v,t,n} \kappa_{v,t,n,c}^{\beta^d_{v,t,n}-1}) p_{v,t,n} a_n + \mu \alpha^r_{v,t,n} \beta^r_{v,t,n} \mathrm{e}^{\beta^r_{v,t,n} \kappa_{v,t,n,c}} = 0. \tag{4.8}$$

Then, we utilize the Lambert W function [39] to represent each $\kappa_{v,t,n,c}$ using μ:

$$\kappa_{v,t,n,c} = \frac{1-\beta^d_{v,t,n}}{\beta^r_{v,t,n}} W(\frac{\beta^r_{v,t,n}}{1-\beta^d_{v,t,n}} \mathrm{e}^{\frac{-\ln \frac{\mu \alpha^r_{v,t,n} \beta^r_{v,t,n}}{-\alpha^d_{v,t,n} \beta^d_{v,t,n} p_{v,t,n} a_n}}{1-\beta^d_{v,t,n}}}). \tag{4.9}$$

Last, we substitute $\kappa_{v,t,n,c}$ into Eq. (4.7) to derive the optimal μ and the corresponding $\mathbf{K_{v,t,c}}$. Notice that, if some optimal solution $\kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$ falls outside of $[\kappa_{\text{min}}, \kappa_{\text{max}}]$, the QP optimizer caps $\kappa_{v,t,n,c}$ at κ_{min} or κ_{max}. It then recalculates $\mathbf{K_{v,t,c}}$ until all QP values fall in the practical range of $[\kappa_{\text{min}}, \kappa_{\text{max}}]$.

```
1: $\mathbf{X^*_{v,c}} \leftarrow \emptyset$
2: for $t \leftarrow 1$ to $T$ do
3:     // QP optimizer
4:     $\mathbf{K_{v,t,c}} \leftarrow$ Solved with Eqs. (4.7) and (4.9)
5:     while $\exists \kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$, where $\kappa_{v,t,n,c} \notin [\kappa_{\text{min}}, \kappa_{\text{max}}]$ do
6:         Set the out-of-range $\kappa_{v,t,n,c}$ to the closest border
7:         $\mathbf{K_{v,t,c}} \leftarrow$ Solved with Eqs. (4.7) and (4.9)
8:     // Optimal rounding algorithm
9:     Solve Eqs. (4.10a)–(4.10d) for $\mathbf{K^*_{v,t,c}}$
10:    Transform $\mathbf{K^*_{v,t,c}}$ to $\mathbf{X^*_{v,t,c}}$
11:    $\mathbf{X^*_{v,c}} \leftarrow \mathbf{X^*_{v,c}} \cup \mathbf{X^*_{v,t,c}}$
12: return $\mathbf{X^*_{v,c}}$
```

Figure 4.5: The pseudocode of the PC-LBA algorithm.

Optimal rounding algorithm. Next, we round the real number QPs in $\mathbf{K}_{v,t,c}$ into integer QPs for the video encoder. We let $\mathbf{K'_{v,t,c}}$ be a subset of $\mathbf{K}_{v,t,c}$ containing tiles in $\mathbf{K_{v,t,c}}$ with non-integer optimal QP values. Our problem is to determine whether to round each $\kappa_{v,t,n,c}$ in $\mathbf{K}'_{\mathbf{v,t,c}}$ up or down to minimize the resulting distortion without consuming excessive bandwidth. For $n \in [1, \left|\mathbf{K}'_{\mathbf{v,t,c}}\right|]$, we define the decision variable $z_{v,t,n,c,0} = 1$ if $\kappa_{v,t,n,c}$ is rounded down, and $z_{v,t,n,c,0} = 0$ otherwise. Similarly, we define $z_{v,t,n,c,1} = 1$ if $\kappa_{v,t,n,c}$ is rounded up and $z_{v,t,n,c,1} = 0$ otherwise. The optimal rounding problem can then be written as:

$$\min \sum_{n=1}^{|\mathbf{K}'_{\mathbf{v,t,c}}|} z_{v,t,n,c,0} d_i(\lfloor \kappa'_{v,t,n,c} \rfloor) + z_{v,t,n,c,1} d_{v,t,n}(\lceil \kappa'_{v,t,n,c} \rceil) \tag{4.10a}$$

$$st: \sum_{n=1}^{|\mathbf{K}'_{\mathbf{v,t,c}}|} z_{v,t,n,c,0} r_i(\lfloor \kappa'_{v,t,n,c} \rfloor) + z_{v,t,n,c,1} r_i(\lceil \kappa'_{v,t,n,c} \rceil)$$

$$\leq \sum_{n=1}^{|\mathbf{K}'_{\mathbf{v,t,c}}|} r_{v,t,n}(\kappa'_{v,t,n,c}); \tag{4.10b}$$

$$z_{v,t,n,c,0} + z_{v,t,n,c,1} = 1 \qquad n \in [1, |\mathbf{K}'_{\mathbf{v,t,c}}|]; \tag{4.10c}$$

$$z_{v,t,n,c,0}, z_{v,t,n,c,1} \in \{0, 1\} \qquad n \in [1, |\mathbf{K}'_{\mathbf{v,t,c}}|]; \tag{4.10d}$$

In this formulation, Eq. (4.10a) minimizes the additional distortion due to rounding, while Eq. (4.10b) makes sure that the bitrate does not exceed the bandwidth constraint. Eq. (4.3g) ensures that each QP value is either rounded down or up, but not both. The formulation can be optimally solved for $\mathbf{K}^*_{\mathbf{v,t,c}}$ using existing solvers, like CPLEX [81] and GLPK [129]. Last, we transform the optimal $\mathbf{K}^*_{\mathbf{v,t,c}}$ back to $\mathbf{X}^*_{\mathbf{v,t,c}} = \{x^*_{v,t,n,c,q} | q \in [1, Q], n \in [1, N]\}$ of the original problem.

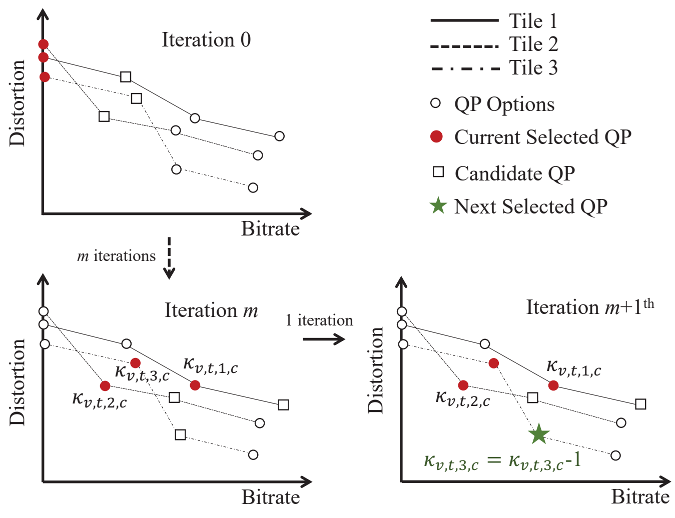


Figure 4.6: An illustrative example of PC-GBA with several iterations.

Pseudocode. Fig. 4.5 presents the pseudocode of our PC-LBA. Lines 4–7 repeatedly solve Eqs. (4.7) and (4.9) until there is no out-of-range $\kappa_{v,t,n,c} \in \mathbf{K}_{\mathbf{v,t,c}}$. Line 9 rounds the real QP values into integers. Line 10 transforms the QP set $\mathbf{K}_{\mathbf{v,t,c}}$ to the binary set $\mathbf{X}_{\mathbf{v,t,c}}$. Line 11 collects the solution for each segment. Line 12 returns the optimal solution

```
1: $\mathbf{X^*_{v,c}} \leftarrow \emptyset$
2: for $t \leftarrow 1$ to $T$ do
3:     Set $\kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$ as $\kappa_{\max}$, $\forall i = 1, 2, \ldots, N$
4:     $B' = b_c - \sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c})$
5:     while $B' > 0$ do
6:         //Tile Selector
7:         $\Theta \leftarrow \{\theta_{v,t,n,c} | n = 1, 2, \ldots, N\}$
8:         $n^* \leftarrow \arg\max \Theta$
9:         //Status Tracker
10:        $B' = B' - (r_{v,t,n^*}(\kappa_{v,t,n^*,c} - 1) - r_{v,t,n^*}(\kappa_{v,t,n^*,c}))$
11:        $\kappa_{v,t,n^*,c} = \kappa_{v,t,n^*,c} - 1$
12:    Transform $\mathbf{K^*_{v,t,c}}$ to $\mathbf{X^*_{v,t,c}}$
13:    $\mathbf{X^*_{v,c}} \leftarrow \mathbf{X^*_{v,c}} \cup \mathbf{X^*_{v,t,c}}$
14: return $\mathbf{X^*_{v,c}}$
```

Figure 4.7: The pseudocode of the PC-GBA algorithm.

$x^*_{v,t,n,c,q} \in \mathbf{X^*_{v,c}}$ for clients in bandwidth class c watching video v. The following lemma analyzes the complexity of PC-LBA.

Lemma 5. *The PC-LBA algorithm runs in time $O(T2^N)$ with space complexity of $O(N)$.*

4.4.3 Greedy-based Algorithm: PC-GBA

PC-LBA may suffer from higher computational complexity due to the ILP formulation of optimal rounding[5]. Therefore, we also propose a more efficient greedy algorithm to solve the problem with discrete QPs. Our greedy algorithm contains two components: (i) status tracker and (ii) tile selector. The status tracker keeps track of the current QP selected for each tile, their accumulated bitrate, and the remaining bandwidth. The tile selector selects the tile with the highest *coding efficiency* to allocate more bitrate by reducing its QP. The status tracker then updates the accumulated bitrate and the remaining bandwidth. The above steps *iterate* until there is no remaining bandwidth or all tiles are coded at the smallest QP values.

The crux of the greedy algorithm is the definition of the coding efficiency $\theta_{v,t,n,c}$ when allocating the additional bitrate to tiled-segment (t, n) of video v. We let $\kappa_{v,t,n,c}$ be the current selected QP for tiled-segment (t, n) of v streamed to class c clients. We then define $\theta_{v,t,n,c}$ as:

$$\frac{[d_{v,t,n}(\kappa_{v,t,n,c} - 1) - d_{v,t,n}(\kappa_{v,t,n,c})]p_{v,t,n}a_n}{r_{v,t,n}(\kappa_{v,t,n,c} - 1) - r_{v,t,n}(\kappa_{v,t,n,c})}, \tag{4.11}$$

where $p_{v,t,n}$ is the viewing probability and a_n is the area scaling factor. $\theta_{v,t,n,c}$ is essentially the slope of the rate-distortion curves at $\kappa_{v,t,n,c}$. Fig. 4.6 shows an illustrative example of

[5]Notice that when optimality is not a major concern, much simpler rounding algorithms, such as rounding down, can be adopted for lower computation complexity.

the proposed PC-GBA streaming a video with three tiles. The current QPs are marked with solid dots, which are selected from all QP options marked with circles. Besides, the next candidate QPs of each tile are represented by squares. In this figure, after iteration m, the tile selector chooses tile three to allocate more bandwidth since it has the steepest slope. Note that if all tiles have the same coding efficiency, we use the weights $p_{v,t,n}a_n$ and then the QPs to break the ties. After all the QP values are determined, we transform $\kappa^*_{v,t,n,c} \in \mathbf{K^*_{v,t,c}}$ to binary indicators $x^*_{v,t,n,c,q} \in \mathbf{X^*_{v,t,c}}$.

Pseudocode. Fig. 4.7 presents the pseudocode of our PC-GBA algorithm that determines the QPs for segments. Lines 3–4 initialize the QP of each tile at the maximum QP and the remaining bandwidth. The while loop between lines 5–11 iteratively allocates more bitrate to the tile with the highest coding efficiency selected in line 8. Lines 10–11 update the status through the status tracker, including the QP value of the selected tile and the remaining bandwidth. Line 12 transforms the determined QP set $\mathbf{K_{v,t,c}}$ to binary set $\mathbf{X_{v,t,c}}$. Line 13 collects the solution for each segment. Line 14 returns the optimal solution $x^*_{v,t,n,c,q} \in \mathbf{X^*_{v,c}}$ for clients in bandwidth class c of video v.

Lemma 6. *The PC-GBA algorithm runs in time $O(TN(\log N)Q)$ with space complexity of $O(N)$.*

4.5 Global Optimization for the Optimal Ladders

We propose two greedy algorithms to solve the global optimization problem. The first algorithm directly solves the problem and considers all possible storage space allocation across multiple videos, which we refer to as the Global InTeR-video Algorithm (GL-ITRA). The second algorithm simplifies the problem by: (i) equally dividing the storage space among all videos and (ii) assuming that the video popularity is uniformly distributed across all videos. We refer to this algorithm as the GLobal InTrA-video Algorithm (GL-ITAA). We present these two algorithms below and compare their performance in Sec. 4.6.

GL-ITRA Algorithm. We propose a greedy algorithm to adjust the per-class solutions $\mathbf{X^*_{v,c}}$ for minimizing the expected distortion while meeting both the client bandwidth constraints and the overall server storage limit. First, $y_{v,t,n,q}$ is initialized as 1 if and only if $\sum_{c=1}^{C} x_{v,t,n,c,q} \geq 1$, where $v \in [1,V], t \in [1,T], n \in [1,N]$,and $q \in [1,Q]$; $y_{v,t,n,q}$ is set to be 0, otherwise. We introduce a constant system parameter δ to denote the step size of QP adjustments. We then compute the weight of each tile considering the client distribution as $\epsilon_{v,t,n,c,q} =$

$$\frac{\sum_{v=1}^{V}\sum_{c=1}^{C} f_{v,c} \cdot [d_{v,t,n}(q+\delta) - d_{v,t,n}(q)]p_{v,t,n}a_n x_{v,t,n,c,q}}{[r_{v,t,n}(q) - r_{v,t,n}(q+\delta)(1 - y_{v,t,n,q+\delta})]y_{v,t,n,q}}. \tag{4.12}$$

$\epsilon_{v,t,n,c,q}$ is the ratio between the weighted distortion gain and the storage reduction if the QP value of tiled-segment (t, n) of v increases[6]. In particular, $d_{v,t,n}(q + \delta) - d_{v,t,n}(q)$ is the distortion gain, while $\sum_{v=1}^{V} \sum_{c=1}^{C} f_{v,c}$ and $p_{v,t,n} a_{v,t,n}$ are the weights. $r_{v,t,n}(q) - r_{v,t,n}(q+\delta)(1-y_{v,t,n,q+\delta})$ denotes the reduced storage space, while $(1-y_{v,t,n,q+\delta})$ indicates whether tiled-segment (t, n) of v has already been chosen to be streamed. That is, if $y_{v,t,n,q+\delta} = 1$, then the reduced storage space is $r_{v,t,n}(q)$; otherwise the reduced storage space is $r_{v,t,n}(q) - r_{v,t,n}(q + \delta)$. The algorithm iteratively increases the QP of the tile having the lowest $\epsilon_{v,t,n,c,q}$ by step size δ until the storage limit S is not exceeded.

```
 1: Initialize $\mathbf{Y}^* = \{y_{v,t,n,q} = 0\}$
 2: // Per-class optimization
 3: $X^* = \{\mathbf{X}^*_{\mathbf{v,c}} | \forall v, c\} \leftarrow$ PC-LBA or PC-GBA
 4: for $v \leftarrow 1$ to $V$, $t \leftarrow 1$ to $T$, $n \leftarrow 1$ to $N$,$q \leftarrow 1$ to $Q$ do
 5:     if $\sum_{c=1}^{C} x_{v,t,n,c,q} \geq 1$ then
 6:         $y_{v,t,n,q} \leftarrow 1$
 7: $S' \leftarrow \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{n=1}^{T} \sum_{q=1}^{Q} r_{v,t,n}(q) y_{v,t,n,q}$
 8: // Global optimization
 9: while $S' > S$ do
10:     $\mathbf{E} \leftarrow \{\epsilon_{v,t,n,c,q} | v \in [1, V], t \in [1, T], n \in [1, N], c \in [1, C], q \in [1, Q]\}$ using Eq. (4.12)
11:     $(v^*, t^*, n^*, c^*, q^*) \leftarrow \arg\min E$
12:     $x_{v^*,t^*,n^*,c^*,q^*} \leftarrow 0$
13:     $x_{v^*,t^*,n^*,c^*,q^*+\delta} \leftarrow 1$
14:     Update $y_{v^*,t^*,n^*,q^*}$, $y_{v^*,t^*,n^*,q^*+\delta}$, and $S'$
15: return $\mathbf{X}^*$,$\mathbf{Y}^*$
```

Figure 4.8: The pseudocode of the proposed GL-ITRA algorithm.

Pseudocode. Fig. 4.8 shows the pseudocode of the proposed GL-ITRA algorithm. Lines 1–3 initialize $\mathbf{Y}^*$ and compile $\mathbf{X}^*$ using the PC-LBA or PC-GBA algorithms. Lines 4–6 set $y_{v,t,n,q}$ according to $x_{v,t,n,c,q}$. Line 7 initializes the current storage size S'. Lines 9–14 greedily select the tile to adjust its QP value according to Eq. (4.12) iteratively until the required storage space S' fits the storage limit S. Lines 12–14 update $\mathbf{X}^*$ and $\mathbf{Y}^*$, and the current required storage space. Line 15 returns the decisions $\mathbf{X}^*$ and $\mathbf{Y}^*$.

Lemma 7. *The GL-ITRA algorithm runs in time $O(VTNC(\log VTNC)\frac{Q}{\delta})$ with space complexity of $O(VTNC)$.*

GL-ITAA Algorithm. We propose a simplified greedy algorithm for lower time and space complexity. In particular, we assume that the storage space is uniformly assigned

[6] $\epsilon_{v,t,n,c,q}$ is undefined when $y_{v,t,n,q} = 0$, which is not an issue because the term is never considered in the algorithm.

to all videos and the probabilities of videos (w_v^v) are equal. The global optimization problems of individual videos are then decoupled and can be independently solved.

```
7: $S_v \leftarrow \frac{S}{V}, \forall v = 1, 2, \cdots, V$
8: **for** $v \leftarrow 1$ to $V$ **do**
9:     $S' \leftarrow \sum_{t=1}^{T} \sum_{n=1}^{T} \sum_{q=1}^{Q} r_{v,t,n}(q) y_{v,t,n,q}$
10:    **while** $S' > S_v$ **do**
11:        $\mathbf{E} \leftarrow \{\epsilon_{v,t,n,c,q} | t \in [1,T], n \in [1,N], c \in [1,C], q \in [1,Q]\}$ using Eq. (4.12)
12:        $(t^*, n^*, c^*, q^*) \leftarrow \arg\min E$
13:        $x_{v,t^*,n^*,c^*,q^*} \leftarrow 0$
14:        $x_{v,t^*,n^*,c^*,q^*+\delta} \leftarrow 1$
15:        Update $y_{v,t^*,n^*,q^*}$, $y_{v,t^*,n^*,q^*+\delta}$, and $S'$
16: **return** $\mathbf{X}^*$,$\mathbf{Y}^*$
```

Figure 4.9: The pseudocode of the proposed GL-ITAA algorithm. Note that lines 1–6 are identical to those in Fig. 4.8 and thus are omitted.

Pseudocode. The pseudocode of GL-ITAA differs from GL-ITRA (in Fig. 4.8) only from line 7, which is presented in Fig. 10. Line 7 initializes the storage limit S_v for each video v. Lines 8–15 greedily adjust the QP values of the selected tiles for each video v. Line 16 returns the $\mathbf{X}^*$ and $\mathbf{Y}^*$ aggregated from the decision of each video. Because the optimization problems of individual videos can be independently solved, its time and space complexities are $O(VTNC(\log TNC)\frac{Q}{\delta})$ and $O(TNC)$.

4.6 Evaluations

Table 4.3: The Comparisons among Ours and Relevant Algorithms

Method	**Phase**	**Problem**	**Objective**	**Constrains**	**Inputs**			**Tile Qualities per Segment**	**Solution Approach**
					Video Model	Viewing Prob.	Client Dist.		
Our Algorithms	Production	Optimal laddering	Viewed distortion minimization	Bandwidth, storage space	✓	✓	✓	Multiple	Lagrangian, ILP, and Greedy
Ozcinar et al. [160] (ISM)	Production	Optimal laddering	Overall distortion & cost minimization	Bandwidth, storage space, min. bitrate gap	✓		✓	Single	ILP
Chakareski et al. [28] (ICC)	Delivery	Per-class	Viewed distortion minimization	Bandwidth	✓	✓		Multiple	Convex Optimization
Corbillon et al. [36] (MM)	Delivery	Per-class	Viewed quality maximization	Bandwidth, max. bitrate gap		✓		Multiple	Heuristic

We compare our proposed algorithms against the state-of-the-art ones through extensive experiments in this section.

