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

利用邊緣運算最佳化360度全景影片之頭戴式虛擬實境串流
Edge-Assisted 360-degree Video Streaming for Head-Mounted Virtual Reality

羅文志
Wen-Chih Lo

學號：105062620
Student ID:105062620

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

中華民國 107 年 7 月
July, 2018

Acknowledgments

I would like to express my gratitude toward all the people who helped me in the past two years. I would not be able to finish my thesis were it not without your help along the way. I want to thank my parents specifically, for it is they who provided me with the firm support and stood behind my decisions. I would also like to thank my labmates in Networking and Multimedia Systems Laboratory, especially Ching-Ling Fan, Hua-Jun Hong, and Pin-Chun Wang, who helped me a great deal in the course of my research. Lastly, I would like to express my gratitude toward my adviser: Prof Cheng-Hsin Hsu. Without the guidance and the suggestion I received from him, I would not be able to accomplish what I have done and learned so much.

致謝

在此我要感謝在過去兩年中所有幫助過我的人， 如果沒有你們的幫助我一定沒有辦法順利完成我的論文。 在此我要特別感謝我的父母，他們提供我堅定不移的支持， 同時也支持我所作的每一個決定。我也要感謝網路與多媒體系統實驗室的同學們，特別是樊慶玲、洪華駿、王品淳及其他實驗室同仁在過去兩年中， 不管是研究或人生課題都幫助我非常的多。 最後，我要感謝我的指導教授：徐正炘教授。如果沒有他的給予我的指導以及建議， 在過去的兩年內我一定沒辦法完成如此多的事情以及學到如此多的東西。

Abstract

Over the past years, 360° video streaming is getting increasingly popular. Watching these videos with Head-Mounted Displays (HMDs), also known as VR headsets, gives a better immersive experience than using traditional planar monitors. However, several open challenges keep state-of-the-art technology away from the immersive viewing experience, including high bandwidth consumption, long turn around latency, and heterogeneous HMD devices. In this thesis, we propose an edge-assisted 360° video streaming system, which leverage edge networks to perform viewport rendering. We formulate the optimization problem to determine which HMD client should be served without overloading the edge devices. We design an algorithm to solve the problem as mentioned earlier, and a real testbed is implemented to prove the concept. The resulting edge-assisted 360° video streaming system is evaluated through extensive experiments with an open-sourced 360° viewing dataset. With the assistance of edge devices, we can reduce the bandwidth usage and computation workload on HMD devices when serving the viewers. Also, the lower network latency is guaranteed. We also conduct several extensive experiment. The results show that compared to current 360° video streaming platforms, like YouTube, our edge-assisted rendering platform can: (i) save up to 62% in bandwidth consumption, (ii) achieve higher viewing video quality at a given bitrate, (iii) reduce the computation workload for those lightweight HMDs. Our proposed system and the viewing dataset are open-sourced and can be leveraged by researchers and engineers to improve the 360° video streaming further.

中文摘要

近年來，360度全景影片蔚為流行。諸多廠商也爭相發表各自的頭戴顯示器。 與傳統的平面式螢幕相比，使用頭戴顯示器觀看360度全景影片，提供使用者更好的觀看體驗。 然而，為了實現高品質的沈浸式觀看體驗，有諸多挑戰必須一一克服，例如：網路延遲、頻寬消耗等。 在這篇論文中，我們提出利用邊緣運算360度全景影片之頭戴式虛擬實境串流系統。 此外，我們更設計了一最佳化演算法，此演算法根據不同的使用者採取不同的串流策略，並能有效地降低網路延遲且提高使用者的觀看體驗。 為了驗證本系統及演算法的效能，我們也蒐集並公開了一360度全景影片使用者觀看習慣資料庫，利用此資料庫，我們具體量化了利用邊端運緣360度全景影片之頭戴式虛擬實境串流系統的優缺點。 在與現有的串流系統相比下，我們的系統（一）降低頻寬消耗、（二）提供較好的使用者觀看體驗及（三）降低使用者端的運算資源消耗。 此串流系統及使用者觀看習慣資料庫皆已在網路上開源，希望能幫助到產、學界對此領域有興趣的研究者。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, augmented and virtual reality (AR/VR) technologies are rapidly growing and have attracted attention from both academia and industries. AR/VR have drawn attention to several possible use scenarios, such as gaming, education, and transportation. According to the report from IDC [87], it indicates that more than a billion people worldwide will regularly use and access AR/VR apps, content, and data by 2021. Market research [2] also shows that the AR/VR market will drive 108 billion USD annual revenue by 2021. Watching AR/VR content (e.g., 360° videos) has become one of the most recent trends.

Using traditional planar monitors, however, to watch a live 360° video, especially activities, such as a football match or a music concert is a *passive* experience. Watching these videos with Head-Mounted Displays (HMDs), also known as VR headsets, viewers can dynamically change their viewports during playout time for a better *immersive* viewing experience. Over the past years, AR/VR products, such as HMDs, have become easily accessible. Many companies release their HMDs, such as Oculus Rift DK2 [7], HTC Vive [9], HTC Vive Focus [10], Samsung Gear VR [16], Google Cardboard [8], and Sony PlayStation VR [17]. These products offer viewers wider Field-of-Views (FoVs) and provide a more immersive experience than traditional televisions and monitors. Furthermore, lots of 360° cameras also come to the market, for instance, Ricoh Theta S [14], Luna 360 VR [12], and Samsung Gear 360 [15]. On top of that, major multimedia streaming service providers, such as Facebook [6] and YouTube [18] now support 360° video streaming for VR content, as shown in Fig. 1.1. With the growing popularity of AR/VR goods and services, people are able to watch 360° videos at anytime and from anywhere. By leveraging recent advances in hardware, storage/memory, communication, computing, computer vision, and other adjunct areas will allow novel applications of 360° videos. For example, Facebook spaces [3] allow users to socialize with their friends from different places in a shared virtual room. For retailers, customers may buy clothes online with virtual fitting rooms to save the commute time and expense. In education, HMDs may capture more students' attentions at lower costs for more intuitive classroom experience. Without a doubt, AR/VR may bring a brand new way of working, communication, and entertainment in the near future.

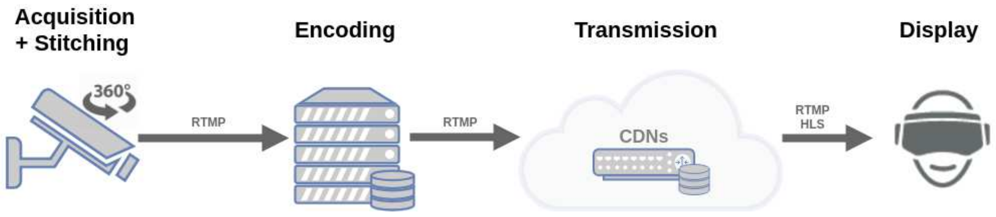


Figure 1.1: Current pipeline of 360° video streaming platform, using YouTube for illustration.

However, there are several open challenges that keep state-of-the-art technology away from immersive viewing experience, including *high bandwidth consumption*, *long turn around latency*, and *heterogeneous HMD devices*. The following are some observations on these three challenges:

- **High bandwidth consumption.** The resolution of 360° videos is commonly above 4K to contain a more comprehensive view than conventional videos, which then increases Internet traffic. However, there is no need to stream whole 360° videos all the time. The viewers never watch the whole 360° videos; instead, they can only rotate his/her head to change the viewing orientation when watching a 360° video and wearing an HMD. Simultaneously, HMD displays the current FoV based on the orientation.
- **Latency-sensitivity.** The 360° viewing experience is extremely latency-sensitive. However, human perception requires accurate and smooth movements, and some researches show that the round-trip latency should be less than 60ms to avoid motion sickness [29, 65]. Severe latency can lead to detached immersive experience and a motion sickness.
- **Heterogeneous HMD devices.** So far there are three types of released HMDs, including i) HMD connected with a PC, like Oculus Rift DK2), ii) HMD connected with a mobile device, such as Samsung Gear VR, and iii) standalone HMD, like HTC Vive Focus. Based on a critical observation [62]: we know that different HMD device demonstrates different computing power and network condition. All three types of HMD devices perform viewport rendering locally. Thus, to accelerate computation and render high-resolution viewports for HMDs, a high-end GPU is needed. Yet, the cables are cumbersome, when HMDs are connected to GPU-equipped PCs. Such standalone HMDs are way heavier because of the extra GPUs

and batteries. The high hardware requirements, however, result in inconvenience and an uncomfortable viewing experience.