4.6.1 Implementations

Table 4.3 summarizes our algorithms and the state-of-the-art algorithms. In this table, Ozcinar et al. [160] and our algorithms solve the optimal laddering problem in the production phase, while other algorithms only solve the per-class optimization problem in the delivery phase. Furthermore, the per-class optimization algorithms (Corbillon et al. [36] and Chakareski et al. [28]) do not consider the client distribution. In addition, Corbillon et al. [36] assume that all tiles have the same complexity levels and do not take video models into account. Besides, the other optimal laddering algorithm (Ozcinar et al. [160]) does not consider the viewing probability of each tile. *Compared to the abovementioned algorithms [28, 36, 160], our algorithms are the only ones that consider all three features: video models, viewing probability, and client distribution.*

We have implemented our proposed algorithms and three state-of-the-art algorithms [28, 36, 160] for evaluations and comparisons. Note that we use Python for implementations as much as we can. Certain optimization problems, however, can be efficiently solved with specialized solvers that are not written in Python. For example, cvx [67] and CPLEX [81] are tailored for solving convex optimization and ILP problems, respectively. Similarly, MATLAB [131] supports optimization problems with symbolic variables. We opt to call these specialized solvers from Python rather than reimplementing them, in order to understand the performance achieved by real-life implementations. In particular, we use MATLAB [131] and CPLEX [81] to implement PC-LBA's QP optimizer and optimal rounding algorithm, respectively. We present the detailed implementations of the state-of-the-art algorithms in the following.

- **Per-class optimization algorithms:**
  - **Chakareski et al. [28] (ICC)** formulate the tile quality selection problem into an ILP problem and propose to solve it using convex optimization. We use cvx [67] to implement this algorithm.
  - **Corbillon et al. [36] (MM)** propose a heuristic algorithm for bitrate allocation within a 360° video [215]. Their algorithm can be extended for tiled videos by classifying tiles into foreground and background tiles. Their classification [215] is based on the ground truth of the user viewport, which is impractical because of the network latency. Therefore, we use 25 percentile of viewing probability as the threshold for the classification[7]. A 3.5 ratio of maximum bitrate gap between the maximum surface bitrate and minimum surface bitrate is set following the recommendation in their paper. Each tile is

[7]We also tried 50 and 75 percentile and observed similar results, which are left out for clarity.

allocated the bitrate proportional to the area scaling factor a_n. We use Python to implement this algorithm.

- **Optimal laddering algorithms:**
  - **Ozcinar et al. [160] (ISM)** formulate the optimal laddering problem into an ILP problem without proposing any efficient algorithms. In addition to the storage limit, they also consider different resolutions and introduce a 1.2 ratio of minimum bitrate gap between any two adjacent representations. We use CPLEX to implement this algorithm.

We adopt the same video models in Sec. 4.2.2 for all algorithms for fair comparisons.

4.6.2 Setup

Parameters, traces, and videos. Several system parameters are varied in our experiments. We fit the bandwidth CDF curve following Cisco's forecast on 2019 fixed broadband bandwidth in North America [33]. If not otherwise specified, we select the bandwidth classes in $\{3.12, 4.68, 7.02, 10.52, 15.78, 23.67, 35.51, 53.28, 79.91, 119.87\}$ Mbps, which is geometric progression with 1.5 times that covers the bandwidth range of the bandwidth CDF curve. Besides, we adopt the smallest step size of 1 for optimal overall distortion. Such fine-grained step size results in slightly longer running time, which however is insignificant (up to 7% in our experiments) compared to the video encoding time. We randomly select 10 users from the 50 users in a public dataset [123] to evaluate the performance of all algorithms. The remaining 40 users are used to derive the viewing probability for fair comparisons. Each selected user watches six 1-min videos. The videos are classified into three categories:(i) Computer-Generated, Fast-Paced (CG-FP), (ii) Natural Image, FP (NI-FP), and (iii) NI, Slow-Paced (NI-SP). Two videos are selected from each video category at a resolution of 3840×1920 with 6×4 tiles; 6×4 tiles have been shown to achieve the best tradeoff between viewport flexibility and bitrate overhead [66], but other numbers of tiles can also be used with our proposed solutions. The considered encoding QPs are in [1,51] if not otherwise specified. Besides, we take the number of views of the considered videos on YouTube as the video popularity in the evaluation. For example, the most popular Hog Rider and Mega Coaster account for the two highest ratios, which are about 38.69% and 32.36%, respectively.

For conservative comparisons, we let ISM take additional resolutions in $\{2560\times1280, 1920\times960\}$ into consideration. Because the production server has limited memory, the ISM algorithm can only consider a QP step of 5 in [1,51] without exceeding a memory consumption of 12 GB. Besides, the ISM algorithm tends to terminate after many days, and we set a practical time limit of 2 hours for each video, which is more than 3 times the average running time of our algorithms.

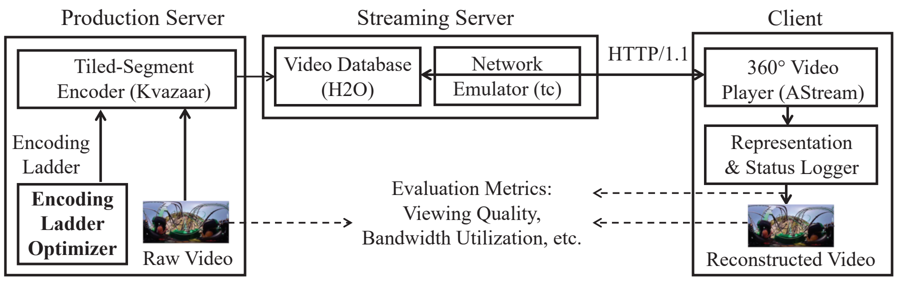

Production Server
Streaming Server
Client
Tiled-Segment Encoder (Kvazaar)
Video Database (H2O)
Network Emulator (tc)
HTTP/1.1
360° Video Player (AStream)
Representation & Status Logger
Encoding Ladder
Encoding Ladder Optimizer
Raw Video
Evaluation Metrics: Viewing Quality, Bandwidth Utilization, etc.
Reconstructed Video

Figure 4.10: The overview of our evaluation testbed.

Testbed. We built a real streaming testbed following the one implemented in Yen et al. [226]. The testbed is illustrated in Fig. 4.10. We set up two Intel i7 workstations with 16 GB RAM running Linux. One of them contains both the production and streaming servers and the other one runs the client. The two workstations are directly connected to each other with a GigE network cable. We employ `tc` [16] in Linux to throttle the network bandwidth. On the server, we adopt *Kvazaar* [204] as the tiled segment encoder to encode the videos. We use *H2O* [45] as the HTTP server to store the representations following the decisions from the encoding ladder optimizer. On the client, we use a Python-based DASH client, *AStream* [95] as the 360° video player. Besides, the status, such as throughput and stalls, and the received representations are logged for further analysis.

We have implemented an ABR algorithm [164] designed for 360° video streaming in the player[8]. Our implemented ABR algorithm performs viewport prediction based on the user's previous orientations. The scene is split into three parts: (i) *viewport*, which is the predicted user's viewport, (ii) *extended area*, which is 30° outside the predicted viewport, and (iii) *background*, which is the remaining tiles of the scene. The ABR algorithm first allocates the lowest representation to each part, then allocates the highest affordable representation to the viewport. The residual bandwidth is allocated to the extended area, followed by the background. To accommodate some background traffic, we instruct the ABR algorithm to set the available bandwidth at 70% of the measured throughput.

We evaluate the results using the following metrics:

- **Viewing quality.** We consider V-PSNR [228], V-SSIM, and V-VMAF, which essentially are the PSNR [116], SSIM [29], and VMAF [142] in the HMD viewports. The computation of V-VMAF is similar to that in Ozcinar et al. [159]. Because V-SSIM and V-VMAF are not pixel-wise metrics, we cut each circle viewport[9] into

[8]Other ABR algorithms in the literature can be adopted in our work as well.
[9]Our HMD (Oculus DK2) is measured to have a circle viewport [50].

its inscribed square when computing them. We believe the negative impacts of removing small areas close to the viewport borders are insignificant as they are far away from the viewport center.

- **Bandwidth utilization.** The ratio between the streamed bitrate to the total bandwidth.
- **Number of stalls.** The total number of stalls of the algorithms throughout each 1-min playout.
- **Running time.** The consumed time of the algorithms for determining the encoding ladder.
- **MPD overhead.** The ratio between the size of the MPD file (meta-data) and the total streamed data in each streaming session.

In the following sections, we first evaluate the per-class optimization algorithms in terms of distortion and viewing quality. After that, we conduct extensive experiments on the testbed to evaluate the performance of optimal laddering algorithms, which solve the global optimization problem. This is followed by a summary of our key findings. The results are reported with a 95% confidence interval plotted as an error bar.

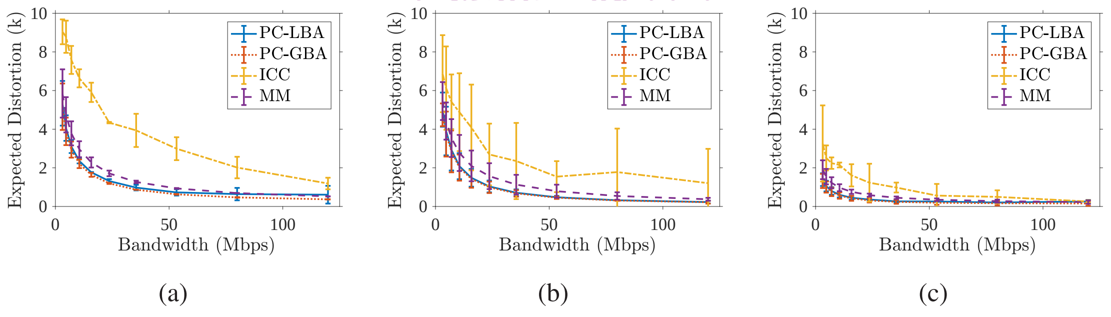


Figure 4.11: Expected distortion for different bandwidths: (a) CG-FP, (b) NI-FP, and (c) NI-SP.

4.6.3 Per-Class Optimization Results

Our proposed PC-LBA and PC-GBA algorithms effectively offer lower expected distortion. Fig. 4.11 plots the expected distortion under different bandwidth levels for different video categories, which is computed with Eq. (4.4a). This figure shows that our proposed algorithms effectively reduce the expected distortion compared to other state-of-the-art algorithms. ICC sometimes (around $\frac{1}{3}$ of the time) fails to find the solutions

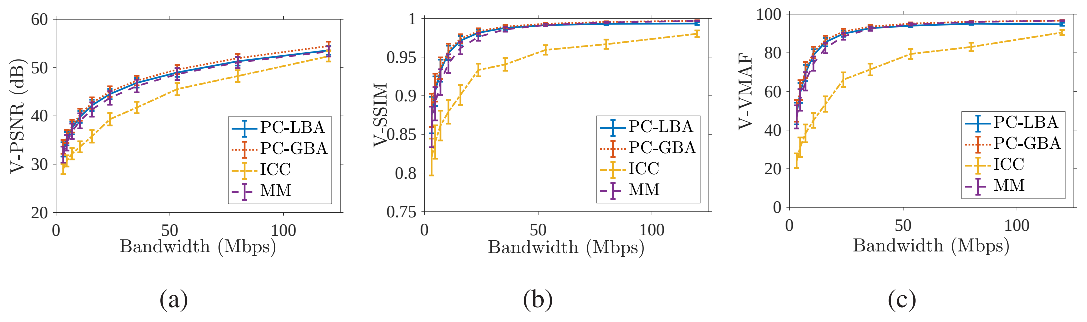


Figure 4.12: The viewing quality for different bandwidths: (a) V-PSNR, (b) V-SSIM, and (c) V-VMAF.

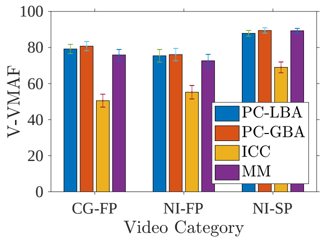


Figure 4.13: The viewing quality of different video categories.

Table 4.4: Maximum and Average of Quality Improvement in V-VMAF Compared to State-of-the-Art Algorithms

PC-LBA		
State-of-the-Art	**Avg.**	**Max.**
ICC	22.57	50.19
MM	1.55	24.82

PC-GBA		
State-of-the-Art	**Avg.**	**Max.**
ICC	23.85	52.17
MM	2.83	26.35

Table 4.5: The Average Per-Segment Running Time of PC-LBA and PC-GBA

Algorithm	**PC-LBA**			**PC-GBA**
Component	**QP Optimizer**	**Optimal Rounding Algorithm**	**Total**	**Total**
Time (s)	73.8315	44.6170	118.4485	0.1074

using cvx, and thus results in the highest expected distortion across all videos. In contrast, our PC-LBA algorithm is a customized algorithm solving a convex programming problem for low distortion. The greedy PC-GBA algorithm also achieves comparable distortion to that of the PC-LBA algorithm. MM is a heuristic algorithm that results in slightly higher distortion compared to our algorithms. Across different video categories, videos in NI-SP result in lower expected distortion in general. This is because NI-SP videos contain simpler scenes and slower movements compared to other videos, which lead to higher coding efficiency.

Our proposed algorithms outperform others more in viewing quality at lower bandwidth. We plot the viewing quality of all videos in the user's viewport in Fig. 4.12. All three quality metrics demonstrate the same trend: our proposed algorithms deliver higher viewing quality than others in most bandwidth classes, especially at lower bandwidths. We emphasize that the performance of low bandwidth classes is more crucial, as clients in these classes are more vulnerable to inferior viewing experience. Because three viewing quality metrics show similar trends, we only report the quality in V-VMAF in the

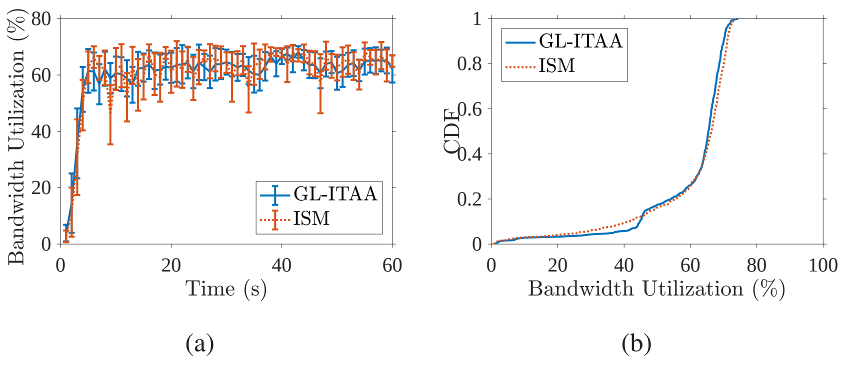


Figure 4.14: The bandwidth utilization across 10 users watching a sample video at storage limit S=1200 MB: (a) over time and (b) CDF.

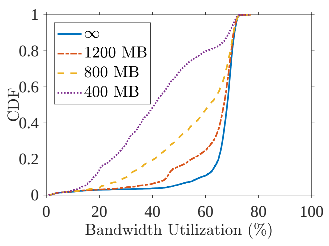


Figure 4.15: CDF of viewing quality across different viewers and videos.

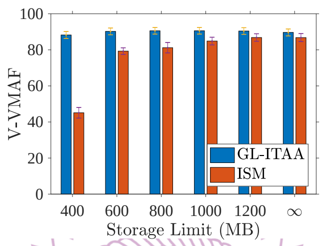


Figure 4.16: Viewing quality under different storage limits.

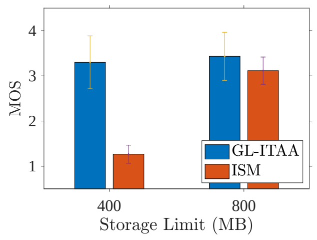


Figure 4.17: The MOS scores under different storage limits.

rest of the chapter.

We next plot the overall viewing quality of different video categories in Fig. 4.13. This figure shows the merits of our algorithms, over ICC and MM algorithms. It also shows that the videos from NI-SP have higher viewing quality for all algorithms, which is consistent with Fig. 4.11. We report the average and maximum improvements of our algorithms over the state-of-the-art algorithms in Table 4.4. This table illustrates that our PC-LBA averagely outperforms ICC and MM by 22.57 and 1.55 in V-VMAF. Moreover, PC-GBA outperforms ICC and MM by 23.85 and 2.83. In fact, the improvements of PC-GBA are as high as 52.17 and 26.35. From the figures, we observe that MM achieves reasonably good viewing quality. However, it only supports a single video *and* a single bandwidth class. In contrast, we also study the optimal laddering problem that takes client distribution and storage limit into consideration, which will be evaluated in Sec. 4.6.4.

PC-GBA achieves even better performance than PC-LBA. Our evaluation results show that PC-GBA offers better viewing quality than PC-LBA (Figs. 4.11–4.13). That is, simultaneously limiting multiple out-of-range QPs into the practical range seriously degrades the viewing quality. In contrast, PC-GBA iteratively and gradually adjusts the

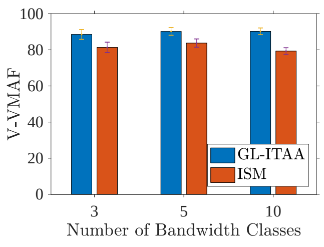


Figure 4.18: Viewing quality under different numbers of representations at storage limit 600 MB.

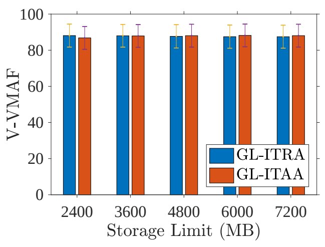


Figure 4.19: The viewing quality of GL-ITRA and GL-ITAA.

Table 4.6: The Ratio of the Meta-Data in DASH for Our Proposed Algorithm

Storage Limit	Meta-Data	Video Data
Unlimited	0.112%	99.888%
1200 MB	0.115%	99.885%
1000 MB	0.118%	99.882%
800 MB	0.123%	99.877%
600 MB	0.133%	99.867%
400 MB	0.152%	99.848%

QP values in a discrete manner, which finds better discrete solutions. We report their average computing time for each segment in Table 4.5. This table shows that PC-GBA runs faster, which is due to the much higher complexity of the Lagrangian multiplier approach in PC-LBA. Because of the better performance of PC-GBA compared to PC-LBA in terms of both viewing quality and computing time, *we adopt PC-GBA as the per-class optimization algorithm* in the rest of this chapter.

4.6.4 Optimal Laddering Results

We use our real testbed to evaluate our optimal laddering algorithms compared to the ISM algorithm. We throttle the network bandwidth of users following the distribution in Cisco's report [33]. Note that the ISM algorithm does not take video popularity into consideration, and thus we only compared it against GL-ITAA, where the storage space limit is evenly divided among all videos.

Our testbed effectively performs adaptive streaming over the network. Fig. 4.14(a) plots the bandwidth utilization across 10 users watching a sample video over time. This figure reveals that our testbed effectively runs the ABR algorithm at the client side, which achieves around 60% of bandwidth utilization after 4 seconds. Besides, Fig. 4.14(b) plots the CDF of the bandwidth utilization indicating that most of the bandwidth utilization is about 65%, which is quite close to the target 70% of the available bandwidth. Throughout the experiments, we observe no stall events.

Our proposed GL-ITAA algorithm outperforms the state-of-the-art algorithm. We then conduct experiments to evaluate the performance under different storage limits. Fig. 4.15 plots the CDF of the bandwidth utilization achieved by GL-ITAA across 10 users watching 6 videos under different storage limits. This figure shows that the smaller storage limits result in lower bitrates of some representations, which in turn lead to lower bandwidth utilization. This demonstrates the effectiveness of our GL-ITAA algorithm. We next compare the overall viewing quality with the ISM algorithm. Fig. 4.16 plots the viewing quality under different storage limits. This figure shows that our proposed GL-ITAA algorithm outperforms the ISM algorithm under all considered storage limits. Moreover, our algorithm outperforms the ISM algorithm by larger margins as the storage limit decreases. In particular, our GL-ITAA algorithm averagely outperforms the state-of-the-art ISM algorithm by 43.14 in V-VMAF when the storage limit is as low as 400 MB per video. This indicates that our GL-ITAA algorithm scales well under different storage limits. This can be attributed to the fact that our GL-ITAA algorithm effectively reduces the required storage by cutting the bitrate allocated to the less important tiles that are rarely viewed.

To confirm that our proposed algorithm outperforms ISM in terms of QoE, we have conducted a user study to quantify the real user experience. We randomly selected a user trace watching 6 videos and generated the viewport videos using the trace. We then recruited 12 subjects to watch these viewport videos in a random order and give overall quality scores from the [1,5] scale. Fig. 5.8 plots the MOS with 95% confidence intervals from different algorithms under 400 and 800 MB storage limits. In this figure, GL-ITAA outperforms ISM across all storage limits. Besides, the p-value from the ANOVA test is 0.0029 (< 0.05), which shows the statistical significance. As an example, GL-ITAA outperforms ISM by more than 2 points (out of a range of 4 points) in MOS at 400 MB storage limit. The above observations are consistent with our earlier observations on Fig. 4.16, which are made in V-VMAF. That is, we found that V-VMAF closely follows the user experience derived from time-consuming and expensive user studies. Hence, we use V-VMAF as the quality metric in the rest of this chapter.

We then consider $C \in \{3, 5, 10\}$, where the considered bandwidths are {3.12, 10.52, 35.52} Mbps and {3.12, 7.02, 15.78, 35.51, 79.91} Mbps for $C = 3$ and 5, respectively. We report the results across all videos and 10 users with storage limit $S = 600$ MB per video in Fig. 4.18. This figure shows the good scalability for our algorithm on different numbers of bandwidth classes. In particular, our GL-ITAA algorithm averagely outperforms the ISM algorithm by 8.2 in V-VMAF for different numbers of bandwidth classes. In summary, our GL-ITAA algorithm delivers higher viewing quality under various conditions. Next, we conducted experiments to see whether GL-ITRA can further improve the viewing quality.

GL-ITAA runs more efficiently while offering similar viewing quality compared to GL-ITRA. Last, we introduce diverse video popularity and compare the performance of GL-ITAA and GL-ITRA. In this experiment, we let each user watch 1 of the 6 considered videos according to the video popularity. Fig. 4.19 plots the viewing quality of both the GL-ITRA and GL-ITAA under different storage limits. This figure shows that the GL-ITRA and GL-ITAA achieve almost the same viewing quality regardless of the storage limits. This is because both algorithms effectively reduce the resource allocated to the tiles with lower viewing probability. However, the running time of the GL-ITRA is at least an order of magnitude longer than the GL-ITAA, while GL-ITAA averagely spends 39 minutes across different storage limits for each video. Moreover, it is suitable in practical usage scenarios, where new videos are gradually added to the streaming servers without: (i) *recomputing* the optimal encoding ladders and (ii) *re-encoding* the new representations of the existing videos. Last, GL-ITAA is easier to be parallelized and thus is more scalable. Hence, we recommend GL-ITAA for solving the optimal laddering problem.

Our proposed algorithms incur small meta-data overhead. Last, we measure the overhead of our proposed algorithms. In particular, our adopted per-tile-per-segment video models occupy averagely 53.33 KB storage space per 1-min long video. This is equivalent to about 0.013% overhead at 400 MB storage limit. We give the average ratio between the size of the meta-data and the total steamed data under different storage limits in Table 4.6. This table shows that the meta-data overhead increases as the storage limit decreases, but it never exceeds 0.2% of the total data size.

4.6.5 Summary of the Key Findings

The following summarizes the findings of the evaluation results.

- **Per-class optimization.** Our PC-LBA and PC-GBA algorithms optimize the viewing quality of the clients in the same bandwidth class. Our evaluation results show that PC-LBA and PC-GBA outperform ICC and MM by up to 52.17 and 26.35 in V-VMAF, respectively. We recommend PC-GBA for per-class optimization for its higher viewing quality and shorter running time.

- **Global optimization under assumptions.** Our GL-ITAA algorithm solves a simplified global optimization problem for optimal laddering. The goal is to optimize the overall viewing quality of the clients, where each video has a pre-determined storage limit. Our evaluation results show that GL-ITAA outperforms ISM by up to 43.14 in V-VMAF when the storage limit is 400 MB per video. Moreover, our GL-ITAA delivers better viewing quality and runs faster than ISM. Our GL-ITAA scales well in terms of both storage limits and number of bandwidth classes.

- **Global optimization.** Our GL-ITRA algorithm solves the most general optimal laddering problem, which jointly optimizes the overall viewing quality of the clients across multiple videos. To the best of our knowledge, none of the existing work in the literature addresses the same problem. While our results show that both GL-ITRA and GL-ITAA achieve high viewing quality, GL-ITAA runs much faster than GL-ITRA.

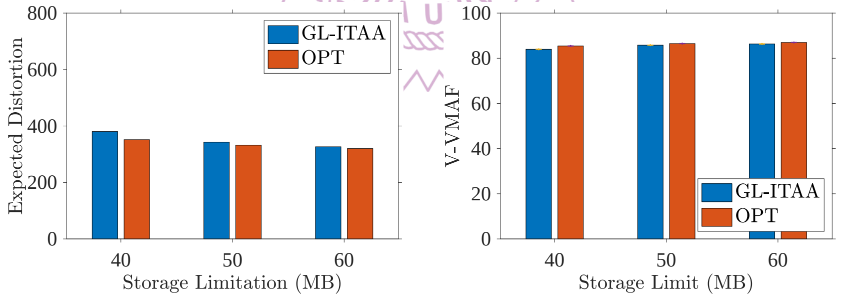
In summary, we recommend GL-ITAA and PC-GBA for solving the optimal laddering problem, which has not been rigorously solved in the literature.

4.7 Discussions

Next, we discuss the performance gap of our algorithms. We also consider an alternate objective function for better fairness among clients.

4.7.1 Comparisons with the Optimal Solution

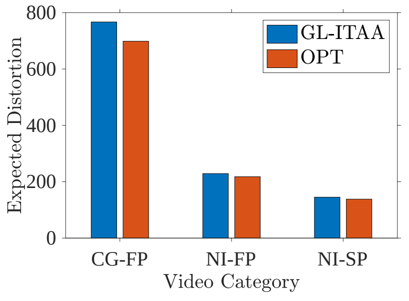
We compare our proposed algorithms to the optimal solution (OPT), where OPT directly solves the ILP problem in Eq. (4.3). We implement OPT using CPLEX. Because OPT may take a prohibitively long time to complete, we only consider smaller problems. In particular, we consider $C = 3$, where the bandwidths are {3.12, 10.52, 35.52} Mbps. We let $T = 15$, where each video plays for 15 seconds.



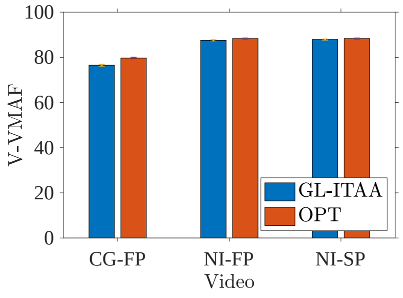
(a) (b)

Figure 4.20: The video quality under different storage limits: (a) expected distortion and (b) V-VMAF.

We first vary the storage limits $S = \{40, 50, 60\}$ MB to compare our proposed GL-ITAA against OPT algorithms. We compute the average performance across all 6 videos and report it along with 95% confidence intervals. Figs. 4.20(a) and 4.20(b) plot the expected distortion and V-VMAF under different storage limits, respectively. These figures show that V-VMAF generally follows the trends of the expected distortion. Besides, *GL-ITAA and OPT result in extremely close expected distortion and video quality*. In particular, the gaps between GL-ITAA and OPT are less than 1.5 in V-VMAF at most when



(a)



(b)

Figure 4.21: The video quality under different video categories: (a) expected distortion and (b) V-VMAF.

Table 4.7: The Ratio of The Running Time of GL-ITAA to OPT

Storage Limit	**Mega Coaster**	**Roller Coaster**	**Shark Shipwreck**	**Hog Rider**	**Chariot Race**	**SFR Sport**	**Average**
60 MB	6.10%	0.09%	0.60%	1.70%	5.21%	5.75%	3.24%
50 MB	3.61%	0.21%	0.58%	3.12%	7.28%	8.09%	3.83%
40 MB	4.62%	0.43%	0.06%	5.47%	4.00%	11.73%	4.39%

the storage limit is 40 MB. We further plot the results under different video categories in Fig. 4.21. This figure is consistent with the observations from Fig. 4.20: GL-ITAA achieves comparable video quality with OPT. Moreover, this figure indicates that videos from CG-FP have inferior video qualities in both the expected distortion and V-VMAF compared to other video categories. This may be attributed to the high texture complexity of the videos from CG-FP. However, the gap between GL-ITAA and OPT remains less than 3 in V-VMAF. Table 4.7 reports the ratio of the running time of GL-ITAA to that of OPT with different storage limits. This table shows the merits of our proposed algorithm: It achieves at least 97% of the viewing quality while consuming at most 11.73% of the running time compared to OPT. Note that as the problem size increases, the running time difference between GL-ITAA and OPT increases exponentially.

4.7.2 Fairness

In addition to minimizing the overall viewing distortion over a broad range of client classes, we discuss the fairness problem in this section. In particular, we consider the following two approaches for better fairness.

- **Max-min fairness** [18] is an objective to maximize the minimum allocated resource for any client class. In particular, it is achieved when any feasible allocation results in the resource decrease from other client classes with equal or smaller allocation.

- **Jain's fairness index** [91] is an index to rate the fairness of a set of values as $J(f_1, f_2, \cdots, f_N) = \frac{(\sum_{n=1}^{N} f_n)^2}{N\sum_{n=1}^{N} f_n^2} = \frac{1}{1+\widehat{\nu_f}^2}$, where $\widehat{\nu_f}$ is the coefficient of variation.

We apply the max-min fairness and Jain's fairness index to the distortion term of our formulation. We discuss the applicability of our algorithms to solve the revised formulations in the following.

1) Max-min Fairness

To apply the max-min fairness to our formulation in Eq. (4.3), we change the objective function Eq. (4.3a):

$$\min \max_{1\le c\le C, 1\le v\le V} D_{v,c}, \tag{4.13}$$

where $D_{v,c} = \sum_{v=1}^{V}\sum_{c=1}^{C} f_{v,c}\sum_{t=1}^{T}\sum_{n=1}^{N} p_{v,t,n}a_n\sum_{q=1}^{Q} d_{v,t,n}(q)x_{v,t,n,c,q}$. This revised objective minimizes the maximum distortion perceived by each client class c watching video v, where $1 \le c \le C$ and $1 \le v \le V$. We apply our divide-and-conquer approach on the revised formulation. Note that the revised formulation can be split into the same subproblems as out per-class optimization in Eq. (4.4), which minimizes the viewing distortion of each client class c watching video v under the bandwidth constraint b_c. In particular, the bandwidth of each class c is restricted to b_c, thus the minimum achievable distortion is restricted through the per-class optimization. Hence, our revised formulation can be solved by both the original PC-LBA and PC-GBA in the per-class optimization stage. Then, we slightly change the global optimization algorithms for the revised objective function Eq. (4.13) as follows. The revised global optimization algorithms iteratively increase the QP of the tile having the lowest $\epsilon_{v,t,n,c,q}$, where $(v, c) = \arg\min D_{v,c}$, by a step size δ until the consumed storage space is lower than the storage limit S. By doing so, our proposed revised global optimization algorithm tries not to increase the objective function value $\max_{1\le c\le C,1\le v\le V} D_{v,c}$ given by the per-class optimization algorithm while meeting the storage limit.

Fig. 4.22 shows the pseudo code of the algorithm revised from GL-ITRA to apply the max-min fairness. Lines 1–3 initialize Y^* and compile X^* using the PC-LBA or PC-GBA algorithms. Lines 4–6 set $y_{v,t,n,q}$ according to $x_{v,t,n,c,q}$. Line 7 initializes the current storage size S'. Lines 9–15 greedily select the tile to adjust its QP value iteratively until the required storage space S' reaches the storage limit S. In particular, line 10 determines the v^* and c^* for the quality fairness among client classes watching different videos. Lines 13–15 update X^* and Y^*, and the current required storage space. Line 16 returns the decisions X^* and Y^*.

```
1: Initialize $\mathbf{Y}^* = \{y_{v,t,n,q} = 0\}$
2: // Per-class optimization
3: $X^* = \{\mathbf{X}^*_{\mathbf{v},\mathbf{c}} | \forall v, c\} \leftarrow$ PC-LBA or PC-GBA
4: for $v \leftarrow 1$ to $V$, $t \leftarrow 1$ to $T$, $n \leftarrow 1$ to $N$,$q \leftarrow 1$ to $Q$ do
5:     if $\sum_{c=1}^{C} x_{v,t,n,c,q} \geq 1$ then
6:         $y_{v,t,n,q} \leftarrow 1$
7: $S' \leftarrow \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{n=1}^{T} \sum_{q=1}^{Q} r_{v,t,n}(q) y_{v,t,n,q}$
8: // Global optimization
9: while $S' > S$ do
10:     $(v^*, c^*) \leftarrow \arg\min_{1 \leq v \leq V, 1 \leq c \leq C} D_{v,c}$
11:     $\mathbf{E}_{\mathbf{v}^*,\mathbf{c}^*} \leftarrow \{\epsilon_{v^*,t,n,c^*,q} | t \in [1,T], n \in [1,N], q \in [1,Q]\}$ using Eq. (4.12)
12:     $(t^*, n^*, q^*) \leftarrow \arg\min E_{v^*,c^*}$
13:     $x_{v^*,t^*,n^*,c^*,q^*} \leftarrow 0$
14:     $x_{v^*,t^*,n^*,c^*,q^*+\delta} \leftarrow 1$
15:     Update $y_{v^*,t^*,n^*,q^*}$, $y_{v^*,t^*,n^*,q^*+\delta}$, and $S'$
16: return $\mathbf{X}^*$,$\mathbf{Y}^*$
```

Figure 4.22: The pseudocode of the revised GL-ITRA algorithm for max-min fairness.

2) Jain's Fariness Index

To apply the Jain's fairness index to our formulation in Eq. (4.3), we change the objective function Eq. (4.3a) to:

$$\max \frac{(\sum_{v=1}^{V} \sum_{c=1}^{C} D_{v,c})^2}{V \sum_{v=1}^{V} C \sum_{c=1}^{C} D_{v,c}^2} = \max \frac{1}{1 + \widehat{\nu_D}^2}, \tag{4.14}$$

where $\widehat{\nu_D}$ is the coefficient of variation of the set of $D_{v,c}$. This revised objective maximizes the Jain's fairness index for the perceived distortion of each client class c watching video v, where $1 \leq c \leq C$ and $1 \leq v \leq V$. We apply our divide-and-conquer approach on the revised formulation. The revised formulation can also be split into the same subproblems as our per-class optimization in Eq. (4.4), which minimizes the viewing distortion of each client class c watching video v under the bandwidth constraint b_c. Hence, our revised formulation can be solved by both the original PC-LBA and PC-GBA in the per-class optimization stage. Then, we slightly change the global optimization algorithm for the revised objective function in Eq. (4.14). In particular, our revised global optimization algorithm iteratively increases the QP of the tile having the lowest $\epsilon_{v,t,n,c,q}$, where $(v, c) = \arg\min D_{v,c}$, by a step size δ until the consumed storage space is lower than the storage limit S. By doing so, the revised global optimization algorithm always slightly increases the distortion of the class with the lowest distortion. This makes $\widehat{\nu_D}$ monotonically decreasing if and only if $\widehat{\nu_\delta} \leq \widehat{\nu_D}$, where $\widehat{\nu_\delta}$ is the resulting coefficient of variation after increasing the QP value of the selected tiled segment by δ. The pseudo code of the revised algorithm applying Jain's fairness index is basically the same as the one applying max-min fairness, which is given in Fig. 4.22.

4.8 Proofs of Lemmas

Lemma 1. *The optimal laddering problem is NP-hard.*

The optimal laddering problem can be reduced from the NP-hard Multiple Knapsack Problem (MKP). The MKP problem puts as many objects as possible into multiple knapsacks with various capacities to maximize the total value. If we let $V = 1$, $T = 1$, and $S = \infty$, we can map knapsacks to each class's available bandwidth and objects to tiled-segments without the storage limit. The value and the weight of each tiled-segment are the reciprocal of expected distortion ($\frac{1}{d_{v,t,n}p_{v,t,n}a_n}$) and bitrate ($r_{v,t,n}$), respectively. In this way, we reduce the MKP problem to our optimal laddering problem in polynomial time.

Lemma 2. *When the power function in Eq. (4.1) is adopted as the distortion model, the objective function in Eq. (4.5a) is convex.*

Proof. Note that a multivariate function is convex if it is twice differentiable and its Hessian matrix is positive semidefinite. We observe that $\alpha^d_{v,t,n} \geq 0$, $\beta^d_{v,t,n} \geq 1$, and $\gamma^d_{v,t,n} \geq 0$. We verify the second derivative of the objective function (Eq. (4.5a)) $\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}}$ as:

$$\alpha^d_{v,t,n}\beta^d_{v,t,n}(\beta^d_{v,t,n}-1)\kappa_{v,t,n,c}^{\beta^d_{v,t,n}-2}p_{v,t,n}a_n, \forall n \in [1, N] \tag{4.15}$$

The sphere area a_n is positive constant and viewing probability $p_{v,t,n}$ are non-negative constant. This shows that the objective function is second differentiable and $\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}} \geq 0$ according to the range of $\alpha^d_{v,t,n}$ and $\beta^d_{v,t,n}$. We then verify the Hessian matrix of the expected distortion:

$$\begin{aligned} H^D &= \begin{bmatrix} \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,1,c}\partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,1,c}\partial \kappa_{v,t,N,c}} \\ \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,2,c}\partial \kappa_{v,t,1,c}} & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,2,c}\partial \kappa_{v,t,N,c}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,N,c}\partial \kappa_{v,t,1,c}} & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,N,c}\partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & 0 & \cdots & 0 \\ 0 & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \end{bmatrix} \end{aligned} \tag{4.16}$$

We know that if the eigenvalues of a Hermitian matrix are non-negative, then the Hessian matrix is positive semidefinite. Let λ_D be the eigenvalue of H^D. Let y be a non-zero vector. According to the property of eigenvalue:

$$H^D y - \lambda_D y = 0; \tag{4.17a}$$

$$(H^D - \lambda_D I)(y) = 0. \tag{4.17b}$$

Note that y is a non-zero vector, which indicates that $H^D - \lambda_D I = 0$:

$$H^D - \lambda_D I = \begin{bmatrix} \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} - \lambda_D & 0 & \cdots & 0 \\ 0 & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} - \lambda_D & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} - \lambda_D \end{bmatrix} \tag{4.18a}$$

$$(\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} - \lambda_D)(\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} - \lambda_D)\cdots(\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} - \lambda_D) = 0 \tag{4.18b}$$

$$\lambda_D = \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}}, \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}}, \cdots, \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \tag{4.18c}$$

Since $\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}} = \frac{\partial d_{v,t,n}}{\partial^2 \kappa_{v,t,n,c}} p_{v,t,n} a_n \geq 0$, the eigenvalue of H^D are non-negative values and H^D is then proved to be positive semidefinite. This shows that our objective function (Eq. (4.5a)) is a convex function.

Lemma 3. *When the exponential function in Eq. (4.2) is adopted as the bitrate model, the constraint in Eq. (4.5b) is convex.*

Proof. Similar to Lemma 2, we first verify the second derivative for each κ_n in Eq. (4.5b):

$$\frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}} = \alpha^r_{v,t,n} (\beta^r_{v,t,n})^2 e^{\beta^r_{v,t,n} \kappa_{v,t,n,c}}, \forall n \in [1, N]. \tag{4.19}$$

Eq. (4.19) shows that the constraint function is second differentiable and $\frac{\partial E^R_{v,t,c}}{\partial \kappa^2_{v,t,n,c}} \geq 0$ according to the range of $\alpha^r_{v,t,n}$ and $\beta^r_{v,t,n}$, where $\alpha^r_{v,t,n} \geq 0$ and $\beta^r_{v,t,n} \leq 0$. We then verify the Hessian matrix of the constraint:

$$H^R = \begin{bmatrix} \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,1,c} \partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,1,c} \partial \kappa_{v,t,N,c}} \\ \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,2,c} \partial \kappa_{v,t,1,c}} & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,2,c} \partial \kappa_{v,t,N,c}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,N,c} \partial \kappa_{v,t,1,c}} & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,N,c} \partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & 0 & \cdots & 0 \\ 0 & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \end{bmatrix} \tag{4.20}$$

Let λ_R be the eigenvalue of H^R. We can then derive λ_R as:

$$\lambda_R = \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}}, \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}}, \cdots, \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \tag{4.21}$$

Thus, we found that the eigenvalue of H^R is non-negative since $\frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}}$ is non-negative values. This shows that the constraint function is a convex function as well.