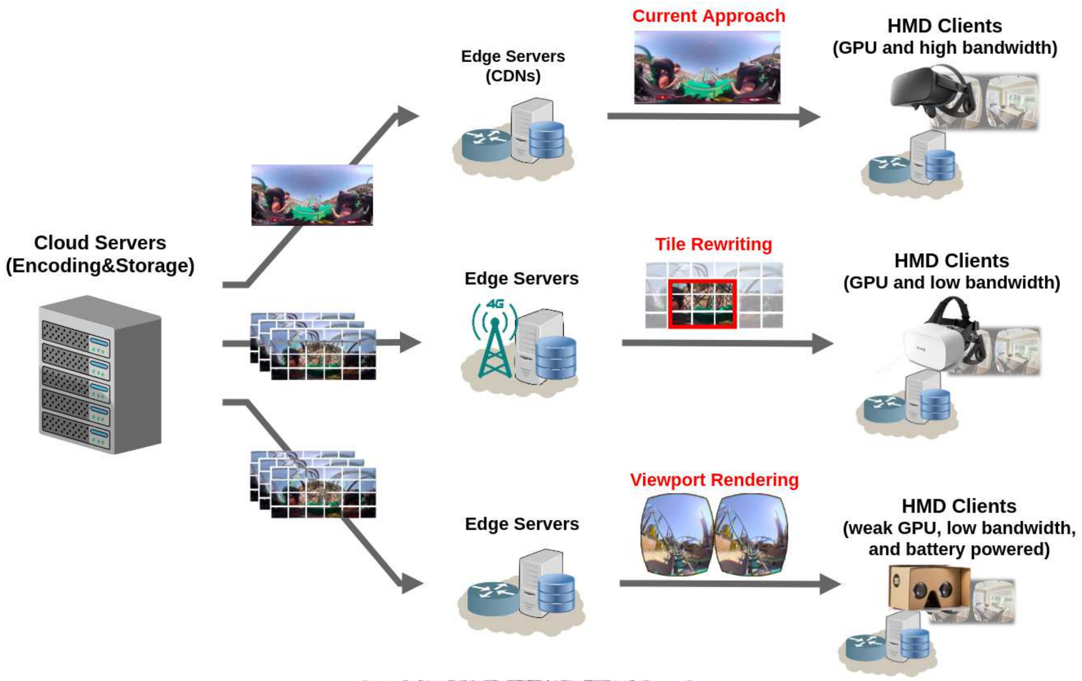


Figure 1.2: 360° video streaming, caching, and rendering with edge networks.

To resolve the challenges, we first study the state-of-the-art streaming pipeline of 360° video [22, 103, 104, 42, 36]. To reduce bandwidth consumption, we will split the 360° videos into tiles of subvideos, which are encoded by modern video codecs, such as HEVC (High Efficiency Video Coding) [95], into video bitstreams, which can be further encoded into multiple various quality at different bitrates. With a variety versions of tiles, HMD client can request and decode those tiles that will be watched in high quality only, and other non-watched tiles will be in low-quality to conserve bandwidth.

Even so, due to the fact that different HMD device has different computing power, we then propose an edge-assisted 360° video streaming system to come across heterogeneous HMD devices. To be more specifically, we will render user's viewport on edge devices, such as base station, access points, and IoT devices, to optimally allocate bitrate and shorten latency. Fig. 1.2 summarizes three different streaming and rendering approaches. Starting from the top is the state-of-the-art 360° video streaming system, which serves as baseline in this thesis. In the middle is the Tile Rewriting (TR) approach. TR approach is suitable for devices with low bandwidth since it streams tiles that will be watched in high quality while the others go in low quality. Viewport Rendering (VPR), at the bottom, on the other hand is compatible for light-weight HMD devices because computation demanding process is offloaded to edge devices. By rendering user's viewport scenes on edge devices, we are able to not only create a lightweight HMD devices which gives users a better viewing experience, but also have guaranteed lower network latency.. However, the computation resource of an edge server is limited. It is challenging to optimally determine which HMDs should be served by edge devices. For example, for those HMDs tethered with a mobile device and standalone HMDs, we tend to perform the viewport rendering on edge devices to reduce the computation workload on those HMDs. Otherwise, for those HMDs connected to GPU-equipped PCs, we are only streaming tiled videos and performing the viewport rendering on HMDs themselves. Fig. 1.2 illustrates from top to bottom: (i) current 360° video streaming, (ii) Tile Rewriting (TR), and (iii) Viewport Rendering (VPR) approach.

To tackle the challenges, we first formulate the optimization problem to determine which HMD should be utilized without overloading the edge devices. We then design an algorithm to solve the aforementioned problems properly, and implemented a real testbed to prove the concept. The resulting edge-assisted 360° video streaming system is evaluated through extensive experiments with an open-sourced 360° viewing dataset [62]. Compared to current 360° video streaming platforms, like YouTube, our edge-assisted rendering platform can: (i) save up to 62% in bandwidth consumption, (ii) achieve higher video quality at a given bitrate, and (iii) reduce the computation workload.

To avoid confusion, Table 1.1 shows the definition of some terms used in this thesis, and their synonyms if necessary.

1.1 Contributions

In this thesis, we study the aforementioned challenges and present our 360° viewing dataset collected from ten YouTube 360° videos and 50 subjects. Also, we develop the edge-assisted 360° videos rendering system adopting and propose the optimization algorithms to perform the 360° video rendering in edge networks. Specifically, in this thesis, we made the following contributions:

- **We study the existing 360° video streaming platforms, discover their limitations, and then improve them by designing a new edge-assisted 360° video streaming system**, which supports TR and VPR approaches, to accommodate the different HMD clients. Our edge-assisted 360° video streaming system can save bandwidth consumption without degrading the video quality while guaranteeing low network latency. Furthermore, by rendering user's viewport scenes on edge devices, we also make HMD devices light weight, which allows users a better viewing experience.
- **We formulate and design an optimization algorithm for the optimal edge-assisted rendering to target HMDs problem**. We then validate the proposed algorithm us-

Table 1.1: Terms and Synonyms Related to 360° Videos Streaming

Term	**Definition**	**Synonyms**
360° Video	A video captures a view in every direction at the same time.	Omnidirectional video, panoramic video, and spherical video
Virtual Reality (VR)	A human-made virtual environment with artificial objects, which can be explored by users.	
Augmented Reality (AR)	Similar to VR, but AR combines some virtual objects with real environments, where users are currently at.	Mixed Reality (MR)
Viewport	A portion of videos that are viewable to a 360° video viewer.	Field-of-View (FoV) and Region-of-Interest (RoI)
Projection	A mathematical transform to convert videos from one space (such as spherical space) to another space (such as 2D planar plane).	Mapping
Head-Mounted Display (HMD)	A display mounted in front of a viewer's eyes for immersive 360° video viewing experience.	VR headset

ing our developed system.

- **We conduct extensive experiments to quantify the performance of our proposed algorithm using the 360° viewing dataset** collected from ten YouTube 360° videos and 50 subjects. We also released the 360° viewing dataset and scripts that we used to calibrate the 360° viewing dataset for any research purposes [62].

1.2 Thesis Organization

The rest of this thesis is organized as followed: we give an introduction and list down the challenges of 360° video streaming in Chapter 1; the background knowledge of 360° videos in Chapter 2; our edge-assisted 360° video streaming system proposal to conquer the aforementioned challenges, the system architecture, detail of the hardware, and software components in Chapter 3; formulations targeting the optimization problem to determine which HMD should be served without overloading the edge devices, and also introduce optimal edge-assisted rendering problem in Chapter 4; collected content and sensor dataset in Chapter 5; extensive experiments to quantify the performance of our proposed algorithm using the 360° viewing dataset in Chapter 6; the related works of

360° video streaming works in Chapter 7; and a conclusion of our contributions and discuss the future works in Chapter 8.

Chapter 2

Background

In this chapter, we briefly introduce the background knowledge of 360° video streaming and cloud/edge computing.

2.1 360° Videos Preprocessing

360° video is usually acquired with multiple camera arrays or a camera device with multiple lenses and image sensors to cover the entire 360° scene with high resolution and frame rate. However, this would burden the storage and transmission of 360° videos. Hence, to stream 360° Video over modern Internet, we have to pre-process the videos, including projecting, tiling, and encoding.

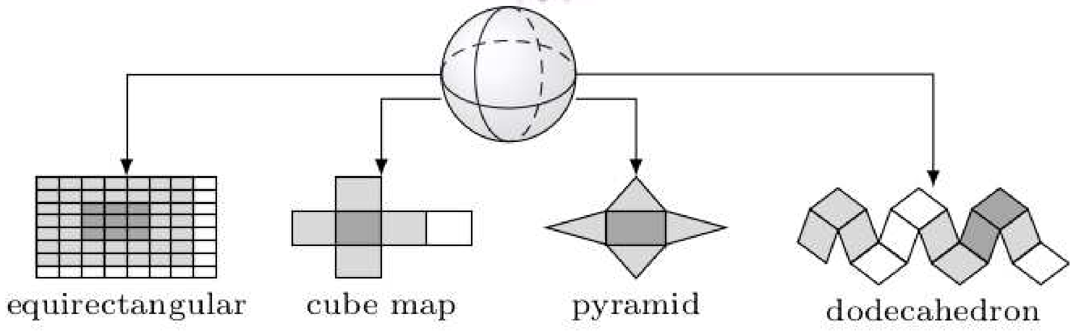


Figure 2.1: Basic planar projections are widely used in the academia and industry.

Planar Projection. To overcome the aforementioned challenges, the use of efficient compression algorithms is inevitable. Several coding standards are available for compressing 360° video sequences, such as Advanced Video Coding (H.264/AVC) [106] and High Efficiency Video Coding (H.265/HEVC) [95]. However, the 360° video is omnidirectional, which means the view in every direction is recorded at the same time. For compressing the omnidirectional videos, a projection onto a two-dimensional (2D) image plane is necessary. Fig. 2.1 illustrates the commonly used projections, including equirectangular projection (ERP), cubemap projection (CMP), pyramid projection (PRP), etc. The coding of omnidirectional content using different projections is widely studied in the literature [23, 44, 57, 68, 98]. Among the other projections we mentioned, the ERP is the most popular format for 360° videos due to its ease of use and wide support in both hardware and software. Thus, on behalf of the ease of use and high portability, we use the ERP as our default projection.

Tile-based Video Coding. As shown in Fig. 2.2, when watching a 360° video, a viewer wearing an HMD would have to rotate his or her head to change the viewing orientation, which can be described by the angles along the x, y, and z axes, or roll, pitch, and yaw respectively. Based on the orientation, the HMD displays the current FoV, which is a fixed-size region, such as a 100°x100° circle. Since a viewer never sees the complete 360° view, streaming the 360° video in its full resolution wastes resources, including bandwidth, storage, and computation. Therefore, we can split each 360° video into grids of sub-images, called tiles[42, 36]. With tiles, an optimized 360° video streaming system to HMDs would strive to stream only the tiles that fall in the viewer's FoV. By doing so, the system satisfies the viewer's needs while consuming less resources than streaming the whole video at its full resolution.

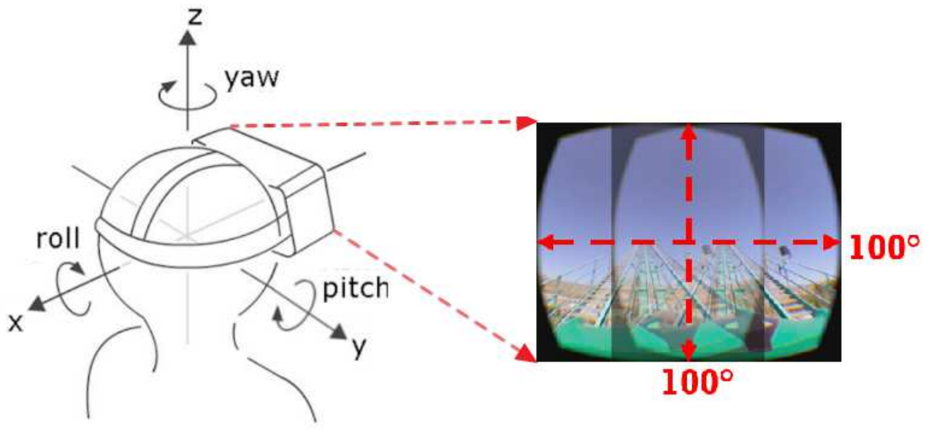


Figure 2.2: A viewer watches a 360° video with an HMD. He or She rotates his or her head to change the viewing orientation and only gets to see a small portion (about 100°x100° circle) of the whole video.

2.2 360° Videos Streaming

Traditional streaming protocols are stateful, such as Real-Time Transport/Streaming Protocol (RTP/RTSP). Once a client connects to a streaming server, the streaming server will keep track of a client's state until it disconnects. Generally speaking, the user and streaming server will have frequent interactions with one other. Once the session between the user and streaming server has been established, the streaming server will continuously send media content as a stream of packets to the users over UDP or TCP protocol.

Dynamic Adaptive Streaming over HTTP (DASH) addresses the weaknesses of RTP/RTSP-based streaming. In contrast to traditional streaming protocol, the HTTP is stateless protocol. One critical difference between these two streaming protocol is that if a stateful stream stops and the system is reset, there is a need to take care of saving the state; on the other hand, stateless streaming protocol does not have any state to save so it's generally simpler. For example, if a client requests some data over HTTP protocol, the server responds and sends the data. Then, the transaction is terminated. Each HTTP request is handled as a standalone one-time transaction. In addition, HTTP is widely supported, only requires a single port, traverse firewalls, media segments can be cached, and sent over Content Distribution Networks (CDNs). DASH streaming generally works by splitting the source files into multiple segments which are then delivered over HTTP protocol. Generally speaking, the same media content can have multiple representations; thus the source file can be encoded for different screen resolutions and with different bitrates (quality levels). These processes enable the adaptive streaming. So when a client first requests the manifest and reads the necessary information then it starts downloading the media segments.

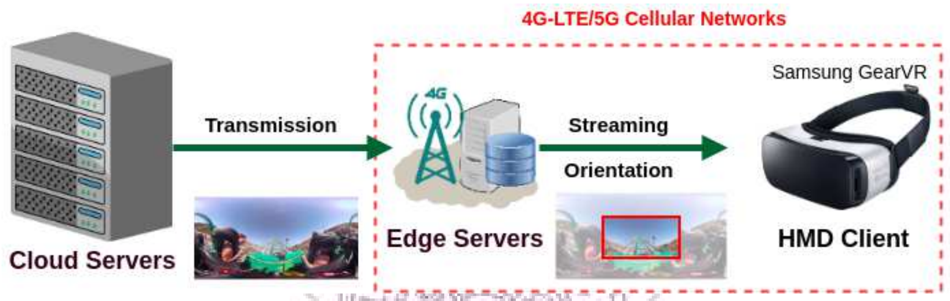
2.3 Edges Computing

Nowadays, cloud computing is getting mature and widely used in multiple areas. However, there are some shortages of cloud computing. For example, cloud computing cannot deal with large amounts of data and satisfy latency-sensitive application, such as 360° video streaming. Transmitting large amounts of data to/from the cloud would result in congestion networks and longer latency.

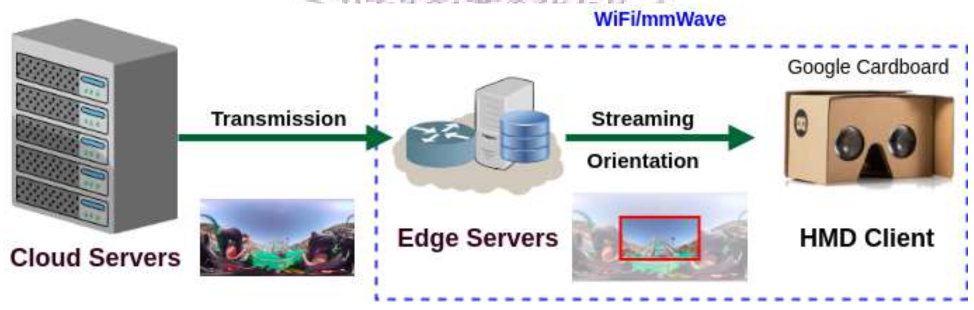
Proposed by Cisco [26], *edge computing* addresses the shortages of cloud computing. Edge computing pushes applications, data, and services away from centralized workstation to the logical extreme (i.e., edge) devices of the Internet, which may spread over a vast area. With the assistance of distributed edge devices, we can use them to carry out computation, storage, communication locally (near the source of the data created), instead of relying on the cloud. With edge computing, we are able to run applications and store data on the edge devices, which are closer to users. It helps reducing latency, saving network resources, and achieving location awareness.

Fig. 2.3 illustrates the two types of edge devices: (a) *remote edge servers*, such as gateways and base stations, delivering data to users over 4G/5G cellular networks, and (b) *local edge servers*, such as access points and IoT devices, delivering data to users over

Wireless Local Area Network (WLAN), such as WiFi and mmWave. The advantage of leveraging edge devices for rendering user's viewport scenes is portability, it allows users to watch 360° videos from any place at anytime without wearing a heavy HMD. Mobility can also be ensured, but it will require the AR/VR applications to support migration between edge servers based on the movement of users. For instance, if the edge server is implemented at a base station of a cellular network (i.e., cell), it could be located far from the users. If the users move from a cell to the neighbor cell, then AR/VR applications have to be migrated to the edge server implemented with the neighbor cell. With the assistance from edge devices, the lower network latency is guaranteed. For example, delivering media data to users from edge servers, which is located closely with each other, can be performed using WLAN and is not expected to have severe network delay.