Lemma 4. *The Lagrangian dual function (Eq. (4.7)) constitutes a lower bound for the objective value of any feasible solution to the Lagrangian primal problem (Eq. (4.6)). In fact, because the strong duality holds here, the optimal solution of the Lagrangian dual problem is also the optimal solution of the original problem.*

Proof. Let $\mathbf{K_p^*}$ be the optimal solution set for the primal problem. For any $\mu \geq 0$:

$$g(\mu) \geq \mathbf{K_p^*} \tag{4.22}$$

Suppose $\tilde{\mathbf{K_p}} = \{\kappa_{v,\tilde{t},1,c}, \kappa_{v,\tilde{t},2,c}, \cdots, \kappa_{v,\tilde{t},N,c}\}$ is a feasible solution for Eq. (4.6)). Then, we have

$$\mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,\tilde{t},n,c}) - b_{v,c})) \leq 0. \tag{4.23}$$

Eq. (4.23) shows the introduction of non-positive constraint. Therefore,

$$\begin{aligned} L(\tilde{\mathbf{K}}, \mu) &= \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,\tilde{t},n,c}) p_{v,t,n} s_x + \mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,\tilde{t},n,c}) - b_{v,c}) \\ &\leq \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,\tilde{t},n,c}) p_{v,t,n} a_n. \end{aligned} \tag{4.24}$$

Then, we can have $g(\mu) =$

$$\inf_{\mathbf{K}} L_c(\mathbf{K}, \mu) \leq L_c(\tilde{\mathbf{K}}, \mu) \leq \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,\tilde{t},n,c}) p_{v,t,n} a_n \tag{4.25}$$

Finding the best lower bound leads to the following optimization problem:

$$\max g(\mu) \tag{4.26a}$$

$$st: \mu \geq 0. \tag{4.26b}$$

We denote the optimal solution set of the Lagrangian dual problem as $\mathbf{K_d^*}$. Then we hold the following inequality:

$$\mathbf{K_d^*} \leq \mathbf{K_p^*}. \tag{4.27}$$

To hold the equality of Eq. (4.27), which indicates the optimal solution set of the dual problem is also the optimal solution set of the primal problem, we verify the strong duality. Since the primal problem is a convex problem, the equality condition holds if it satisfies Slater's condition: there exists a feasible $\tilde{\mathbf{K}} = \{\kappa_{v,\tilde{t},1,c}, \kappa_{v,\tilde{t},2,c}, \cdots, \kappa_{v,\tilde{t},N,c}\}$ such that:

$$\mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,\tilde{t},n,c}) - b_{v,c}) \leq 0. \tag{4.28}$$

holds. Let $r_{v,t,n}^{-1}$ be the inverse function of the $r_{v,t,n}$ function, which takes bitrate as input and outputs the corresponding QP. Let $r'_{v,t,n} = \frac{b_{v,c}}{N+\alpha}$, $\forall i$, where $\alpha \geq 0$. Then, the set $\mathbf{K} = \{r_{v,t,1}^{-1}(r'_{v,t,n}), r_{v,t,2}^{-1}(r'_{v,t,n}), \cdots, r_{v,t,N}^{-1}(r'_{v,t,n})\}$ is a feasible solution that holds the inequality. Therefore, we can solve the original distortion minimization problem by solving its Lagrangian dual problem (Eq. (4.7)).

Lemma 5. *The PC-LBA algorithm runs in time $O(T2^N)$ with space complexity of $O(N)$.*

Proof. The dominating time complexity occurs in lines 4 and 9: (i) line 4 solves Lagrangian equations using Newton's Method with $O(IN^3)$, where I is the iteration times in Newton's Method, and (ii) line 9 solves the ILP in $O(2^N)$. With T segments, the time complexity of the PC-LBA algorithm is $T \times O(IN^3 + 2^N) = O(T2^N)$. In addition, the space complexity is $O(N)$, as each of the N tiles records the selected QP value.

Lemma 6. *The PC-GBA runs in time $O(TN(\log N)Q)$ with space complexity of $O(N)$.*

Proof. The dominating time complexity occurs in lines 5–11: (i) the while-loop starts from line 5 iterates NQ times in the worst case and (ii) lines 7–8 update $\theta_{v,t,n,c}$ values and find out the maximum from them, which can be managed by a max heap with $O(\log N)$ time complexity. Accumulated with T segments, the time complexity of PC-GBA is $T \times O(N(\log N)Q) = O(TN(\log N)Q)$. Besides, the space complexity is N tiles recording the selected QP resulting in $O(N)$ for each segment.

Lemma 7. *The GL-ITRA runs in time $O(VTNC(\log VTNC)\frac{Q}{\delta})$ with space complexity of $O(VTNC)$.*

Proof. The dominating time complexity occurs in lines 9–14: (i) the while-loop starts from line 9 iterates $VTNC\frac{Q}{\delta}$ times in the worst case and (ii) lines 10–11 update $\epsilon_{v,t,n,c,q}$ values and find out the minimum from them, which can be managed by a min heap with $O(\log(VTNC))$ complexity. Collectively, the time complexity of the GL-ITRA algorithm is $O(VTNC(\log VTNC)\frac{Q}{\delta})$. Besides, the dominating space complexity consists of $VTNC$ QP values, which leads to $O(VTNC)$.

4.9 Conclusion

We study the optimal laddering problem for tiled 360° video streaming to HMD viewers. We consider video models, viewing probability, and client distribution to maximize the client viewing quality. We formulate the problem into an ILP problem and take a divide-and-conquer approach to solve the problem. In particular, we decompose it into: (i) per-class optimization for each bandwidth class and (ii) global optimization for the overall client viewing quality optimization. We have proposed two algorithms for each of the per-class optimization and global optimization problems. We have performed both analytical analysis and conducted experiments on a real testbed to quantify the performance of our algorithms compared to three state-of-the-art algorithms. We then recommend a combination of the proposed algorithms to solve the optimal laddering problem, which are the PC-GBA and GL-ITAA algorithms. The evaluation results show that our recommended algorithms outperform state-of-the-art algorithms by up to 52.17 and 26.35

in V-VMAF in per-class optimization problem and by up to 43.14 in global optimization problem. Moreover, our recommended algorithms scale well under different storage limits and run efficiently.

There are some limitations in this work:

- The selection of some parameters should be systematically done. For example, the considered bandwidth classes can be adaptively determined for further maximizing the client viewing quality.
- The considered tiling scheme is fixed, where the tiles are of equal size. However, different content may have higher compression efficiency with different tiling schemes, which can be intelligently determined.

Chapter 5

Consumption Optimization: QoE Modeling

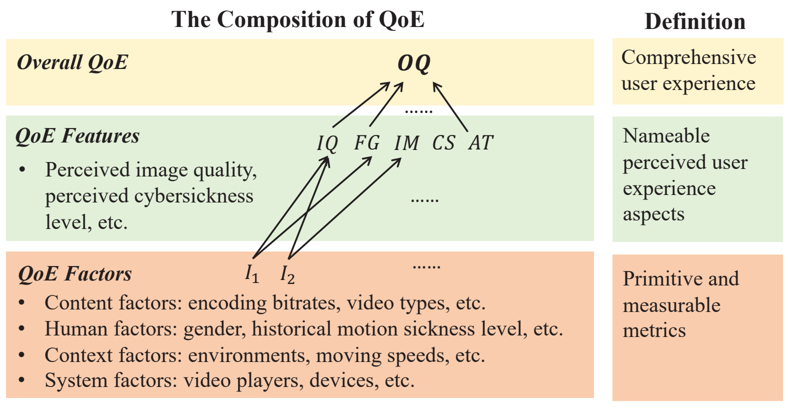

The Composition of QoE
Definition
Overall QoE
OQ
Comprehensive user experience
QoE Features
Perceived image quality, perceived cybersickness level, etc.
IQ FG IM CS AT
Nameable perceived user experience aspects
QoE Factors
I1 I2
Content factors: encoding bitrates, video types, etc.
Human factors: gender, historical motion sickness level, etc.
Context factors: environments, moving speeds, etc.
System factors: video players, devices, etc.
Primitive and measurable metrics

Figure 5.1: The overall QoE is a functions of QoE features and QoE factors. The symbols used in this figure are detailed in Sec. 5.1.1.

After solving the resource allocation problem at both the production and delivery phases, the key to developing a QoE-optimized 360° video streaming system is to derive the *QoE models*. While there have been user studies conducted with 360° videos [19, 57, 74, 184, 201] and tiled 360° videos [186], modeling the QoE for watching tiled 360° videos has not been thoroughly addressed. Hence, in this chapter, we tackle the problem of developing the QoE models for tiled 360° videos watched with HMDs. The compositions of the overall QoE [135, 202] are fairly complex, as illustrated in Fig. 5.1. The *overall QoE* is the comprehensive user experience perceived by subjects. The *QoE features*[1] are nameable user experience aspects that may contribute to the overall QoE, such as the perceived Image Quality (IQ) and CyberSickness (CS) level. The overall QoE and

[1]The QoE features are also known as the Key Quality Indicators (KQIs) perceived by the users in some work in the literature, such as Varela et al. [202].

QoE features can be considered as a *function* of multiple QoE factors [175], where the QoE factors (I_1, I_2, $\cdots$ in Fig. 5.1) are primitive and measurable metrics. These QoE factors can be classified into four categories [22]: (i) *content factors*, which determine the reconstructed content quality, such as the encoding bitrate, (ii) *human factors*, which capture human characteristics, such as his/her historical motion sickness frequency, (iii) *context factors*, which describe the user's environments or interactions [97, 138, 170], such as head rotation speed, and (iv) *system factors*, which are related to hardware or applications, such as the video player. More complete lists of QoE features and factors are introduced in Sec. 5.1.

Modeling the QoE of watching tiled 360° videos using HMDs is quite difficult because:

- Different from conventional videos, the behavior of individual viewers watching 360° videos could be quite different. This makes the viewing regions totally different among individual viewers, even for the same video. More precisely, the uncontrolled environments of user studies make viewer feedback collection and user experience modeling more challenging.
- There are plenty of QoE factors contributing to the QoE features and overall QoE. The dominating QoE factors affecting: (i) overall QoE and (ii) QoE features of watching tiled 360° videos are still unknown. Hence, it is quite challenging to model the relationship among the overall QoE, QoE features, and QoE factors.

To address these challenges, we first identified the potential QoE features and factors that may affect the user experience of watching tiled 360° videos. Then, we designed and conducted a user study to collect and investigate the QoE scores with the considered QoE factors and features. We then derived the models for overall QoE and QoE features in terms of both *MOS* and *IS*. Here, MOS [181] refers to the *average* quality scores of *a set of subjects* rating an experience of watching tiled 360° videos using HMDs. In contrast, IS refers to the quality scores given by *each* subject on his/her own experience, which is reported to be more difficult to model [103]. Last, the dominating factor category and factor of individual models are identified through our in-depth analysis. In this chapter, we have built 5 (1 overall QoE + 4 QoE features[2].) $\times$ 2 (MOS and IS) = 10 QoE models. In contrast, existing studies [41, 42, 225] focus on a single MOS model (overall QoE), which is only one-tenth of our outcomes. Besides, most studies only consider the content factors [41, 225]. Croci et al. [42] further consider the quality metrics in the viewer's viewport, which are some context factors. Different from their work [41, 42, 225], we consider comprehensive sets of content and context factors, including quality variance

[2]One of the QoE features cannot be modeled, which is detailed in Sec. 5.3.2

and head rotation speed. More importantly, our work further incorporates human factors, which, to the best of our knowledge, have never been considered in the literature.

5.1 QoE of 360° Tiled Videos Streamed to HMDs

We introduce the lists of QoE features and factors in this section. We note that while we try to make the lists as complete as we can, readers certainly may come up with additional QoE features and factors. This is not a concern, as our proposed user study design and modeling approach are also applicable to other lists of QoE features and factors.

5.1.1 QoE Features

Table 5.1: Questionnaire of Overall QoE and QoE Features in Our User Study

Acronym	**Question**	**Lowest Score (1)**	**Highest Score (9)**	**Ref.**
OQ	How would you rate the overall quality?	Awful	Excellent	-
IQ	How would you rate the image quality?	Awful	Excellent	[100]
FG	How would you rate the fragmentation level?	None	Severe	[135, 197]
IM	How would you rate the immersion level?	Awful	Excellent	[193]
CS	How would you rate the perceived cybersickness level?	None	Severe	[99, 193]
AT	How would you rate the attractiveness level?	Not Attractive	Attractive	[22]

We identify possible QoE features of watching tiled 360° videos with HMDs by surveying the QoE literature of conventional and 360° videos. Table 5.1 summarizes all of the QoE features. Each QoE feature is associated with a question in the questionnaire used for our user study. The first question is the Overall QoE (OQ) of watching the 360° videos. We describe the considered QoE features in the following:

- **Image Quality (IQ)** is the visual clarity level of video frames.
- **FraGmentation (FG)** is the level of artifacts due to coding and tiling. We noticed that the blocking artifacts caused by tiling are known in the literature [208], and are often mentioned by our subjects.
- **IMmersion (IM)** is the perceived level of being physically present in the video.
- **CyberSickness (CS)** is the experienced cybersickness level when watching the video, which may result in dizziness and nausea.

- **ATtractiveness (AT)** is the level of the video content attracting the subject. We ask the subjects to focus on the attractiveness from the video content instead of the video quality. Each subject only rates each video once.

We note that lower FG and CS scores indicate better user experience, while higher scores of all other QoE features and overall QoE indicate better user experience.

5.1.2 QoE Factors

Table 5.2: The Considered QoE Factors

Cat.	Factor	Definition	Intuition
Content	*Bitrate*	The total encoding bitrate	Different encoding bitrates cause different levels of distortion
	Complexity	SI/TI quantify the spatial/temporal complexity [87]	Scene complexity captures the video characteristics
	Motion	The sum of the optical flow [126] from pixels of two adjacent frames	Higher motion level may cause more cybersickness
	Video Quality	Average PSNR/SSIM/VMAF	Objective metrics capture the distortion level
	Video Quality Variance	Standard deviation (std.) of PSNR/SSIM/VMAF among video tiles	Quality variance among tiles may lead to unpleasant artifacts
Human	*Gender*	The gender of the subject	Female subjects may be more vulnerable to cybersickness [185]
	Historical Motion Sickness	The motion sickness frequency and level in the past	Subjects who suffer from more motion sickness may be more vulnerable to motion sickness
	Avg. Head/Gaze Rotation Speed	The head/gaze rotation speed (in *yaw* and *pitch*) of a subject	The rotation speed may influence a subject's sensitivity to content
Context	*Head/Gaze Rotation Speed*	The head/gaze rotation speed (in *yaw* and *pitch*)	The rotation speed may influence a subject's sensitivity to content
	Gaze Complexity	TI/SI weighted by the fraction of the gaze region	The complexity of the gaze region may influence the perception
	Gaze Motion	Optical flow weighted by the fraction of the gaze region	The motion level of the gaze region may influence the perception
	Gaze Video Quality	PSNR/SSIM/VMAF weighted by the fraction of the gaze region	Objective metrics capture the distortion level of the gaze region
	Gaze Quality Variance	Std. of PSNR/SSIM/VMAF among the tiles in the gaze region	Different tile qualities in the gaze region may cause an unpleasant experience

We pick and summarize some crucial QoE factors of watching tiled 360° videos in the content, human, and context categories[3] in Table 5.2. Some content factors are fairly

[3]We focus on a 360° video streaming system implemented by us, which runs on a given hardware platform to make the size of our user study manageable.

popular in the literature, such as video complexity in Temporal Information (TI) and Spatial Information (SI) [87], and the video quality in PSNR, SSIM [29], and VMAF [142]. In addition to the content factors, the human and context factors capture the difference among individual subjects and behaviors, respectively. For the human factors, the historical motion sickness experience may be used to predict whether the subject perceived cybersickness easily. Following the literature [97, 138, 170, 175], we classify the interactions between content and subjects, such as the head/gaze rotation speed or the content within the gaze region, into context factors. In particular, different head/gaze rotation speeds may cause different degrees of sensitivity to the watched content. Here, *gaze region* refers to a small region surrounding the subject's gaze; we adopt a gaze region of $30^\circ \times 30^\circ$ [212, 213] throughout the chapter. Alternate gaze region, e.g., defined by gaze exploring probability [172], can also be adopted. Note that, because each subject watches totally different content, we define several *gaze factors* (such as gaze PSNR) as weighted sums of *factors* (such as PSNR), where the weights are the fractions of tiles within the gaze region. To capture the negative impacts of tiling, gaze quality variance accounts for the tile quality variance in the gaze region.

5.2 A User Study

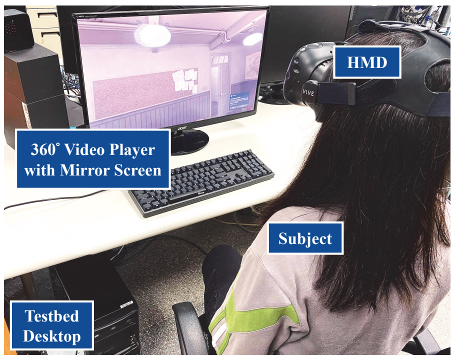
To the best of our knowledge, there is no public dataset on the relation among overall QoE, QoE features, and QoE factors for watching tiled 360° videos in HMDs. Hence, we conducted our own user study in this section.

5.2.1 Testbed

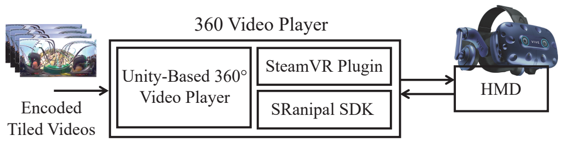
We developed a 360° video player for the latest eye-tracking enabled HTC Vive Pro Eye [77]. Fig. 5.2(a) shows a photo of a subject using the testbed. Fig. 5.2(b) illustrates the testbed architecture. The key entity of the testbed is the *tiled 360° video player*, which is developed using Unity [198] with SteamVR plugin [199] and SRanipal SDK [76] for the HMDs and eye-tracking supports, respectively. When subjects watch 360° videos, the 360° video player saves both the eye gazes and head orientations in log files for further analysis. Fig. 5.2(c) shows our sample log files.

5.2.2 Dataset and Subjects

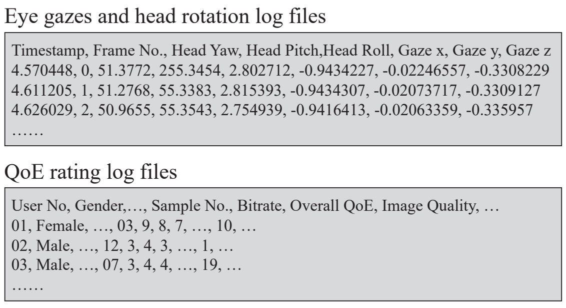
We selected six production-quality 360° videos provided by Joint Video Exploration Team (JVET) [85] for the user study. These videos are summarized in Table 5.3. We transcoded all test videos to 4K (3840×1920) resolution at 30 frame-per-second (fps) and reprojected



(a)



(b)



Eye gazes and head rotation log files

```
Timestamp, Frame No., Head Yaw, Head Pitch,Head Roll, Gaze x, Gaze y, Gaze z
4.570448, 0, 51.3772, 255.3454, 2.802712, -0.9434227, -0.02246557, -0.3308229
4.611205, 1, 51.2768, 55.3383, 2.815393, -0.9434307, -0.02073717, -0.3309127
4.626029, 2, 50.9655, 55.3543, 2.754939, -0.9416413, -0.02063359, -0.335957
……
```

QoE rating log files

```
User No, Gender,…, Sample No., Bitrate, Overall QoE, Image Quality, …
01, Female, …, 03, 9, 8, 7, …, 10, …
02, Male, …, 12, 3, 4, 3, …, 1, …
03, Male, …, 07, 3, 4, 4, …, 19, …
……
```

(c)

Figure 5.2: Our testbed for the user study: (a) a photo of a subject using our testbed, (b) the testbed architecture, and (c) the sample log files.

Table 5.3: The Considered Test Videos from JVET

Class	**Video**	**Resolution**	**Frame Rate**	**Duration**
Fixed Camera	SkateboardTrick (ST)	8192×4096	60 fps	10 sec
	Harbor (HB)	8192×4096	30 fps	10 sec
	PoleVault (PV)	3840×1920	30 fps	10 sec
Moving Camera	Landing (LD)	6144×3072	30 fps	10 sec
	Balboa (BB)	6144×3072	60 fps	10 sec
	BranCastle (BC)	6144×3072	30 fps	10 sec

them into the equi-angular cubic projection[4] using 360lib [84] for a uniform pixel density. Fig. 5.3 plots the complexity of the test videos, where the SI ranges from 36 to 66 and the TI [87] ranges from 2 to 27. This figure shows that our selected test videos have diverse characteristics. We increased the video playout length from 10 to 20 seconds by reversely playing each video after reaching its end. We encoded each video into 12×8 tiles [134] with 1-sec segments using Kvazaar [204] at six video bitrates: 1, 3, 6, 9, 12, and 15 Mbps[5] resulting in 6 *Hypothetical Reference Circuits* [90]. Thus, there are 36 (6×6) *Processed Video Sequences* [88] referred to as test videos in the rest of the chapter for each subject. We opted not to vary the bitrates across the tiles in each processed video sequence for two reasons: (i) diverse tile quality is known to dramatically degrade the QoE [208] and (ii) too many test videos would lead to fatigue of the subjects [89]. Note that although we use the same tile bitrates in each test video, the impacts from the Motion-Constrained

[4]We use an open-source shader [196] to support the equi-angular projection in Unity.

[5]We stop at 15 Mbps because the PSNR values of videos barely increase after that.

Tile Sets (MCTS) features employed by the tile codecs is captured and investigated in our user study. We recruited 24 subjects, and 13 of them are males. The subjects are between 19 and 30 years old[6]. Table 5.4 reports the demography of the subjects. In total, we spent nine days conducting the user study and collected 864 questionnaires.

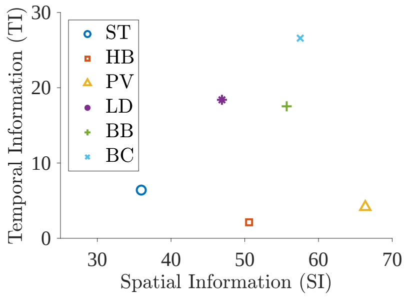


Figure 5.3: Our considered videos have scattered SI and TI values.

Table 5.4: The Demography of Subjects

Gender	Male: 58%, Female: 42%
Age	Range: [19,30], Standard Deviation: 2.78
HMD Experience	Never: 4%, Seldom: 79%, Medium: 17%
Vision Correction	Glasses: 13%, Contacts: 75%, None: 12%
Education	High School: 37%, Bachelor: 42%, Master: 21%

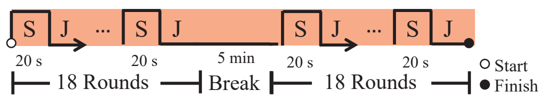
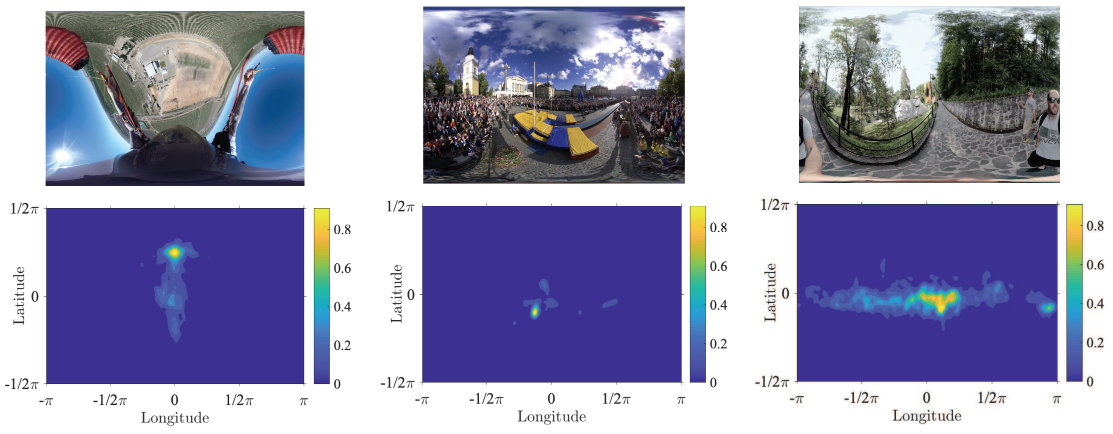


Figure 5.4: Overview of the user study for each subject. S and J represent the stimulation and judgment phases.

5.2.3 Procedure

Our user study follows the recommendations of ITU-T 910 [88]. We employed a rating scale of [1,9] for higher discriminative power [88]. Before the user study, each subject needed to pass a vision test (20/25 in Snellen chart) and watch a random 360° video to familiarize himself/herself with our system. This was followed by the calibration of our eye-tracker. After that, subjects started the user study with the Absolute Category Rating (ACR) procedure. Fig. 5.4 illustrates the process of the user study for each subject, which consists of *36 rounds* in total. Each round contains a *stimulation* and a *judgement phases*. We randomly played a test video at each stimulation phase and asked subjects to rate the overall QoE and QoE features of the 360° video in the judgement phase. In the stimulation phase, subjects sat on swivel chairs when watching the 360° videos and they could freely rotate their heads and bodies. The list of questions used in the judgment phase is given in Table 5.1. Besides, the scores were asked and recorded by an assistant, so that subjects did not need to remove the HMD to complete the questionnaire. The user study took each subject 50 minutes to complete. Each subject had a 5-min break after completing half of the test videos.

[6]Older people are reported to have more severe perceived sickness using HMDs [68] and thus are likely not to watch 360° videos using HMDs. Hence, we did not include older people as our subjects.



(a) (b) (c)

Figure 5.5: Sample video frames from three sample videos and their gaze-level viewing heat maps from all subjects: (a) Landing (LD), (b) PoleVault (PV), and (c) BranCastle (BC). The sample video frames are in equirectangular projection for the sake of presentation.

5.2.4 Viewing Behaviors and Video Classification

Fig. 5.5 shows the sample frames and their corresponding gaze-level viewing heat maps, where a visual angle of 3° centered at each gaze point is used to compensate for the tracking error. In these figures, viewers' gaze levels are normalized to [0, 1]. The higher values indicate higher viewing frequency and vice versa. Fig. 5.5(a) shows that most subjects watched Landing staring ahead. Based on the inputs of the open question in the questionnaire, this can be attributed to two reasons. Some subjects were immersed in the virtual world like they were really paragliding. Others were afraid that rotating their heads may make them even dizzier. Fig. 5.5(b) shows that when watching video PoleVault, most gazes lie on the mattress. This is because most subjects tend to track the athlete in the video and the athlete spends a long time on the mattress after his attempt. Fig. 5.5(c) shows that the viewing directions when watching BranCastle are rather dispersed. This is because there is no main object or target in the video for the subjects to track or observe, which makes them tend to explore around the whole video. Fig. 5.5 reveals that the 360° video content affects the overall viewing behavior.

Fig. 5.6 plots the subject rotation speed on each video at both gaze- and head-level. This figure shows that the subjects tended to move their eye gazes faster than their head. Moreover, the rotation speed at both levels was inline with our observations, as shown in Fig. 5.5: the subjects watching LD and PV have lower rotation speed and less diverse gaze. Based on the observations, we classified these videos according to camera moving directions, Fixed Camera (FC) and Moving Camera (MV), and attention properties, Fixed

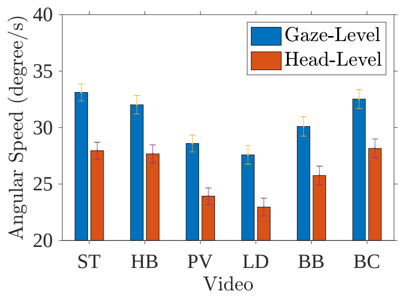


Figure 5.6: The average rotation speed of subjects under different videos.

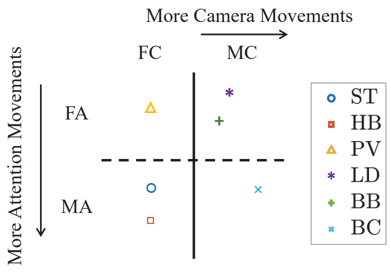


Figure 5.7: The classifications of videos according to camera and attention properties.

Attention (FA) and Moving Attention (MA). Fig. 5.7 illustrates the video classifications, where the videos with lower rotation speed in Fig. 5.6 are classified into FA. In MC class, the closer to the right, the higher camera moving speed is the video. Similarly, the closer to the bottom, the higher attention moving speed is the video, which can be seen in Fig. 5.6.

5.2.5 The Overall QoE and QoE Features

We normalized the opinion scores within each subject using Z-scores, which work as follows. Let $m_{u,s}$ be the s-th rating score from subject u. The corresponding Z-score $z_{u,s}$ is written as $z_{u,s} = \frac{m_{u,s}-\bar{m}_u}{\sigma_u}$, where $\bar{m}_u$ and σ_u denote the mean and standard deviation of all subject u's rating scores [25]. We consider the scores outside of [-3, 3] as outliers [182]. After normalization, we map the Z-score $m'_{u,s}$ back to [1,9] using the equation: $m'_{u,s} = 9\times(z_{u,s}+3)/6$. We plot the resulting MOS of overall QoE and the considered QoE features over different encoding bitrates in Figs. 5.8(a)–5.8(e) with 95% confidence intervals. Fig. 5.8(f) shows the attractive levels of individual videos. Fig. 5.8(a) shows that the overall QoE in MOS scores improve by 50% when the bitrate is increased from 1 to 15 Mbps. Besides, the OQ, IQ, and IM (Figs. 5.8(a), 5.8(b), and 5.8(d)) are increasing functions on the bitrate. In contrast, the FG (Fig. 5.8(c)) is a decreasing function on the bitrate. For the CS (Fig. 5.8(e)), we see that it is influenced more by the video content rather than by the bitrate. Cross-referencing Fig. 5.3 reveals that the CS increases for videos with higher TI. This is because the subjects usually suffer from higher CS level if they are not aware of the moving direction of the video camera in advance [103]. An interesting finding is that AT also affects the overall QoE. For example, video ST has higher IQ, lower FG, and even lower CS compared to video LD. However, the overall QoE of ST and LD are mostly overlapping and sometimes, LD even outperforms ST (e.g., at 1 and 15 Mbps). This may be attributed to the higher attractive level of video LD.

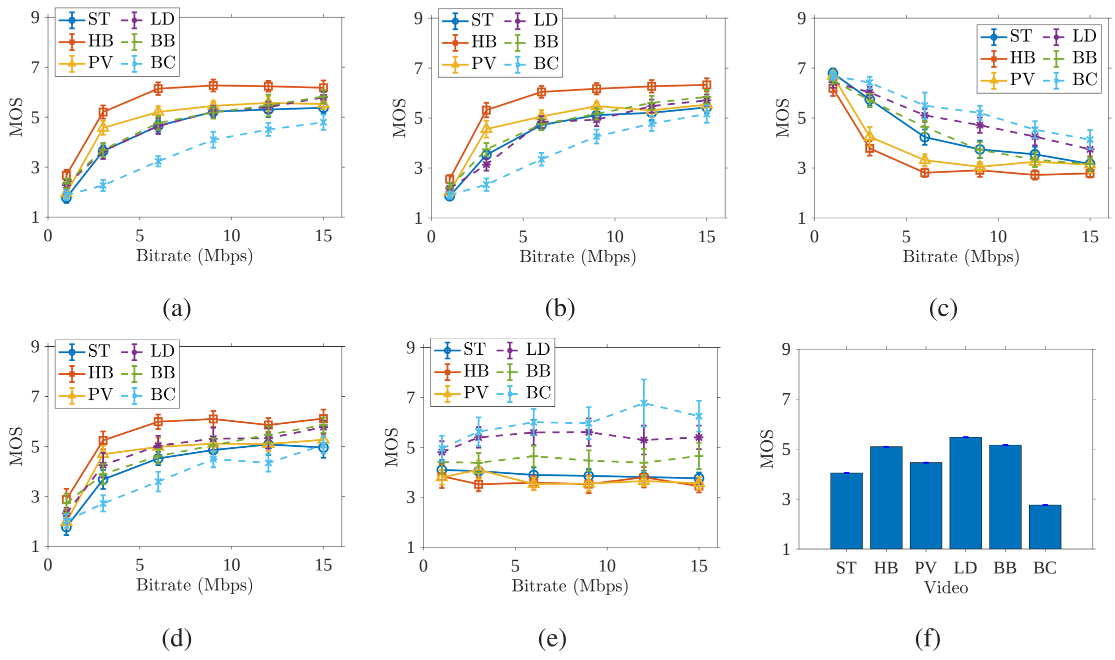


Figure 5.8: The MOS scores at different bitrates: (a) Overall QoE, (b) IQ, (c) FG, (d) IM, (e) CS, and from different videos: (f) AT.

5.2.6 Diverse QoE Models

We built the MOS models for the overall QoE and QoE features other than AT. This is because we only collect AT rating per video. We built models for both the MOS and IS scores with different *sets* of factors: (i) all factors, (ii) a dominating factor category, and (iii) a single dominating factor. The *dominating* factor (category) is the factor (category) that affects the modeling performance the most. We refer to these models as OQ_L^T, IQ_L^T, FG_L^T, IM_L^T, and CS_L^T for the overall QoE, IQ, FG, IM, and CS, respectively, where $T \in \{M, I\}$ for the MOS or IS scores and $L \in \{A, C, F\}$ to indicate the factor sets of all factors (A), the dominating factor category (C), and the dominating factor (F). The models with different factor sets can be used in different 360° video streaming scenarios. For example, the systems with limited resources are more suitable to adopt the models with fewer factors, such as the models with only a single dominating factor. In contrast, when collecting more factors is feasible, models with all factors generally lead to better performance.

5.3 MOS Modeling

In this section, we apply multiple regressors to model the MOS scores for the overall QoE and QoE features.

5.3.1 Regressor Selection

We consider the following widely-adopted regressors:

- **Linear regressor [59]** employs a few parameters and is less vulnerable to overfitting.
- **Random forest regressor [117]** is an ensemble method consisting of multiple decision trees. It is good at processing a large amount of inputs and is efficient on learning. It comes with a few hyper-parameters. For example, the maximum number of features restricts the number of features considered at each branch split of the decision trees.
- **Gradient boosting regressor [60]** is also an ensemble method which employs stage-wise learning. It reduces the complexity of each decision tree to avoid overfitting by, for example, adding costs to complex tree structures.
- **Support-vector regressor [14]** finds the hyperplane for predicting the data distribution, where a tolerance level ϵ is specified as a hyper-parameter. Higher ϵ values may lead to higher training loss, while lower ϵ values may increase the probability of overfitting.

Table 5.5: Performance of Regressors: Sample Results from OQ_A^M

Regressor	**Hyper-Parameters**			**Training Set**		**Validation Set**	
				PLCC	**SROCC**	**PLCC**	**SROCC**
Linear	-			*0.9925*	*0.9823*	*0.9518*	*0.9175*
Random Forest	**Max No. Features**	**No. Estimators**	**Max Depth**	0.9686	0.9501	0.9215	0.8541
	30	200	8				
Gradient Boosting	**Max No. Features**	**No. Estimators**	**Learning Rate**	0.9934	0.9761	0.9451	0.8962
	5	100	0.01				
Support Vector	**Tolerance Level**	**Max Iterations**	**Penalty**	0.9880	0.9730	0.9350	0.9021
	0.05	20	10				

We implemented the considered regressors using Scikit-Learn [163]. We used the dataset collected from the user study (Sec. 5.2) to derive and evaluate models. We split the dataset by *subjects*, where the training set and testing set accounted for samples from 70% and 30% of subjects, respectively. Besides, 20% of the training set was used as the validation set for a 5-fold cross validation. We opted to isolate the training and testing datasets on subjects (instead of videos) to be closer to the real usage scenarios, where the number of viewers is much larger than the number of videos. Take the Video-on-Demand (VoD) scenario as an example; a single video will be viewed by tens of thousands, if nor more, customers. Hence, the MOS or IS scores of a customer watching the current video can be predicted using the models derived from the inputs from other customers who have watched the same video. In VoD scenarios, prior inputs of a viewer when watching other videos could be used to predict his/her future QoE, which is not done in this work since inputs from different subjects are isolated. Alleviating this constraint will further improve the performance of our QoE models.

We grid search on the best hyper-parameters of individual regressors using the validation set. In particular, we compute PLCC and Spearman Rank Order Correlation Coefficient (SROCC) [190] to evaluate the correlation between the MOS scores from our models and the subjects. Table 5.5 gives the sample results for the OQ_A^M from each regressor. The italic font indicates the best results among different regressors. *The results show that the linear regressor achieves the highest PLCC and SROCC scores.* We therefore adopt it as our regressor for all models in the rest of this chapter.

5.3.2 Derived Model Performance

In this section, we evaluate our models using the testing set. Besides, we compare the performance with the state-of-the-art QoE models, which are listed below.

- Yao et al. [225] developed a QoE model for 360° videos considering QP, SI, and TI.
- VI-VMAF [41] is the state-of-the-art objective quality metric designed for 360° videos, which is reported to offer better correlation with the overall QoE for watching 360° videos. We used their binary tool with the best parameters they reported in their paper to calculate the VI-VMAF values.
- VI-VA-VMAF [42] is an extension of VI-VMAF, which employs the visual attention map [93] as the weights when computing the final values. The visual attention map is generated using the subject's gazes.

For fair comparisons, we used MATLAB [131] to (re)train the above models using our training set. Among them, VI-VMAF [41] and VI-VA-VMAF [42] adopt a complicated logistic function [86] without specifying how to select the starting points. Hence, we repeated each training 10 times with random starting points and chose the best parameters that lead to the highest adjusted R^2 value. We make the following observations.

Our derived MOS models achieve high correlation. We first consider all content, human, and context factors for MOS modeling. Fig. 5.9 plots the predicted versus collected MOS scores for OQ_A^M. This figure shows that the predicted MOS scores from OQ_A^M closely follow the collected MOS scores. We also observe similar trends on other QoE features; figures are omitted for brevity. We plot the CDF of the Absolute Errors

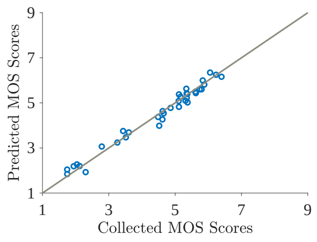


Figure 5.9: The predicted versus collected MOS scores of the overall QoE.

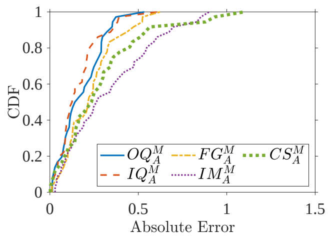


Figure 5.10: The CDF curves of the AE for the MOS scores. We zoom into [0,1.5], where the range is [1,8].

Table 5.6: Performance of Derived MOS Models with All Factors

Model	PLCC	SROCC	MAE	RMSE
$\mathbf{OQ_A^M}$	0.988	0.971	0.180	0.218
$\mathbf{IQ_A^M}$	0.989	0.977	0.165	0.208
$\mathbf{FG_A^M}$	0.980	0.975	0.233	0.277
$\mathbf{IM_A^M}$	0.944	0.889	0.342	0.422
$\mathbf{CS_A^M}$	0.908	0.902	0.293	0.389

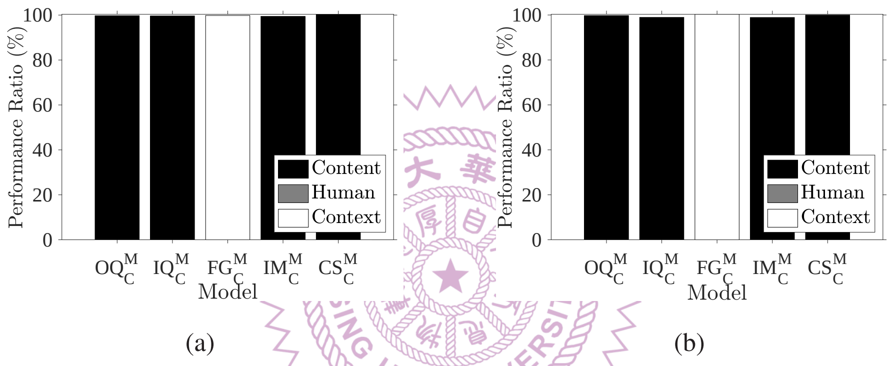
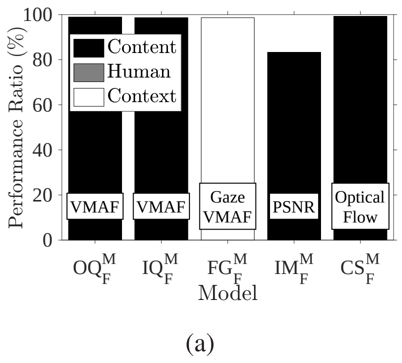


Figure 5.11: The performance ratio achieved by our derived MOS models using factors from the dominating category: (a) PLCC and (b) SROCC.

(AE) of each predicted MOS scores compared to the collected ones in Fig. 5.10. This figure shows that the maximum AE is about 1, which is relatively small compared to the maximal QoE score of 9. For completeness, we report the PLCC, SROCC, Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) of our derived MOS models in Table 5.6. Among them, higher PLCC and SROCC indicate better model performance. In contrast, lower MAE and RMSE indicate better model performance. This table shows that our models perform quite well. For example, our derived model for the overall QoE achieves 0.988 and 0.971 in PLCC and SROCC, respectively. In terms of the QoE features, our derived models lead to 0.908 to 0.989 in PLCC and 0.889 to 0.977 in SROCC. Besides, the MAE (RMSE) of the overall QoE and QoE features is always below 0.342 (0.422).

Content category dominates the factor categories. We investigated the most representative factor category of our derived models as follows. We iteratively identified the category with the least influence, and removed that factor category. The last fac-



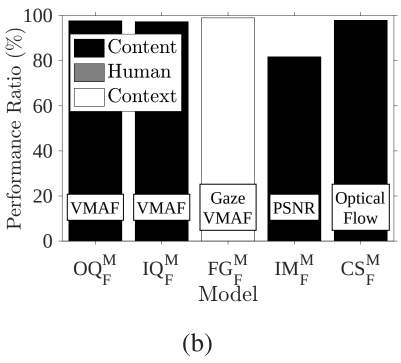


Figure 5.12: The performance ratio achieved by our derived MOS models using the dominating factor: (a) PLCC and (b) SROCC.

tor category is the dominating category. Fig 5.11 plots the *performance ratio* between MOS models with factors in the dominating category and all categories, where the performance ratio is the performance normalized to the one achieved by the model with all factors. This figure shows that the models with the dominating categories achieve $> 98\%$ of performance ratio in both PLCC and SROCC. Furthermore, the dominating category is *content*, except for FG. In contrast, FG depends more on *context* factors, e.g., what subjects see in their gaze regions.