(a)



(b)

Figure 2.3: Rendering AR/VR content on edge servers then delivered through (a) Cellular Networks and (b) Internet.

2.4 360° Videos Quality Assessments

Video Quality-of-Experience (QoE) refers to the video quality human perceive, which can only be quantified using rigorously designed testbeds and procedures. The QoE metrics are either: (i) *subjective* or (ii) *objective*. The subjective metrics are from user inputs, most likely through some questionnaires; on the other hand, the objective metrics are from computer algorithms. The subjective metrics are better at depicting actual human perception, but it requires more efforts to design, conduct, and analyze. In contrast, the objective metrics are easier to derive, but may deviate from the real human perception. QoE metrics have been crucial for multimedia application. However, the traditional QoE metrics are not suitable for AR/VR content (e.g., 360° videos), due to the planar projection, which we performed. Such as the ERP, it will cause severe distortion at the area near the zenith and nadir, which would result in the inaccurate data.

To conquer the aforementioned challenges, several metrics have been proposed to optimize the 360° videos in terms of QoE levels, such as Viewport PSNR (V-PSNR), Spherical PSNR (S-PNSR), Weighted S-PNSR (WS-PSNR), S-PSNR-NN, WS-PSNR, and CPP-PSNR [112, 113]. Table 2.1 lists and summarizes these proposed objective metrics. In this thesis, we adopt WS-PSNR to be our default objective metrics.

Table 2.1: The Proposed Objective Quality Metrics For 360° Videos

Metrics	**Description**
V-PSNR	Calculate the PSNR based on a point set only fallen in the viewer's viewport.
S-PSNR	Calculate the PSNR of a given set of points that are defined on a sphere surface.
WS-PSNR (chosen)	Calculate the PSNR of a given set of points that are defined on a sphere with different weight on each point.
S-PSNR-I/S-PSNR-NN	Calculate the PSNR of a given set of point with/without interpolation (rounding to the nearest integer).
CPP-PSNR	Calculate the PSNR between different projections both of them are mapped into equal area projection.

Chapter 3

System Architecture

In this chapter, we will give an overview of our proposed edge-assisted 360° video system and introduce the detailed components that contributes the system. Fig. 3.1 illustrates the architecture of our edge-assisted 360° video streaming system. Our edge-assisted 360° streaming system contains three major components: cloud server, edge server, and HMD client. We will introduce each of them below.

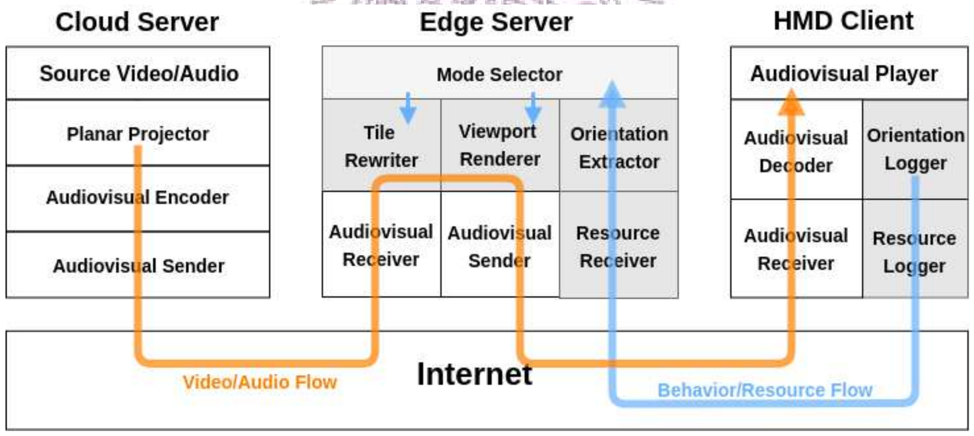
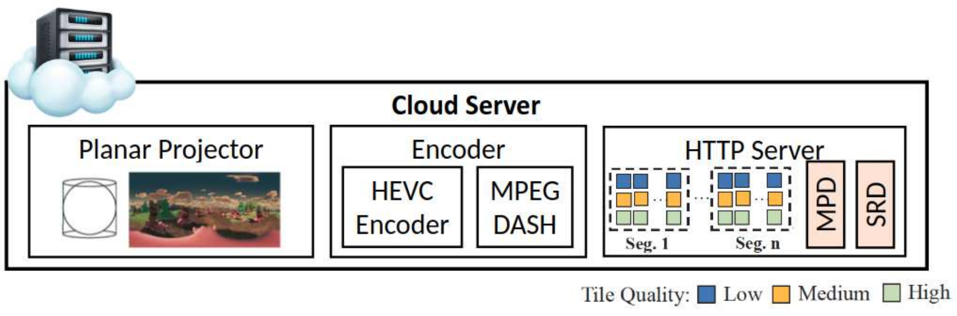


Figure 3.1: The system design of our proposed 360° edge-assisted streaming system.

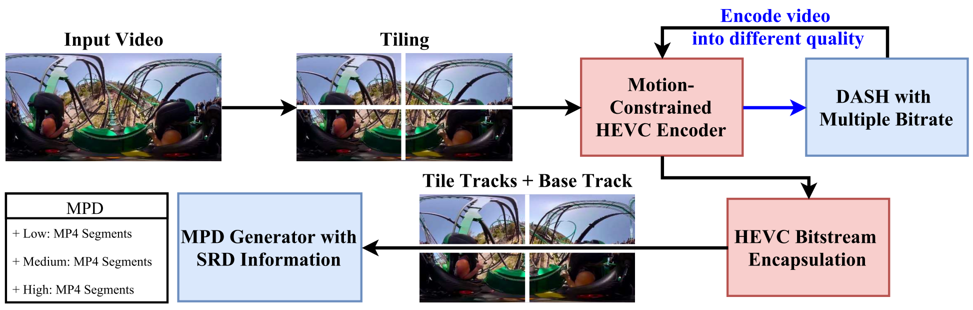
3.1 Cloud Server

We implement three modules inside the cloud server. Fig. 3.2(a) gives diagram of the components on cloud server. The cloud server contains three software components: audiovisual preprocessor, encoder, and sender.

The uncompressed videos are first processed by the audiovisual processor, which includes the stitching and transforming projection format. As we mentioned earlier, we adopt ERP as the default planar projection format. Fig. 3.2(b) gives a processing pipeline



(a)


Input Video
Tiling
Encode video into different quality
Motion-Constrained HEVC Encoder
DASH with Multiple Bitrate
Tile Tracks + Base Track
HEVC Bitstream Encapsulation
MPD Generator with SRD Information
MPD
+ Low: MP4 Segments
+ Medium: MP4 Segments
+ High: MP4 Segments

(b)

Figure 3.2: (a) The implemented modules of cloud server and (b) a video processing pipeline of ERP video.

of ERP video. Each video is spilt into tiles, and encoded using HEVC. Encoding a tiled video using HEVC encoder is a unique process. We summarize the key differences below.

Motion constrained. First, we encode the tiled video with motion constrained. In motion prediction, a tile could refer to data of another tile in a previous or future reference frames, leading to decoding glitches. This might induce some drawbacks on the client: requiring tracking which tiles in a frame are referred to which other tiles in other frames, and downloading the supplementary or non-displayed information for the purpose of reconstructing the tiles correctly. Therefore, Feldmann et al. [40] propose to constrain the tiles encoding so that each tile only refers to the same tiles in previous or future frames, which reduces the complexity on the clients and in the networks.

In-loop Deblocking Filter. Second, we disable the In-loop Deblocking Filter (DBF). HEVC uses an In-loop DBF and a Simple Adaptive Offset Filter (SAO) to improve the decoded video quality. Because of the motion-constrained encoding, the filtering operation may miss some information from the neighboring tiles, which leads to negative side effects. Therefore, the in-loop filter needs to be disabled at the border of tiles to avoid the artifacts.

The coded videos are then streamed independently using MPEG DASH. This works by splitting the source files into multiple segments which are then delivered over the

HTTP protocol. The same content can have multiple representations, which means the source file can be encoded in different screen resolutions and with different bitrates (quality levels). There are two main features enable DASH streaming of 360° videos: *Media Presentation Description* (MPD) [94] and *Spatial Representation Description* (SRD) [70], which are detailed below.

MPD. MPD is an Extensible Markup Language (XML) document that describes an adaptive streaming session. It contains the media segment information, such as timestamp, URL, video resolution, bitrates, and bandwidth restrictions.