Video quality dominates the factors. We zoom into the *dominating factor* within the dominating category, and compute the performance ratio between MOS models with the dominating factor and all factors. Fig. 5.12 plots the performance ratio, and the annotations indicate the dominating factors. This figure shows that the video quality is the dominating factor for most models. In particular, (gaze) VMAF is the dominating factor of the models for the overall QoE, IQ, and FG, which achieves $> 97\%$ performance ratio in both PLCC and SROCC. Besides, the model for CS is dominated by the optical flow, which also has $> 97\%$ performance ratio in both PLCC and SROCC. With a dominating factor (PSNR), our model for IM only achieves $\sim 82\%$ performance ratio. This indicates that immersive experience requires more factors for better modeling performance. Cross-referencing Fig. 5.11, a QoE model with all content factors is more suitable for the IM: 99% versus 82%.

Comparing the state-of-the-art objective quality metrics and QoE models. Fig. 5.13 plots the predicted versus collected MOS scores of our overall QoE model, compared to the state-of-the-art objective quality metrics [41, 42] and model [225]. This figure clearly shows that OQ_A^M achieves high linear correlation with the collected MOS scores. Among other metrics and models, Yao et al. [225] leads to the most deviated results. VI-VMAF [41] and VI-VA-VMAF [42] result in similar deviation levels, which are between Yao et al. [225] and OQ_A^M. Fig. 5.14 further plots the sample performance ratios among our derived models for the overall QoE (OQ_A^M, OQ_C^M, and OQ_F^M) and the state-of-the-art metrics and models, where our model with all factors (OQ_A^M) is used for normalization. This figure shows that Yao et al. [225] achieve the worst performance in terms of both PLCC and SROCC. On the other hand, the performance ratios of VI-VMAF fall between our model using the dominating factor (OQ_F^M) and the dominating factor category (OQ_C^M) in terms of both PLCC and SROCC. VI-VA-VMAF is, however, slightly inferior to VI-VMAF, although the difference is extremely small. Based on these observations, we recommend OQ_A^M as the QoE model if additional human and context factors can be acquired. Otherwise, OQ_C^M is recommended for the overall QoE if all content factors are available. If only a single factor is considered, OQ_F^M and VI-VMAF can be considered; between them VI-VMAF achieves a slightly higher performance ratio (∼0.5%). Last, we emphasize that we can *only* compare the other metrics [41, 42] and model [225] in the literature against the overall QoE models for MOS scores. This is because their studies do not build QoE models for other QoE features (IQ, FG, IM CS). Indeed, the current chapter is, to the best of our knowledge, the *first* comprehensive study trying to model a wide spectrum of QoE features, as summarized in Table 5.1

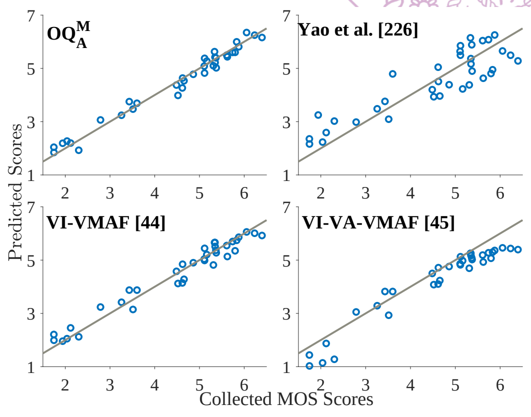


Figure 5.13: The predicted versus collected MOS scores for the overall QoE.

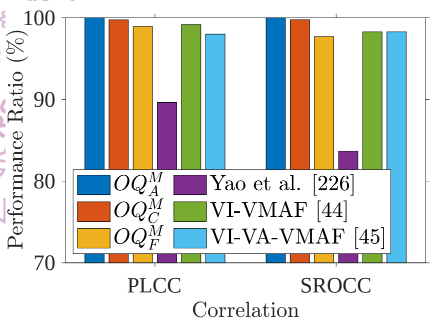


Figure 5.14: The performance ratio comparisons among our models, VI-VA-VMAF, VI-VMAF, and Yao et al. for overall QoE.

5.4 IS Modeling

In this section, we use the linear regressor to model the IS scores for the overall QoE and QoE features using the same setting in Sec. 5.3. We describe the modeling performance and observations in this section. Furthermore, because this work is the very first attempt to build IS models for 360° tiled videos, we cannot compare our IS models against existing ones in the literature.

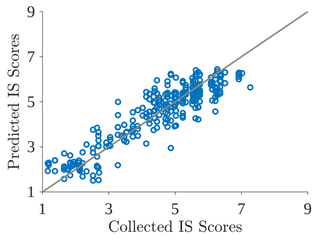


Figure 5.15: The predicted versus collected IS scores of the overall QoE.

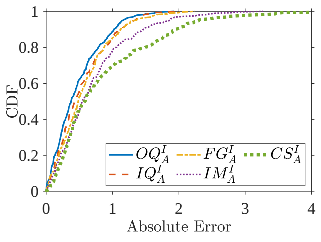
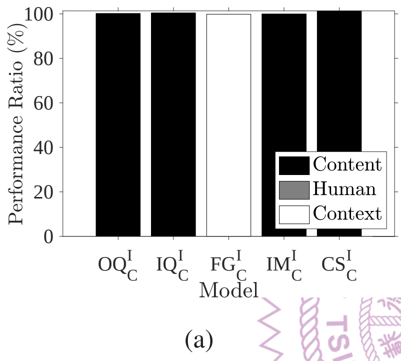


Figure 5.16: The CDF curves of the AE for the IS scores. We zoom into [0,4], where the range is [1,8].

Table 5.7: Performance of Derived IS Models with All Factors

Model	PLCC	SROCC	MAE	RMSE
$\mathbf{OQ^I_A}$	0.915	0.868	0.472	0.604
$\mathbf{IQ^I_A}$	0.896	0.847	0.532	0.667
$\mathbf{FG^I_A}$	0.883	0.868	0.565	0.704
$\mathbf{IM^I_A}$	0.801	0.725	0.696	0.899
$\mathbf{CS^I_A}$	0.579	0.594	0.862	1.165



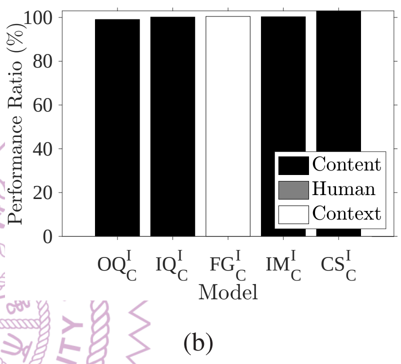
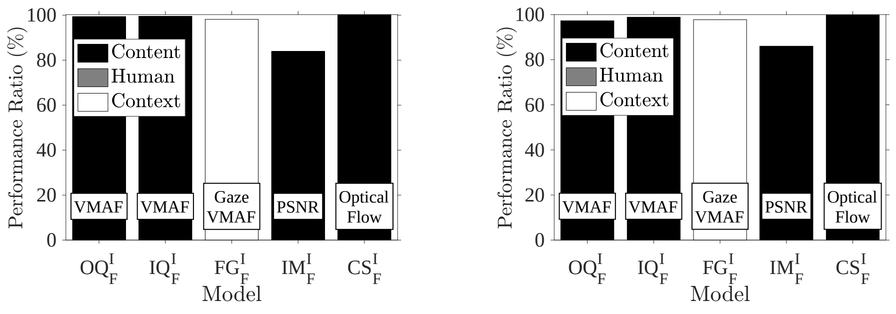


Figure 5.17: The performance ratio achieved by our derived IS models using factors from the dominating category: (a) PLCC and (b) SROCC.

Our derived IS models perform well on the overall QoE and most QoE features. We first consider all content, human, and context factors for the IS modeling (e.g., OQ^I_A). We plot the predicted versus collected IS scores of overall QoE in Fig. 5.15. Compared to Fig. 5.9, this figure shows that the linear correlation between our predicted and collected IS scores is slightly weaker than that for MOS scores. This is intuitive because IS scores involve individual differences from the subjects and thus are more challenging to model [103]. We further plot the CDF curves of the AE of each predicted IS score compared to the collected one in Fig. 5.16. This figure demonstrates a similar trend as seen in Fig. 5.15: the AE of IS modeling is slightly worse than that of MOS modeling, compared to Fig. 5.10. Furthermore, CS leads to higher AE compared to others. Table 5.7 reports the modeling performance of our derived IS models with all the considered factors. This table shows that with all QoE factors, our derived model for overall QoE achieves 0.915 and 0.868 in PLCC and SROCC, respectively. Besides, our derived models lead to $>$ 0.80 on PLCC for most QoE features, except CS. The gap (cf. Table 5.6) between the predicted MOS and IS scores for CS may be attributed to diverse: (i) personality and (ii)



(a) (b)

Figure 5.18: The performance ratio achieved by our derived IS models using the dominating factor: (a) PLCC and (b) SROCC.

prior experience, which impose nontrivial impacts on CS. This indicates that *more human factors should be included for modeling IS scores for cybersickness*. For example, the sleeping quality or the sense of balance may affect the perceived CS level. Adding more factors for IS modeling of CS is left as our future work.

The investigation on the dominating factor (category) for IS modeling shows a similar trend to MOS modeling. We then investigated the dominating factor category of IS models. We plotted the performance ratio achieved by our derived models between using the dominating factor category and using all factors in Fig. 5.17. This figure shows the same trend as the MOS models (cf. Fig. 5.11): the content factors dominate, except for FG. In particular, the context factors dominate the IS models for FG. The performance ratio achieved by the dominating factor category is quite high: $> 99\%$ for both PLCC and SROCC. Fig. 5.18 further plots the performance ratio of our derived IS models between using the dominating factor and using all factors. This figure also shows similar results as MOS models (cf. Fig. 5.12): (i) (gaze) VMAF dominates the IS models for the overall QoE, IQ, and FG (ii) the optical flow is the dominating factor for CS, and (iii) IM requires at least all content factors to achieve acceptable modeling performance.

5.5 Conclusions

In this chapter, we compile a wide spectrum of QoE features and QoE factors that may affect the overall QoE of watching tiled 360° videos using HMDs. We design and conduct a user study for collecting the dataset of the overall QoE, QoE features, and QoE factors. We then derive the models for the overall QoE and QoE features using the considered QoE factors in terms of both MOS and IS scores. Our key findings are:

- Our derived models for overall QoE perform very well and achieve 0.99 and 0.97 in PLCC and SROCC, respectively. Although modeling IS is harder, our derived models also perform well with most QoE features, except CS. For example, our derived IS model for the overall QoE achieves 0.92 and 0.87 in PLCC and SROCC, respectively.

- This chapter is the first work that considers comprehensive QoE factors spanning over the content, human, and context categories. Our modeling results reveal that *content* factors dominate the factor categories for the overall QoE, and most QoE features. The only exception is FG, which is affected more by the context factors.

- *(Gaze) VMAF* is the dominating factor for the overall QoE and most of the QoE features (IQ, and FG). This can be attributed to the fact that VMAF was trained with the *subjective* scores in Netflix's dataset [141]. VI-VMAF and VI-VA-VMAF further improve VMAF, because they are tailored for 360° videos.

- IM cannot be well modeled using a single dominating factor, and requires all content factors for reasonable modeling performance.

This chapter is the first study that models: (i) not only overall QoE but also multiple QoE features and (ii) not only MOS scores but also IS scores. Several of our findings are, therefore, novel and have not been raised in the literature. To the best of our knowledge, all existing work can be seen as a subset of our work. We have made the collected dataset and derived models public [150], which will stimulate more studies in the growing research area.

There are some limitations in this work that should be noted as follows:

- There is a nontrivial performance gap, (~ 0.30 in PLCC/SROCC) between the MOS and IS models for CS. This indicates that CS is strongly subject-dependent, which is intuitive. The gap may be reduced by considering more human and context factors, such as the physical conditions and mental states of subjects.

- We observe that AT affects the overall QoE. Hence, the models can be enhanced by additional surveys on subjects' preferences. This can also be approximated with the historical logs in real deployments, e.g., by analyzing the watched videos of each subject.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Emerging VR applications are growing in popularity because of the increasing availability of 360° videos and commodity HMDs. Streaming 360° videos to HMDs is, however, very challenging due to large video sizes, stringent real-time requirements, and complex human visual systems. In this thesis, we address these problems by optimizing the three critical phases of 360° video streaming to HMDs: (i) delivery, (ii) production, and (iii) consumption phases. In particular, we optimize the resource allocation in the first two phases and investigate the perceived QoE in the last phase.

In our first work, we avoided wasting resources on the unwatched part of 360° videos by predicting the viewer's future viewport. In particular, we leverage both sensor and content data to train a neural network for viewer fixation prediction, which predict the viewing probability of each tile. Several novel enhancements are proposed to improve the prediction accuracy, including generating virtual viewports, considering future content, and reducing the feature sampling rate. Our trace-driven simulation shows that our proposed fixation prediction network saves up to 41% of network bandwidth while achieving comparable video quality and lower rebuffering time compared to the current practices.

After optimizing the resource allocation in the delivery phase, we move to the earlier phase. In particular, we optimize the encoding ladder in the production phase, where the encoding ladder is used to determine the representations of individual tiled segments to be encoded and stored on the streaming servers. We formulate the problem into an ILP problem considering video models, viewing probability, and client distribution to maximize the client viewing quality. A divide-and-conquer approach is proposed to solve the formulated problem. Besides, mathematical optimization, e.g., convex optimization using Lagrangian multiplier, is applied to optimally solve the subproblems. Our experiment results demonstrate that our proposed algorithms result in better viewing quality and have better scalability under different storage limits compared to the state-of-the-art algorithm.

After optimizing the first two phases in system perspective, we then move to investigating the perceived QoE from the user perspective in the consumption phase. We designed and conducted a user study to investigate the complex relationship among the overall QoE, QoE features, and QoE factors on watching tiled 360° videos using HMDs. In particular, we considered 5 QoE features and 30 QoE factors from content, human, and context categories. We analyzed the viewing behaviors and modeled the overall QoE and various QoE features using the considered QoE factors. The dominating factor categories and factors were identified for the overall QoE and QoE features. Several observations are made based on our investigation. For example, content factors dominate the overall QoE and most QoE features. Besides, the cybersickness requires more human factors for better modeling performance. The proposed optimization and investigation in the three critical phases help achieve a QoE-driven optimized 360° video streaming to HMDs system.

6.2 Future Work

In the current work, we have optimized the three critical phases of 360° video streaming to HMDs, which are investigated from systems to users. Although our work focuses on 360° video streaming in VoD scenarios, the proposed studies can be extended for various scenarios. This will also enable the exploration of further immersive applications in the future. We discuss the opportunities below.

6.2.1 Live 360° Video Streaming

Our current work mainly focuses on the VoD scenario, where the content is encoded and stored on the streaming servers in the production phase. The content features are, thus, available to be exploited in our proposed optimization algorithms in different phases. For example, the saliency maps and motion maps are detected and stored on the streaming server in the production phase for the fixation prediction during the delivery phase. However, in live 360° video streaming, the content features are not available beforehand because the conteniss directly captured, encoded, and streamed to the clients in real-time. To apply our proposed algorithms to live video streaming, the extraction and analysis of content features need to be accelerated. For example, a real-time saliency detection [231] using contour information can be adopted for the fixation prediction networks. Another possible way is to eliminate the dependencies of the real content. For example, the video prediction network [183] can be exploited to predict the future video frames for content feature extraction and analysis. With the content features being available, Real-Time Protocol (RTP) can be adopted for low latency streaming, where the per-class optimization in our proposed optimal laddering algorithms can be directly exploited to determine the live bitrate.

6.2.2 6DoF Content Streaming

360° videos only support 3DoF interaction models, where the viewport is determined only by the viewer's head orientation assuming the viewer does not change his/her position. That is, the viewport would not reflect the position changes when the viewer walks around or moves close to certain objects. This obviously degrades the perceived immersive experience. 6DoF interaction models address the above limitations by rendering the viewport depending on the viewer's position when he/she walks. To support 6DoF content, there are several representations, which can be classified into two classes: (i) video- and (ii) volumetric-based representations. The video-based representations leverage the highly optimized 2D compression algorithms while the volumetric-based representations employ specialized data structure for compressions. Streaming such 6DoF content, however, is challenging because of the even larger amount of data size and more complex reconstruction process compared to 3DoF content (360° videos). This indicates that the problems faced by 360° video streaming also occur in 6DoF content streaming and become even more severe. Hence, it is essential to investigate the pros and cons of different representations for 6DoF content and develop suitable solutions, where the solutions can be completely designed for 6DoF content or extended from the solutions for 360° videos. For example, semantic features based on 3D object detection may be leveraged to improve the performance of fixation prediction in 6DoF content. In addition, the perceived QoE for 6DoF content adaptive streaming, such as density-based point cloud and image-based view synthesis, requires further investigation.

6.2.3 VR Gaming with Multiple Observers

6DoF streaming content expands the possibilities of various future immersive applications, including VR gaming. For popular computer game tournaments, the game scenes are also streamed to non-gamers, which are referred to as *observers*. The observers watch live game streaming to learn the skills from elite gamers and the story lines of games. To support observers in VR gaming, a naive approach is to render the gamer's viewport to the observers throughout the live streaming. However, this makes the VR features totally useless because there is no freedom of changing viewports anymore. Hence, the support of multiple free-viewpoint observers of VR gaming should be studied to realize an innovative way to observe live gaming streaming with better user-driven viewing quality. Cloud gaming [78] can be leveraged to lower the hardware requirements of the VR gamers and observers. However, offloading the computation resources onto the cloud may increase the latency. To overcome this, Mobile Edge Computing (MEC) servers may be used to replace cloud servers for reducing the latency. This is because MEC servers are closer to the VR gamers and observers, so as to prevent the traffic congestion in the core network. However, a high density of observers or unbalanced resource requirements in some regions may overload the edge servers, which will lead to inferior performance. Assigning game observers to the *best* possible edge servers is no easy task because it requires considering the dynamic environments [80], latency-sensitive characteristics [72], and interplay among multiple servers/clients [106]. Hence, an algorithm to efficiently assign edge servers to the cloud VR gamers and observers is required to minimize their perceived latency .

6.2.4 Movie Creation for XR Content

The development of 360° videos also stimulates the development of the 360° movie industry. *360° movies* not only share immersive experience to others as 360° videos. Instead, there are story lines in 360° movies. Delivering the story lines via 360° movies is, however, challenging, because the audience is allowed to freely change his/her viewport. That is, the audience may miss important scenes or hints in the movies for understanding and following the story. Besides, any scene cuts or transitions in 360° movies may cause cybersickness that degrade the immersive experience. Hence, a trade-off between the story presented by the director and the immersive experience perceived by the audience needs to be made. We believe that a more comprehensive user study investigating the factors that may affect the user experience of watching 360° movies need to be designed and conducted. In particular, the factors that may help attract viewer's gazes and the transition effects that may cause the least cybersickness are essential to be identified. The investigation can be exploited to provide presentation or transition recommendations to movie directors according to the characteristics of the stories and the concerns of the perceived cybersickness.

Bibliography

[1] Is 360-degree and VR video the future of marketing?, July 2017. `https://www.marketingtechnews.net/news/2017/jul/06/360-degree-and-vr-video-future-marketing/`.

[2] M. Abdallah, C. Griwodz, K. Chen, G. Simon, P. Wang, and C. Hsu. Delay-sensitive video computing in the cloud: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 14(3s):54:1–54:29, 2018.

[3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[4] P. Alface, M. Aerts, D. Tytgat, S. Lievens, C. Stevens, N. Verzijp, and J. Macq. 16K cinematic VR streaming. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 1105–1112, Mountain View, CA, October 2017.

[5] P. Alface, J. Macq, and N. Verzijp. Interactive omnidirectional video delivery: A bandwidth-effective approach. *Bell Labs Technical Journal*, 16(4):135–147, March 2012.

[6] T. Alshawi, Z. Long, and G. AlRegib. Understanding spatial correlation in eye-fixation maps for visual attention in videos. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'16)*, pages 1–6, Seattle, WA, July 2016.

[7] R. Anderson, D. Gallup, J. Barron, J. Kontkanen, N. Snavely, C. Hernandez, S. Agarwal, and S. Seitz. Jump: Virtual reality video. *ACM Transactions on Graphics*, 35(6):198:1–198:13, November 2016.

[8] M. Anwar, J. Wang, W. Khan, A. Ullah, S. Ahmad, and Z. Fei. Subjective QoE of 360-degree virtual reality videos and machine learning predictions. *IEEE Access*, 8:148084–148099, August 2020.

[9] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon. Transcoding live adaptive video streams at a massive scale in the cloud. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'15)*, pages 49–60, Portland, OR, March 2015.

[10] S. Aroussi and A. Mellouk. Survey on machine learning-based QoE-QoS correlation models. In *Proc. of IEEE International Conference on Computing, Management and Telecommunications (ComManTel'14)*, pages 200–204, Da Nang, Vietnam, April 2014.

[11] M. Assens, X. Giro-i-Nieto, K. McGuinness, and N. O'Connor. Saltinet: Scan-path prediction on 360 degree images using saliency volumes. In *Proc. of IEEE International Conference on Computer Vision (ICCV'17)*, pages 2331–2338, Venice, Italy, October 2017.

[12] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'18)*, pages 1–6, San Diego, CA, July 2018.

[13] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu. Motion-prediction-based multicast for 360-degree video transmissions. In *Proc. of IEEE International Conference on Sensing, Communication, and Networking (SECON'17)*, pages 1–9, San Diego, CA, June 2017.

[14] D. Basak, S. Pal, and D. Patranabis. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203–224, October 2007.

[15] A. Bentaleb, B. Taani, A. Begen, C. Timmerer, and R. Zimmermann. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials*, 21(1):562–585, 2018.

[16] Bert Hubert. tc, 2019. `https://linux.die.net/man/8/tc`.

[17] D. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 2014.

[18] D. Bertsekas, R. Gallager, and P. Humblet. *Data networks*, volume 2. Prentice-Hall International New Jersey, 1992.

[19] M. Bessa, M. Melo, D. Narciso, L. Barbosa, and J. Vasconcelos-Raposo. Does 3D 360 video enhance user's VR experience: An evaluation study. In *Proc. of*

International Conference on Human Computer Interaction (Interaction'16), pages 16:1–16:4, Salamanca, Spain, September 2016.

[20] A. Borji, M. Cheng, H. Jiang, and J. Li. Salient object detection: A survey. *arXiv preprint arXiv:1411.5878*, 2014.

[21] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. of International Conference on Computational Statistics (COMPSTAT'10)*, pages 177–186. Paris, France, August 2010.

[22] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid. Quality of experience for streaming services: Measurements, challenges and insights. *IEEE Access*, 8:13341–13361, January 2020.

[23] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[24] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[25] L. Bush, U. Hess, and G. Wolford. Transformations for within-subject designs: a monte carlo investigation. *Psychological bulletin*, 113(3):566, 1993.

[26] E. Canessa and L. Tenze. FishEyA: Live broadcasting around 360 degrees. In *Proc. of ACM Symposium on Virtual Reality Software and Technology (VRST'14)*, pages 227–228, Edinburgh, Scotland, November 2014.

[27] S. Chaabouni, J. Benois-Pineau, and C. Amar. Transfer learning with deep networks for saliency prediction in natural video. In *Proc. of IEEE International Conference on Image Processing (ICIP'16)*, pages 1604–1608, Phoenix, Arizona, September 2016.

[28] J. Chakareski, R. Aksu, X. Corbillon, G. Simon, and V. Swaminathan. Viewport-driven rate-distortion optimized 360° video streaming. In *Proc. of IEEE International Conference on Communications (ICC'18)*, pages 1–7, Kansas, MO, May 2018.

[29] S. Channappayya, A. Bovik, C. Caramanis, and R. Heath. SSIM-optimal linear image restoration. In *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'08)*, pages 765–768, Las Vegas, NV, March 2008.

[30] X. Chen, A. Kasgari, and W. Saad. Deep learning for content-based personalized viewport prediction of 360-degree VR videos. *IEEE Networking Letters*, 2(2):81–84, June 2020.

[31] H. Cheng, C. Chao, J. Dong, H. Wen, T. Liu, and M. Sun. Cube padding for weakly-supervised saliency prediction in 360° videos. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, pages 1420–1429, Salt Lake City, UT, June 2018.

[32] K. Choi and K. Jun. Real-Time panorama video system using networked multiple cameras. *Journal of Systems Architecture*, 64:110–121, March 2016.

[33] Cisco Inc. Cisco visual networking index: Forecast and trends, 2017—2022 white paper, 2017. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html`.

[34] Cisco Systems, Inc. The Zettabyte Era: Trends and Analysis, 2017. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html`.

[35] C. Concolato, J. L. Feuvre, F. Denoual, E. Nassor, N. Ouedraogo, and J. Taquet. Adaptive streaming of HEVC tiled videos using MPEG-DASH. *IEEE Transactions on Circuits and Systems for Video Technology*, PP(99):1–1, March 2017.

[36] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski. Optimal set of 360-degree videos for viewport-adaptive streaming. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 943–951, Mountain View, CA, October 2017.

[37] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski. Optimal set of 360-Degree videos for Viewport-Adaptive streaming. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 943–951, Mountain View, CA, October 2017.

[38] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *Proc. of IEEE International Conference on Communications (ICC'17)*, pages 1–7, Paris, France, May 2017.

[39] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth. On the LambertW function. *Advances in Computational mathematics*, 5(1):329–359, 1996.

[40] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara. A deep multi-level network for saliency prediction. In *Proc. of International Conference on Pattern Recognition (ICPR'16)*, pages 3488–3493, Cancun, Mexico, December 2016.

[41] S. Croci, C. Ozcinar, E. Zerman, J. Cabrera, and A. Smolic. Voronoi-based objective quality metrics for omnidirectional video. In *Proc. of International Conference on Quality of Multimedia Experience (QoMEX'19)*, pages 1–6, Berlin, Germany, June 2019.

[42] S. Croci, C. Ozcinar, E. Zerman, S. Knorr, J. Cabrera, and A. Smolic. Visual attention-aware quality estimation framework for omnidirectional video using spherical voronoi diagram. *Springer Quality and User Experience*, 5(1):1–17, April 2020.

[43] L. D'Acunto, J. van den Berg, E. Thomas, and O. Niamut. Using MPEG DASH SRD for zoomable and navigable video. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'16)*, pages 34:1–34:4, Klagenfurt, Austria, May 2016.

[44] DASH Industry Forum. Guidelines for implementation: Dash-if interoperability points. *DASH Industry Forum*, November 2018.

[45] DeNA Co., Ltd. et al. H2O the optimized HTTP/1.x, HTTP/2 server, 2019. `https://h2o.example.net/`.

[46] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 248–255, Miami, Florida, June 2009.

[47] F. Duanmu, E. Kurdoglu, S. Hosseini, Y. Liu, and Y. Wang. Prioritized buffer control in two-tier 360 video streaming. In *Proc. of ACM SIGCOMM Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network'17)*, pages 13–18, Los Angeles, CA, August 2017.

[48] D. Egan, S. Brennan, J. Barrett, Y. Qiao, C. Timmerer, and N. Murray. An evaluation of heart rate and electrodermal activity as an objective QoE evaluation method for immersive virtual reality environments. In *Proc. of International Conference on Quality of Multimedia Experience (QoMEX'16)*, pages 1–6, Lisbon, Portugal, June 2016.

[49] T. El-Ganainy and M. Hefeeda. Streaming virtual reality content. *arXiv preprint arXiv:1612.08350*, December 2016.

[50] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu. Fixation prediction for 360° video streaming in head-mounted virtual reality. In *Proc. of ACM SIGMM*

Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17), pages 67–72, Taipei, Taiwan, June 2017.

[51] C. Fan, W. Lo, Y. Pai, and C. Hsu. A survey on 360° video streaming: Acquisition, transmission, and display. *ACM Computing Surveys*, 52(4):71:1–71:30, August 2019.

[52] C. Fan, S. Yen, C. Huang, and C. Hsu. Optimizing fixation prediction using recurrent neural networks for 360° video streaming in head-mounted virtual reality. *IEEE Transactions on Multimedia*, 22(3):744–759, March 2019.

[53] C. Fan, S. Yen, C. Huang, and C. Hsu. On the optimal encoding ladder of tiled 360° videos for head-mounted virtual reality. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–14, 2020. Accepted to Appear.

[54] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[55] X. Feng, Y. Liu, and S. Wei. LiveDeep: Online viewport prediction for live virtual reality streaming using lifelong deep learning. In *IEEE Conference on Virtual Reality and 3D User Interfaces (VR'20)*, pages 800–808, Atlanta, Georgia, March 2020.

[56] X. Feng, V. Swaminathan, and S. Wei. Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking. *Proc. of ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):1–22, June 2019.

[57] A. Fernandes and S. Feiner. Combating VR sickness through subtle dynamic field-of-view modification. In *Proc. of IEEE Symposium on 3D User Interfaces (3DUI'16)*, pages 201–210, Greenville, SC, March 2016.

[58] A. Ferworn, B. Waismark, and M. Scanlan. CAT 360: Canine augmented technology 360-Degree video system. In *Proc. of IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR'15)*, pages 1–4, West Lafayette, IN, October 2015.

[59] D. Freedman. *Statistical models: theory and practice*. Cambridge University Press, 2009.

[60] J. Friedman. Greedy function approximation: a gradient boosting machine. *JSTOR Annals of Statistics*, 29(5):1189–1232, October 2001.

[61] C. Fu, L. Wan, T. Wong, and C. Leung. The rhombic dodecahedron map: An efficient scheme for encoding panoramic video. *IEEE Transactions on Multimedia*, 11(4):634–644, June 2009.

[62] V. Gaddam, H. Ngo, R. Langseth, C. Griwodz, D. Johansen, and P. Halvorsen. Tiling of panorama video for interactive virtual cameras: Overheads and potential bandwidth requirement reduction. In *Proc. of Picture Coding Symposium (PCS'15)*, pages 204–209, Cairns, Australia, May 2015.

[63] L. Gaemperle, K. Seyid, V. Popovic, and Y. Leblebici. An immersive telepresence system using a real-time omnidirectional camera and a virtual reality head-mounted display. In *Proc. of IEEE International Symposium on Multimedia (ISM'14)*, pages 175–178, Taichung, Taiwan, December 2014.

[64] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722, June 2010.

[65] Google Inc. ExoPlayer : An extensible media player for android, 2017. `https://github.com/google/ExoPlayer`.

[66] M. Graf, C. Timmerer, and C. Mueller. Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, pages 261–271, Taipei, Taiwan, June 2017.

[67] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, March 2019. `http://cvxr.com/cvx`.

[68] J. Hakkinen, T. Vuori, and M. Paakka. Postural stability and sickness symptoms after HMD use. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 147–152, October 2002.

[69] E. Hjelmås and B. Low. Face detection: A survey. *Computer vision and image understanding*, 83(3):236–274, September 2001.

[70] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, November 1997.

[71] X. Hou, S. Dey, J. Zhang, and M. Budagavi. Predictive adaptive streaming to enable mobile 360-degree and VR experiences. *IEEE Transactions on Multimedia*, pages 1–16, 2020.

[72] X. Hou, Y. Lu, and S. Dey. Wireless VR/AR with edge/cloud computing. In *Proc. of International Conference on Computer Communication and Networks (IC-CCN'17)*, pages 1–8, Vancouver, Canada, July 2017.

[73] X. Hou, J. Zhang, M. Budagavi, and S. Dey. Head and body motion prediction to enable mobile VR experiences with low latency. In *IEEE Global Communications Conference (GLOBECOM'19)*, pages 1–7, Waikoloa, HI, December 2019.

[74] C. Hsu, A. Chen, C. Hsu, C. Huang, C. Lei, and K. Chen. Is foveated rendering perceivable in virtual reality: Exploring the efficiency and consistency of quality assessment methods. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 55–63, Mountain View, CA, October 2017.

[75] HTC Corporation. VIVE: Discover virtual reality beyond imagination, 2017. https://www.vive.com/us/.

[76] HTC Corporation. Eye tracking SDK (SRanipal), 2020. https://developer.vive.com/resources/knowledgebase/vive-sranipal-sdk/.

[77] HTC Corporation. VIVE PRO EYE, 2020. https://enterprise.vive.com/us/product/vive-pro-eye/.

[78] C. Huang, K. Chen, D. Chen, H. Hsu, and C. Hsu. GamingAnywhere: The first open source cloud gaming system. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 10:1–10:24(1), Jan 2014.

[79] M. Huang, Q. Shen, Z. Ma, A. C. Bovik, P. Gupta, R. Zhou, and X. Cao. Modeling the perceptual quality of immersive images rendered on head mounted displays: Resolution and compression. *IEEE Transactions on Image Processing*, 27(12):6039–6050, December 2018.

[80] F. Huber and S. Satish. Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning. In *Proc. of ACM Workshop on Mobile Cloud Computing and Services (MCS'13)*, pages 9–16, Taipei, Taiwan, June 2013.

[81] IBM Corp. IBM ILOG CPLEX optimizer, 2018. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[82] M. Inoue, H. Kimata, K. Fukazawa, and N. Matsuura. Interactive panoramic video streaming system over restricted bandwidth network. In *Proc. of ACM Interna-*

tional Conference on Multimedia (MM'10), pages 1191–1194, Firenze, Italy, October 2010.

[83] Information technology – dynamic adaptive streaming over HTTP (DASH) – part 1: Media presentation description and segment formats, March 2012.

[84] Algorithm descriptions of projection format conversion and video quality metrics in 360Lib. Standard, International Telecommunication Union, 2017.

[85] ITU. JVET - joint video experts team, 2019. `https://www.itu.int/en/ITU-T/studygroups/2017-2020/16/Pages/video/jvet.aspx`.

[86] Final report from the video quality experts group on the validation of objective models of video quality assessment. Technical report, VQEG, 2000.

[87] ITU-T Study Group 9. Subjective video quality assessment methods for multimedia applications. *ITU Series P: Audiovisual quality in multimedia services*, 1999.

[88] ITU Telecommunication Standardization Sector. Subjective video quality assessment methods for multimedia applications. *ITU-T Recommendation*, P.910, April 2008.

[89] ITU Telecommunication Standardization Sector. Methods for the subjective assessment of video quality, audio quality and audiovisual quality of internet video and distribution quality television in any environment. *ITU-T Recommendation*, P.913, March 2016.

[90] ITU Telecommunication Standardization Sector. Series p: Terminals and subjective and objective assessment methods. *ITU-T Recommendation*, P.800.2, July 2016.

[91] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation*, pages 1–37, September 1984.

[92] N. Jiang, V. Swaminathan, and S. Wei. Power evaluation of 360 VR video streaming on head mounted display devices. In *Proc. of ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*, pages 55–60, Taipei, Taiwan, June 2017.

[93] B. John, P. Raiturkar, O. Le Meur, and E. Jain. A benchmark of four methods for generating 360° saliency maps from eye tracking data. *World Scientific International Journal of Semantic Computing*, 13(03):329–341, September 2019.

[94] R. Ju, J. He, F. Sun, J. Li, F. Li, J. Zhu, and L. Han. Ultra wide view based panoramic VR streaming. In *Proc. of ACM SIGCOMM Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network'17)*, pages 19–23, Los Angeles, CA, August 2017.

[95] P. Juluri. AStream, 2019. `https://github.com/pari685/AStream`.

[96] P. Juluri, V. Tamarapalli, and D. Medhi. Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):401–418, February 2016.

[97] S. Jumisko-Pyykkö and T. Vainio. Framing the context of use for mobile HCI. *IGI Global International Journal of Mobile Human Computer Interaction*, 2(4):1–28, January 2010.

[98] Y. Kavak, E. Erdem, and A. Erdem. A comparative study for feature integration strategies in dynamic saliency estimation. *Signal Processing: Image Communication*, 51:13–25, February 2017.

[99] R. Kennedy, N. Lane, K. Berbaum, and M. Lilienthal. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3(3):203–220, November 1993.

[100] I. Ketykó, K. De Moor, T. De Pessemier, A. Verdejo, K. Vanhecke, W. Joseph, L. Martens, and L. De Marez. Qoe measurement of mobile YouTube video streaming. In *Proc. of Workshop on Mobile Video Delivery*, pages 27–32, Firenze, Italy, October 2010.

[101] H. Kim, H. Lim, S. Lee, and Y. Ro. VRSA net: VR sickness assessment considering exceptional motion for 360 VR video. *IEEE Transactions on Image Processing*, 28(4):1646–1660, April 2019.

[102] H. Kim, J. Yang, M. Choi, J. Lee, S. Yoon, Y. Kim, and W. Park. Immersive 360° VR tiled streaming system for Esports service. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'18)*, pages 541–544, Amsterdam, The Netherlands, June 2018.

[103] J. Kim, W. Kim, S. Ahn, J. Kim, and S. Lee. Virtual reality sickness predictor: Analysis of visual-vestibular conflict and VR contents. In *Proc. of IEEE International Conference on Quality of Multimedia Experience (QoMEX'18)*, pages 1–6, Sardinia, Italy, May 2018.

[104] H. Kimata, M. Isogai, H. Noto, M. Inoue, K. Fukazawa, and N. Matsuura. Interactive panorama video distribution system. In *Proc. of Technical Symposium at ITU Telecom World (ITUWT'11)*, pages 45–50, Geneva, Switzerland, October 2011.

[105] H. Kimata, D. Ochi, A. Kameda, H. Noto, K. Fukazawa, and A. Kojima. Mobile and multi-device interactive panorama video distribution system. In *Proc. of IEEE Global Conference on Consumer Electronics (GCCE'12)*, pages 574–578, Tokyo, Japan, October 2012.

[106] K. Kitae, L. Jared, K. Seokwon, and H. Seon. Prediction based sub-task offloading in mobile edge computing. In *Proc of IEEE International Conference on Information Networking (ICOIN'19)*, pages 448–452, January 2019.

[107] J. Kopf. 360° video stabilization. *ACM Transactions on Graphics*, 35(6):195:1–195:9, November 2016.

[108] J. Kua, G. Armitage, and P. Branch. A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP. *IEEE Communications Surveys & Tutorials*, 19(3):1842–1866, 2017.

[109] E. Kuzyakov and D. Pio. Next-generation video encoding techniques for 360 video and VR, 2016. `https://engineering.fb.com/virtual-reality/next-generation-video-encoding-techniques-for-360-video-and-vr/`.

[110] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and Z. Shi. The QUIC transport protocol: Design and internet-scale deployment. In *Proc. of the ACM International Conference on Special Interest Group on Data Communication (SIGCOMM'17)*, pages 183–196, Los Angeles, CA, August 2017.

[111] J. Le Feuvre and C. Concolato. Tiled-based adaptive streaming using MPEG-DASH. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'16)*, pages 41:1–41:3, Klagenfurt, Austria, May 2016.

[112] T. Lee, J. Yoon, and I. Lee. Motion sickness prediction in stereoscopic videos using 3D convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1919–1927, May 2019.

[113] B. Li, H. Li, L. Li, and J. Zhang. λ domain rate control algorithm for high efficiency video coding. *IEEE Transactions on Image Processing*, 23(9):3841–3854, September 2014.

[114] B. Li, D. Zhang, H. Li, and J. Xu. QP determination by lambda value. In *9th Meeting of the JCT-VC, no. JCTVC-I0426*, May 2012.

[115] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 5455–5463, Boston, MA, June 2015.

[116] Z. Li, M. Drew, and J. Liu. *Lossy Compression Algorithms*. Springer, 2004.

[117] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, December 2002.

[118] S. Lim, J. Seok, J. Seo, and T. Kim. Tiled panoramic video transmission system based on MPEG-DASH. In *Proc. of International Conference on Information and Communication Technology Convergence (ICTC'15)*, pages 719–721, Jeju, Korea, October 2015.

[119] K. Lin, S. Liu, L. Cheong, and B. Zeng. Seamless video stitching from hand-held camera inputs. *Computer Graphics Forum*, 35(2):479–487, May 2016.

[120] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. Zitnick. Microsoft COCO: Common objects in context. In *Proc. of European Conference on Computer Vision (ECCV'14)*, pages 740–755, Zurich, Switzerland, September 2014.

[121] F. Liu and M. Gleicher. Region enhanced scale-invariant saliency detection. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'06)*, pages 1477–1480, Toronto, Canada, July 2006.

[122] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H. Shum. Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence*, 33(2):353–367, February 2011.

[123] W. Lo, C. Fan, J. Lee, C. Huang, K. Chen, and C.-H. Hsu. 360° video viewing dataset in head-mounted virtual reality. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, pages 211–216, Taipei, Taiwan, June 2017.

[124] W. Lo, C. Fan, S. Yen, and C. Hsu. Performance measurements of 360° video streaming to head-mounted displays over live 4G cellular networks. In *Proc.*

of Asia-Pacific Network Operations and Management Symposium (APNOMS'17), pages 205–210, Seoul, Korea, September 2017.

[125] K. Lu, A. Ortega, D. Mukherjee, and Y. Chen. Perceptually inspired weighted MSE optimization using irregularity-aware graph Fourier transform. *arXiv preprint arXiv:2002.08558*, 2020.

[126] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, Canada, August 1981.

[127] Y. Ma and H. Zhang. Contrast-based image attention analysis by using fuzzy growing. In *Proc. of ACM International Conference on Multimedia (MM'03)*, pages 374–381, Berkeley, CA, November 2003.