SRD. Describing the relationship among tiles, SRD, is a feature extending the MPD of MPEG DASH. It provides additional information to further help DASH clients on determining which tiles to request. SRD puts the media content in a 2D coordinate system, providing the x-axis, y-axis, width, and height attributes. Note that SRD only describes how the content is spatially organized, but does not presume anything on how a DASH client uses this information. This enables the HMD clients to freely select and display only those tiles that are relevant. To stream these tiles, we then implement an Apache HTTP server to be our audiovisual sender.

3.2 Edge Server

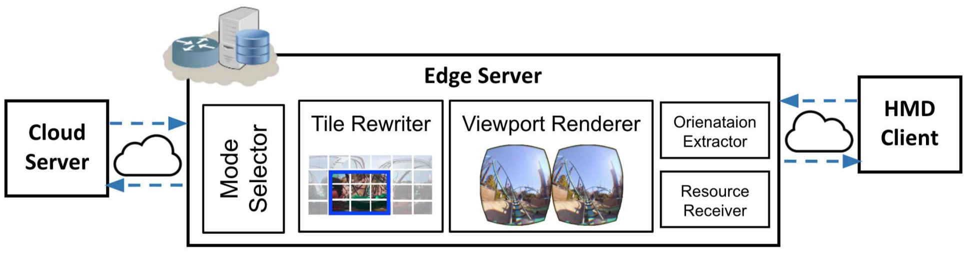
The edge server is the main component of our proposed system, which acts like a rendering approach handler. As emphasized in Fig. 3.1, there are several software components implemented on edge server: Resource Logger/Receiver, Orientation Logger/Extractor, Mode Selector, Tile Rewriter, and Viewport Renderer. We define two types of network flows in the architecture: the behavior/resource flow and the video/audio flow.

Behavior/Resource flow. The behavior/resource flow is consisted of non-audiovisual data. Inside the edge server, the *Resource Receiver* is to get the specifications of HMD client, such as resolution, frame rate, and FoV, which are recorded by *Resource Logger* implemented on the HMD client. The *Orientation Extractor* is to get the viewer's head position and orientation, which are logged by *Orientation Logger* implemented on the HMD client. Then, the *Mode Selector* leverages the collected data to determine which streaming approach (TR or VPR) we should use.

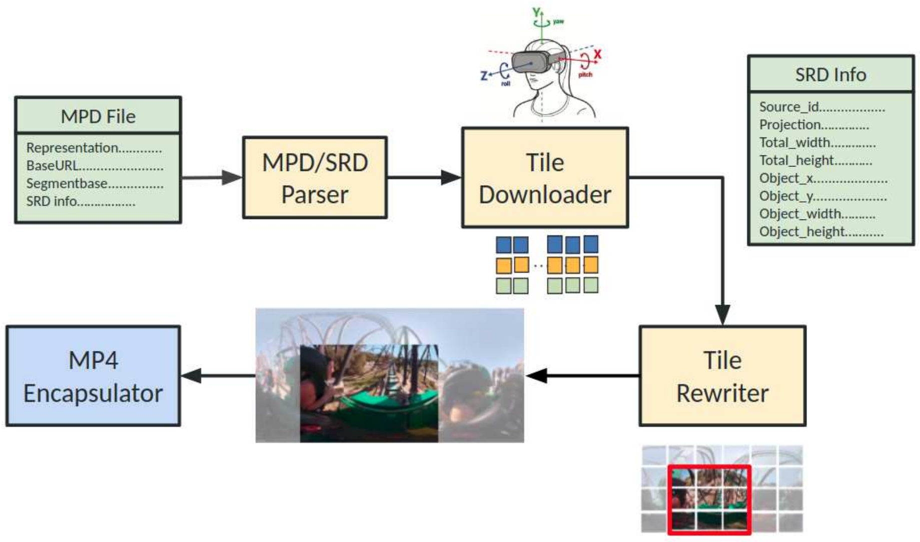
Audio/Video flow. The audio/video flow is consisted of audiovisual data, such as tiles. The tiles are first downloaded from cloud server to edge server. After *Mode selector* determines the streaming approach, the tiles will be processed by *Tile Rewriter* or *Viewport Renderer* appropriately. The processed video content is then encapsulated into a single HEVC bitstream and delivered to the HMD client to generate the viewer's FoV for display.

3.2.1 Tile Rewriting (TR)

Fig. 3.3(b) shows the processing pipeline of TR. It first parses the MPD (with SRD info). According to viewer's orientation, the edge server downloads only those tiled videos that fall in the viewer's FoV from cloud server. Based on the SRD info, **Tile Rewriter** is responsible for concatenating and encapsulating the tiled videos into a single HEVC bitstream. Fig. 3.4 gives a sample clip of TR approach.



(a)



(b)

Figure 3.3: (a) The implemented modules of edge server and (b) TR and VPR streaming pipeline.

3.2.2 Viewport Rendering (VPR)

The edger server downloads only the tiled videos that fall in the viewer's FoV from the cloud server. **Viewport Renderer** is responsible for rendering the viewer's FoV and encapsulating the viewport video into a single HEVC bitstream. Fig. 3.5 gives a sample clip of the VPR approach.

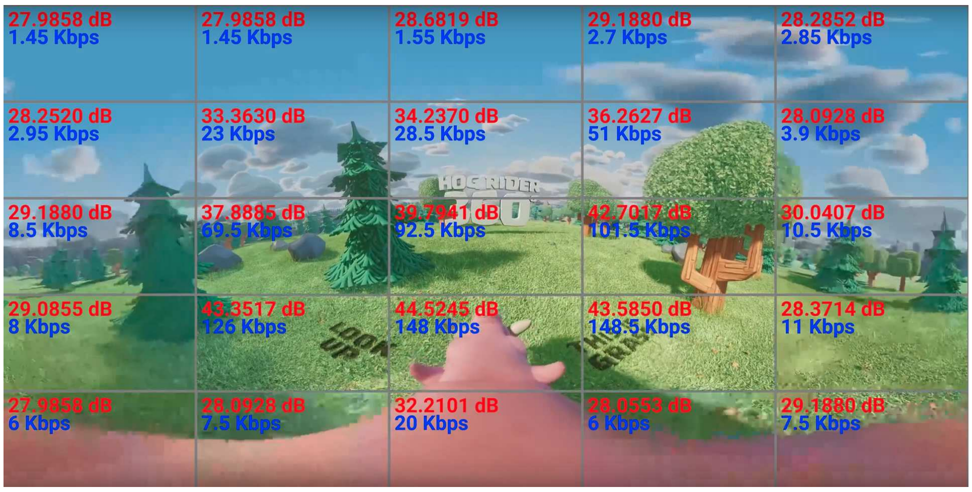


Figure 3.4: A sample clip of TR approach in equirectangular projection.

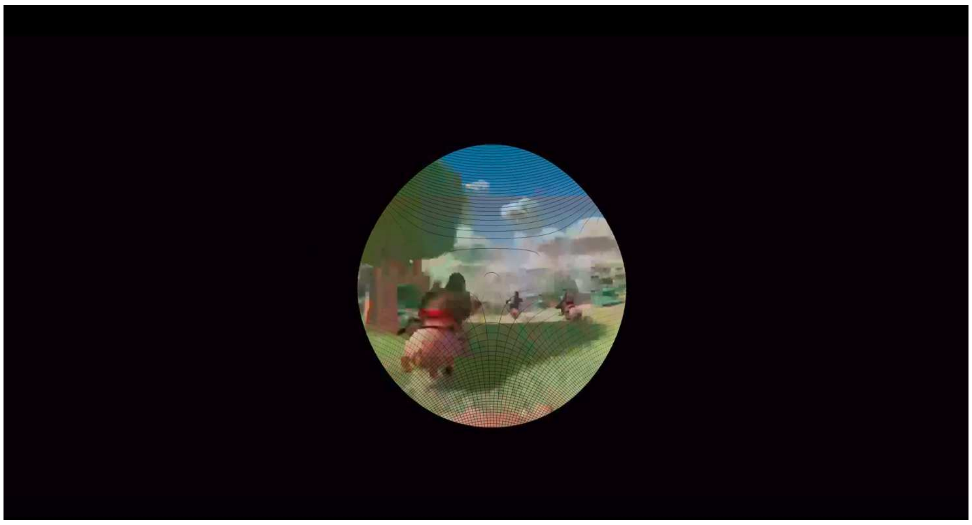


Figure 3.5: A sample clip of VPR approach in equirectangular projection.

3.3 HMD Client

With HMD, each viewer only gets to see a small part of the whole 360° video. Therefore, streaming the entire 360° video in full resolution is a waste of bandwidth. Thus, we leverage a crowd-sourced 360° viewing dataset [62] to select and download only the tiles that fall in the viewers FoV.