[128] Magnifyre. 360-degree video case study, 2017. `https://www.magnifyre.com/360-degree-video-case-study/`.

[129] A. Makhorin. GLPK (GNU linear programming kit), 2019. `https://www.gnu.org/software/glpk/`.

[130] MarketWatch. Global head mounted display (hmd) market - global countries data, insights, market size & growth, forecast to 2026, 2020. `https://www.marketwatch.com/press-release/global-head-mounted-display-hmd-market---global-countries-data-insights-market-size-growth-forecast-to-2026-2020-02-18`.

[131] MATLAB - MathWorks, 2020. `https://www.mathworks.com/products/matlab.html`.

[132] A. Mavlankar and B. Girod. Video streaming with interactive pan/tilt/zoom. In M. Mrak, M. Grgic, and M. Kunt, editors, *High-Quality Visual Experience*, chapter 19, pages 431–455. Springer, June 2010.

[133] T. Mikolov, M. Karafiat, L. Burget, J.Cernocky, and S. Khudanpur. Recurrent neural network based language model. In *Proc. of Conference on International Speech Communication Association (Interspeech'11)*, pages 1045–1048, Florence, Italy, August 2010.

[134] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou. An overview of tiles in HEVC. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):969–977, December 2013.

[135] S. Moller, M. Waltermann, and M. Garcia. Features of quality of experience. In S. Moller and A. R. Editors, editors, *Quality of Experience*, chapter 5, pages 73–84. Springer US, 2014.

[136] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic. Salnet360: Saliency maps for omni-directional images with CNN. *Signal Processing: Image Communication*, 69:26–34, Novemebr 2018.

[137] MPlayer. MPlayer: The movie player, 2017. `http://www.mplayerhq.hu`.

[138] B. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. The MIT Press, 1996.

[139] A. Nasrabadi, A. Mahzari, J. Beshay, and R. Prakash. Adaptive 360-degree video streaming using scalable video coding. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 1689–1697, Mountain View, CA, October 2017.

[140] A. Nasrabadi, A. Samiei, and R. Prakash. Viewport prediction for 360° videos: a clustering approach. In *Proc. of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'20)*, pages 34–39, Istanbul, Turkey, June 2020.

[141] Netflix Inc. NFLX dataset, 2016. `https://drive.google.com/drive/u/0/folders/0B3YWNICYMBIweGdJbERlUG9zc0k`.

[142] Netflix Inc. VMAF - video multi-method assessment fusion, 2019. `https://github.com/Netflix/vmaf`.

[143] Netflix Technology Blog. Per-title encode optimization, 2015. `https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2`.

[144] K. Ngo, R. Guntur, and W. Ooi. Adaptive encoding of zoomable video streams based on user access pattern. In *Proc. of ACM Conference on Multimedia Systems (MMSys'11)*, pages 211–222, San Jose, CA, February 2011.

[145] A. Nguyen, Z. Yan, and K. Nahrstedt. Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction. In *Proc. of ACM International Conference on Multimedia (MM'18)*, pages 1190–1198, Seoul, Korea, October 2018.

[146] D. Nguyen, T. Huyen, and T. Thang. An evaluation of tile selection methods for viewport adaptive streaming of 360-degree video. *ACM Transactions on Multimedia Computing Communications and Applications*, 16(1):1–24, 2020.

[147] D. Nguyen, H. T. Tran, A. Pham, and T. Thang. A new adaptation approach for viewport-adaptive 360-degree video streaming. In *Proc. of IEEE International Symposium on Multimedia (ISM'17)*, pages 38–44, Taichung, Taiwan, December 2017.

[148] T. Nguyen, M. Xu, G. Gao, M. Kankanhalli, Q. Tian, and S. Yan. Static saliency vs. dynamic saliency: a comparative study. In *Proc. of ACM International Conference on Multimedia (MM'13)*, pages 987–996, Barcelona, Spain, October 2013.

[149] O. Niamut, A. Kochale, J. Hidalgo, R. Kaiser, J. Spille, J. Macq, G. Kienast, O. Schreer, and B. Shirley. Towards a format-agnostic approach for production, delivery and rendering of immersive media. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'13)*, pages 249–260, Oslo, Norway, February 2013.

[150] nmsl-nthu. QoE-modeling-for-360-degree-videos-dataset, 2020. `https://github.com/nmsl-nthu/QoE-Modeling-for-360-Degree-Videos-Dataset`.

[151] NS-3 network simulator, 2018. `http://www.nsnam.org/`.

[152] An MPEG/DASH client-server module for simulating rate adaptation mechanisms over HTTP/TCP, 2018. `https://github.com/djvergad/dash`.

[153] D. Ochi, Y. Kunita, K. Fujii, A. Kojima, S. Iwaki, and J. Hirose. HMD viewing spherical video streaming system. In *Proc. of the ACM International Conference on Multimedia (MM'14)*, pages 763–764, Orlando, FL, November 2014.

[154] D. Ochi, Y. Kunita, A. Kameda, A. Kojima, and S. Iwaki. Live streaming system for omnidirectional video. In *Proc. of IEEE Virtual Reality (VR'15)*, pages 349–350, Arles, France, March 2015.

[155] Oculus VR, LLC. Oculus rift, 2017. `https://www.oculus.com/`.

[156] J. Ohm and G. Sullivan. High efficiency video coding: the next frontier in video compression [standards in a nutshell]. *IEEE Signal Processing Magazine*, 30(1):152–158, January 2013.

[157] Opensignal. Global state of mobile networks (february 2017), 2017. https://opensignal.com/reports/2017/02/global-state-of-the-mobile-network.

[158] M. Orduna, C. Díaz, L. Muñoz, P. Pérez, I. Benito, and N. García. Video multimethod assessment fusion (VMAF) on 360VR contents. *IEEE Transactions on Consumer Electronics*, 66(1):22–31, February 2019.

[159] C. Ozcinar, J. Cabrera, and A. Smolic. Visual attention-aware omnidirectional video streaming using optimal tiles for virtual reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):217–230, March 2019.

[160] C. Ozcinar, A. De Abreu, S. Knorr, and A. Smolic. Estimation of optimal encoding ladders for tiled 360 VR video in adaptive streaming systems. In *IEEE International Symposium on Multimedia (ISM'17)*, pages 45–52, Taichiung, Taiwan, December 2017.

[161] C. Ozcinar, A. De Abreu, and A. Smolic. Viewport-aware adaptive 360° video streaming using tiles for virtual reality. In *Proc. of IEEE International Conference on Image Processing (ICIP'17)*, pages 2174–2178, Beijing, China, November 2017.

[162] N. Padmanaban, T. Ruban, V. Sitzmann, A. Norcia, and G. Wetzstein. Towards a machine-learning approach for sickness prediction in 360 stereoscopic videos. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1594–1603, April 2018.

[163] F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, M. Blondel, M. Blondel, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, October 2011.

[164] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck. An HTTP/2-based adaptive streaming framework for 360 virtual reality videos. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 306–314, Mountain View, CA, October 2017.

[165] S. Petrangeli, J. van der Hooft, T. Wauters, R. Huysegems, P. Alface, T. Bostoen, and F. De Turck. Live streaming of 4k ultra-high definition video over the internet. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'16)*, pages 27:1–27:4, Klagenfurt, Austria, May 2016.

[166] B. Petry and J. Huber. Towards effective interaction with omnidirectional videos using immersive virtual reality headsets. In *Proc. of ACM Augmented Human International Conference (AH'15)*, pages 217–218, Singapore, Singapore, March 2015.

[167] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proc. of International Conference on Mobile Computing and Networking (MobiCom'18)*, pages 99–114, New Delhi, India, October 2018.

[168] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan. Optimizing 360 video delivery over cellular networks. In *Proc. of Workshop on All Things Cellular Operations, Applications and Challenges (ATC'16)*, pages 1–6, New York, NY, October 2016.

[169] A. Raake, A. Singla, R. Rao, W. Robitza, and F. Hofmeyer. SiSiMo: Towards simulator sickness modeling for 360° videos viewed with an HMD. In *Proc. of IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW'20)*, pages 583–584, Atlanta, GA, 2020.

[170] M. Rahman, A. El Saddik, and W. Gueaieb. Augmenting context awareness by combining body sensor networks and social networks. *IEEE Transactions on Instrumentation and Measurement*, 60(2):345–353, February 2010.

[171] Y. Rai, J. Gutiérrez, and P. Le Callet. A dataset of head and eye movements for 360 degree images. In *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, pages 205–210, Taipei, Taiwan, June 2017.

[172] Y. Rai, P. Le Callet, and P. Guillotel. Which saliency weighting for omni directional image quality assessment? In *Proc. of IEEE International Conference on Quality of Multimedia Experience (QoMEX'17)*, pages 1–6, Erfurt, Germany, May 2017.

[173] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[174] G. Regal, R. Schatz, J. Schrammel, and S. Suette. VRate: a Unity3D asset for integrating subjective assessment questionnaires in virtual environments. In *Proc. of IEEE International Conference on Quality of Multimedia Experience (QoMEX'18)*, pages 1–3, Sardinia, Italy, May 2018.

[175] U. Reiter, K. Brunnström, K. De Moor, M. Larabi, M. Pereira, A. Pinheiro, J. You, and A. Zgank. Factors influencing quality of experience. In S. Moller and A. R. Editors, editors, *Quality of Experience*, chapter 4, pages 55–72. Springer US, 2014.

[176] Research and Markets. 360-degree camera market: Global industry trends, share, size, growth, opportunity and forecast 2020-2025, 2020. https://www.researchandmarkets.com/reports/5009145/360-degree-camera-market-global-industry-trends?utm_source=dynamic&utm_medium=BW&utm_code=sfklg3&utm_campaign=1375885+-+Global+360-Degree+Camera+Market+Report%2C+2020-2025%3A+Trends%2C+Share%2C+Size%2C+Growth%2C+Opportunities%2C+Competition&utm_exec=joca220bwd.

[177] Y. Reznik, K. Lillevold, A. Jagannath, J. Greer, and J. Corley. Optimal design of encoding profiles for ABR streaming. In *Proc. of ACM Workshop on Packet Video (PV'18)*, pages 43–47, Amsterdam, The Netherlands, June 2018.

[178] Samsung Electronics. The GearVR framework), 2017. https://github.com/Samsung/GearVRf.

[179] R. Schafer, P. Kauff, R. Skupin, Y. Sanchez, and C. weissig. Interactive streaming of panoramas and VR worlds. *SMPTE Motion Imaging Journal*, 126(1):35–42, January 2017.

[180] R. Schatz, G. Regal, S. Schwarz, S. Suettc, and M. Kempf. Assessing the QoE impact of 3D rendering style in the context of VR-based training. In *Proc. of IEEE International Conference on Quality of Multimedia Experience (QoMEX'18)*, pages 1–6, Sardinia, Italy, June 2018.

[181] I. T. S. Sector. Mean opinion score (mos) terminology. *ITU-T Recommendation*, P.800.1, July 2016.

[182] R. Shiffler. Maximum z scores and outliers. *The American Statistician*, 42(1):79–80, 1988.

[183] O. Shouno. Photo-realistic video prediction on natural videos of largely changing frames. *arXiv preprint arXiv:2003.08635*, 2020.

[184] A. Singla, S. Fremerey, W. Robitza, P. Lebreton, and A. Raake. Comparison of subjective quality evaluation for HEVC encoded omnidirectional videos at different bit-rates for UHD and FHD resolution. In *Proc. of ACM Multimedia Thematic Workshops Thematic Workshops'17))*, pages 511–519, Mountain View, CA, October 2017.

[185] A. Singla, S. Fremerey, W. Robitza, and A. Raake. Measuring and comparing QoE and simulator sickness of omnidirectional videos in different head mounted displays. In *Proc. of International Conference on Quality of Multimedia Experience (QoMEX'17)*, pages 1–6, Erfurt, Germany, May 2017.

[186] A. Singla, S. Goring, A. Raake, B. Meixner, R. Koenen, and T. Buchholz. Subjective quality evaluation of tile-based streaming for omnidirectional videos. In *Proc. of ACM Conference on Multimedia Systems (MMSys'19)*, Amherst, MA, February 2019.

[187] R. Skupin, Y. Sanchez, Y. Wang, M. Hannuksela, J. Boyce, and M. Wien. Standardization status of 360 degree video coding and delivery. In *Proc. of IEEE International Conference on Visual Communications and Image Processing (VCIP'17)*, pages 1–4, Taichung, Taiwan, December 2017.

[188] I. Sodagar. The MPEG-DASH standard for multimedia streaming over the Internet. *IEEE Multimedia*, 18(4):62–67, April 2011.

[189] W. Song, Y. Xiao, D. Tjondronegoro, and A. Liotta. QoE modelling for VP9 and H.265 videos on mobile devices. In *Proc. of ACM International Conference on Multimedia (MM'15)*, pages 501–510, Brisbane, Australia, October 2015.

[190] C. Spearman. The proof and measurement of association between two things. *American journal of Psychology*, 15(1):72–101, 1904.

[191] G. Sullivan, J. Ohm, W. Han, and T. Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, December 2012.

[192] G. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15(6):74–90, 1998.

[193] K. Tcha-Tokey, E. Loup-Escande, O. Christmann, and S. Richir. A questionnaire to measure the user experience in immersive virtual environments. In *Proc. of ACM Virtual Reality International Conference (VRIC'16)*, pages 1–5, Laval, France, March 2016.

[194] Telecom ParisTech. MP4Box, 2017. `https://gpac.wp.imt.fr/mp4box/`.

[195] Telecom ParisTech. MP4Client, 2017. `https://gpac.wp.imt.fr/player`.

[196] Trejkaz. Equi-angular cubemap skybox for Unity, 2020. https://github.com/trejkaz/EACSkyboxShader.

[197] I. Tucker. Perceptual video quality dimensions. Master's thesis, Technische Universität Berlin, Berlin, Germany, 2011.

[198] Unity, 2017. https://unity3d.com/.

[199] Unity Technologies. SteamVR plugin, 2020. https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647.

[200] E. Upenik, M. Rerabek, and T. Ebrahimi. Testbed for subjective evaluation of omnidirectional visual content. In *Proc. of Picture Coding Symposium (PCS'16)*, pages 1–5, Nuremberg, Germany, December 2016.

[201] E. Upenik, M. Rerabek, and T. Ebrahimi. On the performance of objective metrics for omnidirectional visual content. In *Proc. of International Conference on Quality of Multimedia Experience (QoMEX'17)*, pages 1–6, Erfurt, Germany, May 2017.

[202] M. Varela, L. Skorin-Kapov, and T. Ebrahimi. Quality of service versus quality of experience. In S. Moller and A. R. Editors, editors, *Quality of Experience*, chapter 6, pages 85–96. Springer US, 2014.

[203] J. Vielhaben, H. Camalan, W. Samek, and M. Wenzel. Viewport forecasting in 360° virtual reality videos with machine learning. In *IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR'19)*, pages 74–747, San Diego, CA, December 2019.

[204] M. Viitanen, A. Koivula, A. Lemmetti, A. Yla-Outinen, J. Vanne, and T. Hamalainen. Kvazaar: Open-source HEVC/H.265 encoder. In *Proc. of ACM International Conference on Multimedia (MM'16)*, pages 1179–1182, Amsterdam, The Netherlands, October 2016.

[205] S. Vlahovic, M. Suznjevic, and L. Skorin-Kapov. Subjective assessment of different locomotion techniques in virtual reality environments. In *Proc. of IEEE International Conference on Quality of Multimedia Experience (QoMEX'18)*, pages 1–3, Sardinia, Italy, June 2018.

[206] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. In *Proc. of IEEE Conference on Virtual Reality (VR'10)*, pages 211–218, Waltham, MA, March 2010.

[207] H. Wang, M. C. Chan, and W. Ooi. Wireless multicast for zoomable video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 12(1):5:1–5:23, August 2015.

[208] H. Wang, V. Nguyen, W. Ooi, and M. Chan. Mixing tile resolutions in tiled video: A perceptual quality assessment. In *Proc. of ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'14)*, pages 25:25–25:30, Singapore, Singapore, March 2014.

[209] K. Wang, L. Lin, J. Lu, C. Li, and K. Shi. PISA: Pixelwise image saliency by aggregating complementary appearance contrast measures with edge-preserving coherence. *IEEE Transactions on Image Processing*, 24(10):3019–3033, October 2015.

[210] M. Wang, K. Ngan, and H. Li. An efficient frame-content based intra frame rate control for high efficiency video coding. *IEEE Signal Processing Letters*, 22(7):896–900, July 2015.

[211] WebVR. WebVR: Bringing virtual reality to the web, 2017. `https://webvr.info/`.

[212] M. Weier, T. Roth, E. Kruijff, A. Hinkenjann, A. Pérard-Gayot, P. Slusallek, and Y. Li. Foveated real-time ray tracing for head-mounted displays. In *Computer Graphics Forum*, volume 35, pages 289–298. Wiley Online Library, 2016.

[213] F. Weymouth. Visual sensory units and the minimal angle of resolution. *Elsevier American Journal of Ophthalmology*, 46(1):102–113, July 1958.

[214] C. Wu, R. Zhang, Z. Wang, and L. Sun. A spherical convolution approach for learning long term viewport prediction in 360 immersive video. In *Proc. of the AAAI Conference on Artificial Intelligence (AAAI-20)*, volume 34, pages 14003–14040, June 2020.

[215] Xavier Corbillon. Optimal set of 360-degree videos for viewport-adaptive streaming, 2019. `https://github.com/xmar/optimal-set-representation-viewport-adaptive-streaming`.

[216] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo. 360ProbDASH: Improving QoE of 360 video streaming using Tile-Based HTTP adaptive streaming. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 315–323, Mountain View, CA, October 2017.

[217] L. Xie, X. Zhang, and Z. Guo. CLS: A cross-user learning based system for improving QoE in 360-degree video adaptive streaming. In *Proc. of ACM International Conference on Multimedia (MM'18)*, pages 564–572, Seoul, Korea, October 2018.

[218] S. Xie, Y. Xu, Q. Qian, Q. Shen, Z. Ma, and W. Zhang. Modeling the perceptual impact of viewport adaptation for immersive video. In *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS'18)*, pages 1–5, Florence, Italy, May 2018.

[219] X. Xie and X. Zhang. POI360: Panoramic mobile video telephony over LTE cellular networks. In *Proc. of International Conference on Emerging Networking EXperiments and Technologies (CoNEXT'17)*, pages 336–349, Incheon, Korea, December 2017.

[220] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang. Predicting head movement in panoramic video: A deep reinforcement learning approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):1–14, November 2018.

[221] T. Xu, B. Han, and F. Qian. Analyzing viewport prediction under different VR interactions. In *Proc. of International Conference on Emerging Networking Experiments And Technologies (CoNEXT'19)*, pages 165–171, Orlando, FL, December 2019.

[222] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao. Gaze prediction in dynamic 360 immersive videos. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, pages 5333–5342, Salt Lake City, Utah, June 2018.

[223] Z. Xu, X. Zhang, K. Zhang, and Z. Guo. Probabilistic viewport adaptive streaming for 360-degree videos. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS'18)*, pages 1–5, Florence, Italy, May 2018.

[224] M. Yahia, Y. Le Louedec, G. Simon, and L. Nuaymi. HTTP/2-based streaming solutions for tiled omnidirectional videos. In *Proc. of IEEE International Symposium on Multimedia (ISM'18)*, pages 89–96, Taichung, Taiwan, December 2018.

[225] S. Yao, C. Fan, and C. Hsu. Towards quality-of-experience models for watching 360° videos in head-mounted virtual reality. In *Proc. of International Conference on Quality of Multimedia Experience (QoMEX'19)*, pages 1–6, Berlin, Germany, June 2019.

[226] S. Yen, C. Fan, and C. Hsu. Streaming 360° videos to head-mounted virtual reality using DASH over QUIC transport protocol. In *Proc. of ACM Workshop on Packet Video (PV'19)*, pages 7–12, Amherst, MA, June 2019.

[227] M. Yu, H. Lakshman, and B. Girod. A framework to evaluate omnidirectional video coding schemes. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR'15)*, pages 31–36, Fukuoka, Japan, September 2015.

[228] M. Yu, H. Lakshman, and B. Girod. A framework to evaluate omnidirectional video coding schemes. In *Proc. of IEEE International Symposium on Mixed and Augmented Reality (ISMAR'15)*, Fukuoka, Japan, September 2015.

[229] A. Zare, A. Aminlou, M. Hannuksela, and M. Gabbouj. HEVC-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proc. of ACM International Conference on Multimedia (MM'16)*, pages 601–605, Amsterdam, The Netherlands, October 2016.

[230] Z. Zhang, Y. Xu, J. Yu, and S. Gao. Saliency detection in 360 videos. In *Proc. of European Conference on Computer Vision (ECCV'18)*, pages 488–503, Munich, Germany, September 2018.

[231] H. Zhou, X. Xie, J. Lai, Z. Chen, and L. Yang. Interactive two-stream decoder for accurate and fast saliency detection. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'20)*, pages 9141–9150, June 2020.