A single HEVC decoder is used to decode the received video, including the tiled and viewport ones. Watching 360° videos with 4K UHD resolution or higher is quite challenging due to hardware limitations (slow CPUs or single decoder chip) on some resource-constrained end devices. The tile and viewport video content are encapsulated into a single HEVC bitstream, and thus, only requires a single standard HEVC decoder. The HEVC decoder passes the decoded videos to the audiovisual player to generate the viewer's FoV for display.

Chapter 4

Optimal Edge-Assisted Rendering to Head-Mounted Displays

Different HMDs have varied computing power and network conditions. For example, HMDs tethered to PCs have the luxury of powerful GPUs and high network bandwidth, while standalone HMDs are limited by weak (or zero) GPUs and battery capacity. We consider the problem of capitalizing the edge servers to *assist* HMDs to render scenes. Each edge server however has a limited computing power and network bandwidth. Thus, we must carefully choose the best *ways* to assist individual HMDs for maximizing the overall video quality improvement without overloading: (i) networks, (ii) edge servers, and (iii) HMDs. For example, an edge server should allocate its resources to HMDs with weaker GPU or lower battery levels. In this chapter, we formulate an *edge-assisted rendering* problem to systematically allocate the resources of every edge server and propose an algorithm to solve this problem.

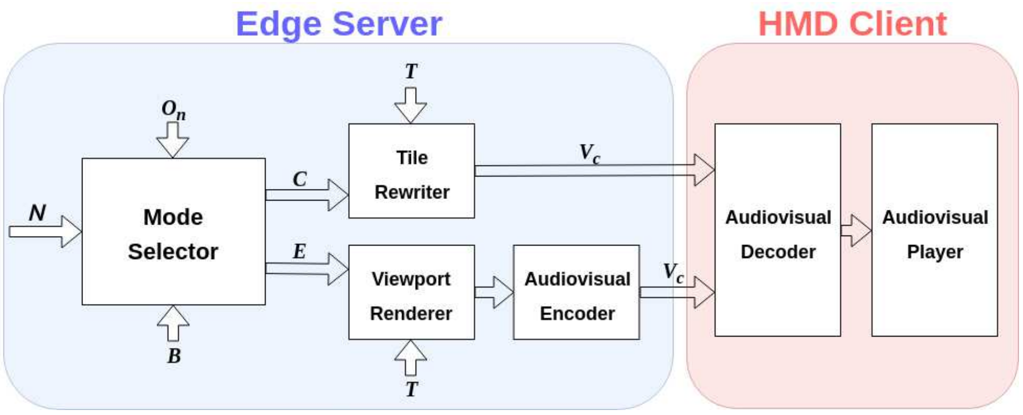


Figure 4.1: Interactions among the core components of our proposed edge server.

4.1 Notations and Models

Table 4.1 summarizes the symbols used in the thesis. Each 360° video is divided into S-second segments, and each segment is cut into T tiles. The tiles then are encoded at two bitrates: b_h for high-quality tiles and b_l for low-quality tiles, where b_h and b_l are system parameters. Let N be the number of HMD clients connected to an edge server. The edge server has an outbound bandwidth of B, and can ender scenes for at most E HMD clients. The viewer of HMD client n ($\forall n = 1, 2, \cdots, N$) has an orientation O_n, where O_n is the central tile index; the FoV of n is given by the width f_n^w and height f_n^h in the unit of tiles.

Tile Rewriting (TR) renders the FoV with higher bitrate b_h while rendering other regions (e.g., out of FoV) with lower bitrate b_l. On the other hand, Viewport Rendering (VPR) only renders the viewport with higher bitrate b_h.

Table 4.1: Symbols Used Throughout This Paper

Symbol	**Description**
N	Set of all HMD clients
n	Index of a HMD client
B	Outbound bandwidth of an edge server
T	Number of tiles
t	Index of a tile
S	Video segment length in second
f_n^w	The width of tiles of HMD client n's viewport
f_n^h	The height of tiles of HMD client n's viewport
$\mathbf{V}_n$	Set of tiles overlapped with the viewer's FoV
b_h, b_l	High/Low encoding bitrate
O_n	Viewer's orientation collected from HMD client n
α_n	Consumed bandwidth of HMD client n for Viewport Rendering (VPR)
β_n	Consumed bandwidth of HMD client n for Tile Rewriting (TR)
q_n	Video quality of HMD client n for TR
q_n'	Video quality of HMD client n for VPR
E	Maximum number of HMD clients that an edge server can serve
x_n	Decision variable of the problem formulation

We build the bitrate and quality model for our edge rendering work. The bitrate model takes encoding bitrate b_h and b_l as inputs, then predicts the consumed bandwidth of HMD client n. We let $f_n^w f_n^h b_h + (T - f_n^w f_n^h) b_l$ be the consumed bandwidth of tile rewriting (TR) and $f_n^w f_n^h b_h$ be the consumed bandwidth for viewport rendering (VPR).

The video quality model predicts the objective video quality of each HMD client, when solving the optimal edge-assisted rendering problem. Let q_n be the video quality model for TR and q'_n be the video quality model for VPR. To derive the model parameters, we encode each tile with high encoding bitrate b_h and low encoding bitrate b_l, and measure the video quality. Using the measured results, we empirically derive the video quality model q_n and q'_n.

4.2 Formulation

Our optimal edge-assisted rendering problem is to determine: (i) the rendering approach for each HMD client n represented by boolean variables $x_n \in \{0, 1\}, \forall n = 1, 2, \ldots, N$, where $x_n = 0$ means TR is selected, and $x_n = 1$ means VPR is selected and (ii) the video quality of HMD client n in order to maximize the overall video quality improvement $q'_n - q_n, \forall n = 1, 2, \ldots, N$.

With the notations define above, our research problem can be written as:

$$\text{maximize} 1/N \cdot \sum_{n=1}^{N} x_n(q'_n - q_n) \tag{4.1a}$$

$$s.\ t.\ \sum_{n=1}^{N} x_n \leq E \tag{4.1b}$$

$$\sum_{n=1}^{N} [x_n(f_n^w f_n^h b_h) + (1 - x_n)(f_n^w f_n^h b_h + (T - f_n^w f_n^h) b_l)] \leq B \tag{4.1c}$$

$$x_n \in \{0, 1\}, \forall n = 1, 2, \ldots, N \tag{4.1d}$$

We let $\alpha_n = f_n^w f_n^h b_h$ and $\beta_n = f_n^w f_n^h b_h + (T - f_n^w f_n^h) b_l$. Eq. (4.1c) can be written as:

$$\sum_{n=1}^{N} [\alpha_n x_n + \beta_n(1 - x_n)] \leq B \tag{4.2a}$$

$$\sum_{n=1}^{N} (\alpha_n - \beta_n) x_n + \beta_n \leq B \tag{4.2b}$$

Consider Eq. (4.2b), our problem can be written as:

$$\text{maximize} 1/N \cdot \sum_{n=1}^{N} x_n(q'_n - q_n) \tag{4.3a}$$

$$s.\ t.\ \sum_{n=1}^{N} x_n \leq E \tag{4.3b}$$

$$\sum_{n=1}^{N} (\alpha_n - \beta_n)x_n + \beta_n \leq B \tag{4.3c}$$

$$x_n \in \{0, 1\}, \forall n = 1, 2, \ldots, N \tag{4.3d}$$

The objective function in Eq. (4.3a) maximizes the average expected video quality considering the given bitrates and adopts rendering approach across all HMD clients. Eq. (4.3b) makes sure that the edge server does not become overloaded when serving HMD clients which adopted VPR approach. Eq. (4.3c) makes sure that the total outbound bandwidth of edge server does not exceed the available bandwidth. The solutions for the formulation in Eq. (4.3a) depend on the given encoding bitrate, and will be discussed in the next section.

4.3 Proposed Algorithm

We proposed an efficient algorithm to solve the optimal edge-assisted rendering problem. The proposed algorithm is selecting HMD clients to do VPR that can boost the objective function Eq. (4.3a) the most. More specifically, it iteratively allocates computing resource to the HMD client with the highest video quality improvement and available bandwidth limitation B. First, we calculate the consumed bandwidth and video quality improvement $q'_n - q_n$. Then, we sort video quality improvement in descending order. Next, we follow this order accordingly to allocate the computing resource E to perform VPR until we reach the limitation of the constraint $\sum_{n=1}^{N} x_n \leq E$. Its pseudocode is given in Algorithm 1. Complexity analysis is given in Lemma 2.

Lemma 1 (Optimality of proposed algorithm) *The proposed algorithm produces optimal video quality improvement if bandwidth constraint Eq. (4.3c) is loose.*

Proof 1

We first make a few observation on bandwidth consumption of TR and VPR approaches, respectively. Based on our prior work [63], we found that if a video is split into smaller tiles, HMD client can download and display the tiles based on viewer's FoV more precisely. Hence, we know the consumed bandwidth of VPR is lower than TR, which leads to the inequality:

Algorithm 1 Mode Selector.

```
1:  // We first initialize variables
2:  for each $n$ in $N$ do
3:      $\mathbf{x}[\mathbf{n}] \leftarrow 0$; $\mathbf{Qual}[\mathbf{n}] \leftarrow q'_n - q_n$; $\mathbf{Band}[\mathbf{n}] \leftarrow \beta_n$; $\mathbf{Rati}[\mathbf{n}] \leftarrow (q'_n - q_n)/(\beta_n - \alpha_n)$
4:  sort $\mathbf{Qual}[\mathbf{N}]$, $\mathbf{Band}[\mathbf{N}]$, $\mathbf{Rati}[\mathbf{N}]$ in desc. order
5:  // Case 1: ideal case, absolute quality boost
6:  select first $E$ HMD clients with the maximal $\mathbf{Qual}[\mathbf{n}]$
7:  for each $e$ in $E$ do
8:      $\mathbf{x}[\mathbf{e}] \leftarrow 1$
9:      $\mathbf{Band}[\mathbf{e}] \leftarrow \alpha[e]$
10: // Check if $B$ is sufficient
11: if sum$(\mathbf{Band}[\mathbf{N}]) \leq B$ then
12:     return $\mathbf{x}[\mathbf{N}]$
13: // If $B$ is insufficient, go to Case 2: per-unit-rate quality boost
14: select first $E$ HMD clients with the maximal $\mathbf{Rati}[\mathbf{n}]$
15: for each $e$ in $E$ do
16:     $\mathbf{x}[\mathbf{e}] \leftarrow 1$
17:     $\mathbf{Band}[\mathbf{e}] \leftarrow \alpha[e]$
18: // Check if $B$ is sufficient
19: if sum$(\mathbf{Band}[\mathbf{N}]) \leq B$ then
20:     return $\mathbf{x}[\mathbf{N}]$
21: // If $B$ is insufficient, go to Case 3: least bandwidth
22: select first $E$ HMD clients with the maximal $\mathbf{Band}[\mathbf{n}]$
23: for each $e$ in $E$ do
24:     $\mathbf{x}[\mathbf{e}] \leftarrow 1$
25:     $\mathbf{Band}[\mathbf{e}] \leftarrow \alpha[e]$
26: // Check if $B$ is sufficient
27: if sum$(\mathbf{Band}[\mathbf{N}]) \leq B$ then
28:     return $\mathbf{x}[\mathbf{N}]$
29: return no feasible solution
```

$$f_n^w f_n^h b_h < f_n^w f_n^h b_h + (T - f_n^w f_n^h b_h) b_l. \tag{4.4}$$

That is, only streaming viewer's viewport leads to increased bandwidth saving, compared to tiled video, as shown in Fig. 4.2, Now, if the bandwidth constraint in Eq. (4.3c) is loose, i.e., the total bandwidth consumption when all HMD clients adopting TR does not exceed available bandwidth B, or $\sum_{n=1}^{n} \beta_n \leq B$. We can readily transform our optimization problem into a 0/1 knapsack problem.

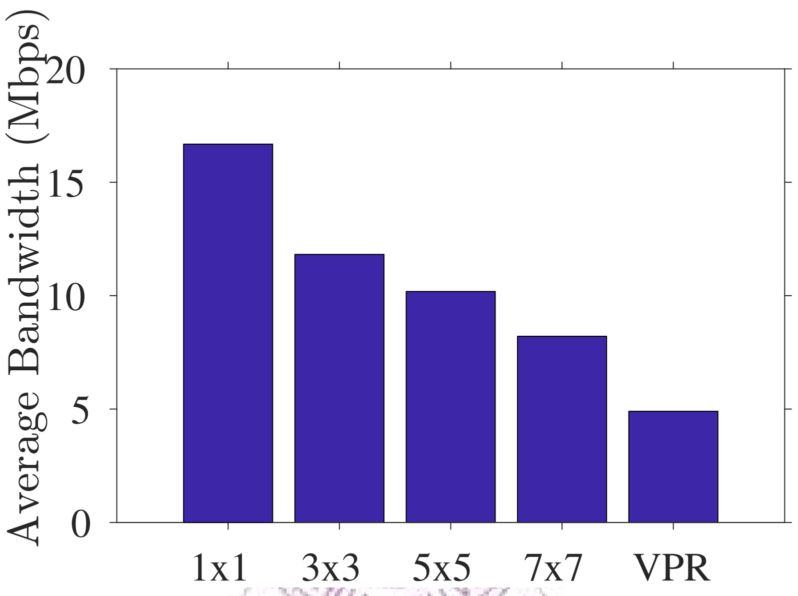


Figure 4.2: The average bandwidth consumption when viewer watching 360° videos, including 1x1, 3x3, 5x5, 7x7 tiles, and only viewport scene.

We then plot the video quality in Fig. 4.3, in which the video quality improvement $q'_n - q_n$ is proportional to the slope. Fortunately, the slope of video quality improvement $q'_n - q_n$ is monotonically decreasing. This means that our algorithm always finds the steepest slope at current step (locally) and across all future steps (globally). Hence, the proposed algorithm produces optimal video quality improvement.

Lemma 2 (Time Complexity) *Our proposed algorithm runs in polynomial time.*

Proof 2

The pseudocode is given in Algorithm 1. Between lines 2–3, we first calculate the consumed bandwidth and video quality improvement for each HMD client n respectively, which has the complexity of $O(N)$. At line 4, we then sort them in descending order, which has the complexity of $O(N \log N)$. Between lines 6–9, we follow the sorted order to pick first E HMD client to allocate the computing resource, which has the complexity

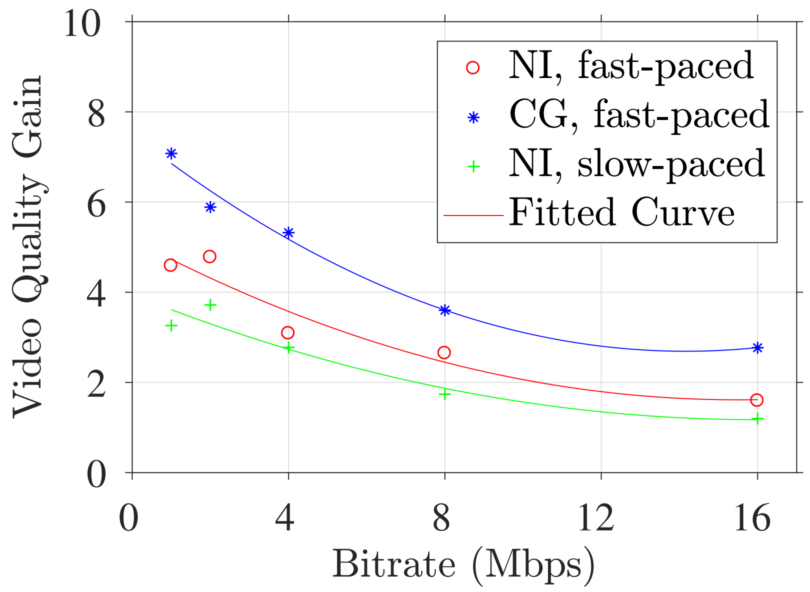


Figure 4.3: Video quality gain if the rendering approach is transformed from TR to VPR.

of $O(E)$. Next, between lines 11–12, we check the overall bandwidth if it exceeds available bandwidth B, which also costs $O(N)$. If the overall bandwidth exceeds available bandwidth B, we repeat the same work according to the ratio of consumed bandwidth and video quality improvement between lines 14–20. The time complexity of this part is $O(E + N)$. If the overall bandwidth still exceeds available bandwidth B, we repeat the same work according to consumed bandwidth only between lines 22–28. The time complexity of this part is also $O(E + N)$. We return no feasible solution after we cannot find a suitable HMD client after the above steps. Hence, the complexity of our proposed algorithm is $O(E + N \log N)$.

Chapter 5

360° Videos Viewing Dataset

Optimizing the 360° video streaming to HMDs is highly data and viewer dependent, and thus dictates real datasets. However, there are no public 360° videos viewing dataset, such as those datasets used in [33, 112] are from the industry and proprietary. To overcome such limitation, and promote reproducible research, we build up our own 360° video testbed 5.2 for collecting traces from real viewers watching 360° videos using HMDs. We then collect content (360° videos) and use the testbed to collect sensor (HMDs worn by viewers) dataset. The resulting dataset can be used to, for example, predict which parts of 360° videos attract viewers to watch the most [39]. The dataset, however, can also be leveraged in various novel applications in a much broader scope. For example, using our dataset, content provider could get to compute the most common FoVs among viewers, and derive the crowd-driven camera movements, which may be used to guide viewers through 360° videos via innovative user interfaces. Through deeper investigations, we could even identify the essential elements for gaining viewers' attentions in 360° videos streamed to HMDs.

Our dataset contains content and sensor data from ten videos available on YouTube [18] and 50 viewers between ages of 20 and 48. More precisely, we analyze the 10 videos to extract the crucial features: the image saliency map [27] that identifies the objects attracting the viewers' attention the most and the motion map [55] that highlights the moving objects. We also log the sensor readings from the HMDs and process them (along with the 360° videos) to derive viewer orientation and viewed tile numbers. Our dataset is unique because we collect both content and sensor data.

5.1 Content Traces

In this section, we describe our content traces. Fig. 5.1 gives sample video frames, image saliency maps, and motion maps from four 360° videos.

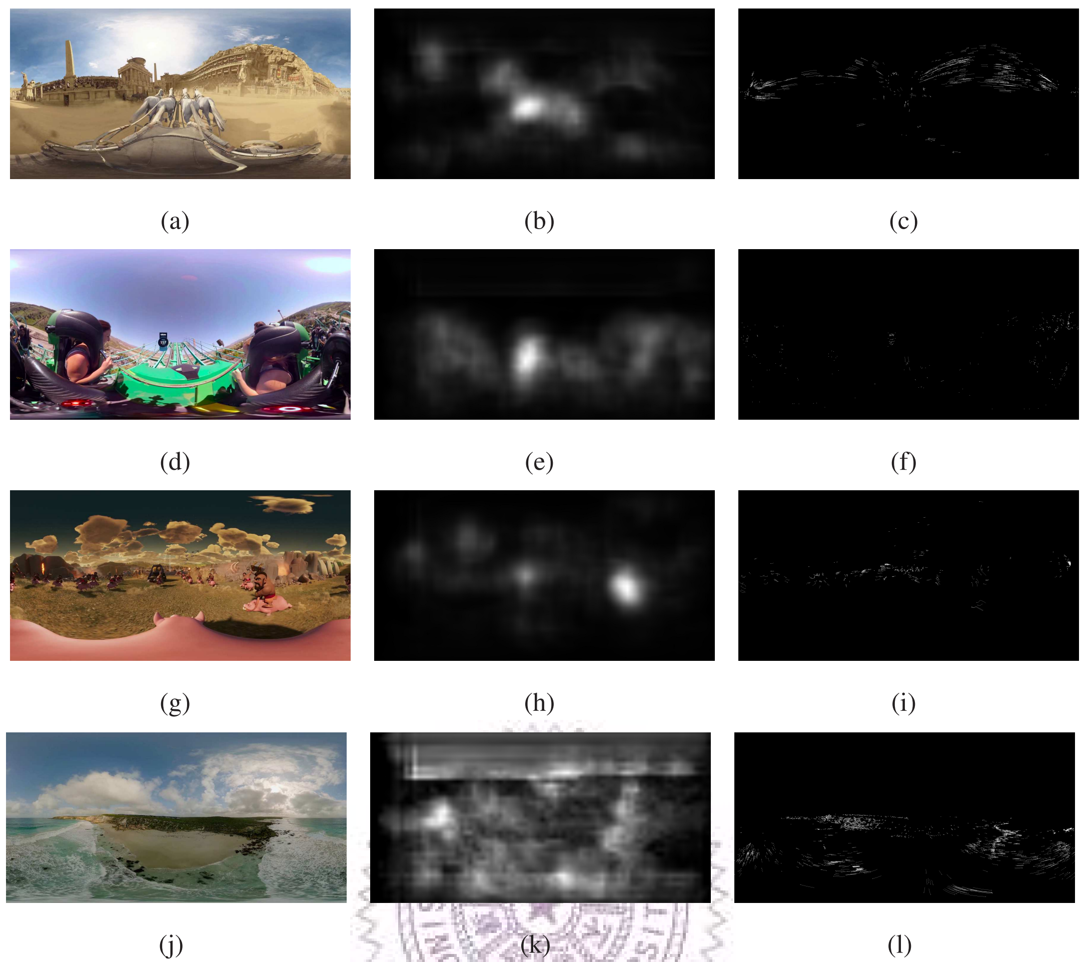


Figure 5.1: (a), (d), (g), (j): sample 360° video frames; (b), (e), (h), (k): image saliency maps; and (c), (f), (i), (l): motion maps. Samples from Chariot Race: (a), (b), (c); from Roller Coaster: (d), (e), (f); from Hog Rider: (g), (h), (i); and from Kangaroo Island: (j), (k), (l).

Video Traces. We collect ten 360° videos with diverse characteristics from YouTube [18]. Table 5.1 summarizes the aspects that we targeted in the 360° videos. All the videos are in 4K resolution at 30 frame-per-second (fps). The videos come in different lengths, so we extract 1-min segment from each of them for experiments. The 360° videos are divided into 3 categories: (i) CG, fast-paced, (ii) NI, fast-paced, and (iii) NI, slow-paced. The 360° videos are encoded and stored in MP4 container files. We do not re-encode the videos, but report their size in Table 5.1. We note that H.264/AVC and H.265/HEVC codecs only support rectangular video frames. Therefore, YouTube adopts the *equirectangular* projection that maps the longitude and latitude of the sphere videos to the horizontal and vertical coordinates of the rectangular video. Although equirectangular projection leads to serious shape distortion (especially when close to the two poles), it is still widely used due to its simplicity.

Table 5.1: Specifications of Ten 360° Videos from YouTube

Category	Videos	Used Segment	Size (MB)	Link
NI, fast-paced	**Mega Coaster**	0:20 - 1:20	160	https://youtu.be/-xNN-bJQ4vI
	Roller Coaster	1:30 - 2:30	153	https://youtu.be/8lsB-P8nGSM
	Driving with	0:48 - 1:48	117	https://youtu.be/LKWXHKFCMO8
NI, slow-paced	**Shark Shipwreck**	0:30 - 1:30	114	https://youtu.be/aQd41nbQM-U
	Perils Panel	0:10 - 1:10	60	https://youtu.be/kiP5vWqPryY
	Kangaroo Island	0:01 - 1:01	126	https://youtu.be/MXlHCTXtcNs
	SFR Sport	0:16 - 1:16	51	https://youtu.be/lo5N90TlzwU
CG, fast-paced	**Hog Rider**	0:00 - 1:00	138	https://youtu.be/yVLfEHXQk08
	Pac-Man	0:10 - 1:10	50	https://youtu.be/p9h3ZqJa1iA
	Chariot Race	0:02 - 1:02	149	https://youtu.be/jMyDqZe0z7M

Saliency Maps. Image saliency maps (see Figs. 5.1(b), 5.1(e), 5.1(h), 5.1(k)) indicate the attraction levels of the video frames. We process the ten 360° videos and generate the image saliency map (as videos) using Convolutional Neural Network (CNN) [89], which is widely used on images and videos. We use a deep neural network [34] based on the pre-trained VGG-16 network [89], which is combined with the weighted features from different levels of CNN. The image saliency map is a gray-scale image (from 0 to 255), varying from black, indicating the least interesting pixels, to white indicating the most interesting pixels. For each 360° video, we first split each video into 1,800 images. Thus, we apply the Keras-based [30] script developed in Cornia et al. [34] to generate the image saliency map. Last, we concatenate the 1,800 image saliency maps into a 1-min video, and encode it using H.264 in MP4 format.

Motion Maps. We analyze the optical flow [64] of the consecutive video frames from each 360° video. Optical flows indicate the relative motions between the objects in 360° videos and the viewers. They can be attributed to either local motions (individual objects move) or global motions (the camera moves), which may catch viewers' attentions. We generate the motion maps (see Figs. 5.1(c), 5.1(f), 5.1(i), 5.1(l)) using OpenCV [1]. The motion maps are in black-and-white images (0 or 1), where a white pixel indicates that the pixel is on one of the optical flows. Then, we specifically each 360° video into 1,800 images, and generate 1,800 black-and-white images using OpenCV-based script. Last, we concatenate these motion maps into a 1-min video, and encode it using H.264 in MP4 format.

5.2 Sensor Traces

We first give an overview on the architecture of our testbed. The 360° video testbed contains four major components: HMD, 360° video player, frame capturer, and sensor logger. Fig. 5.2 shows a photo of our testbed with the four components highlighted. We detail these components in the following.

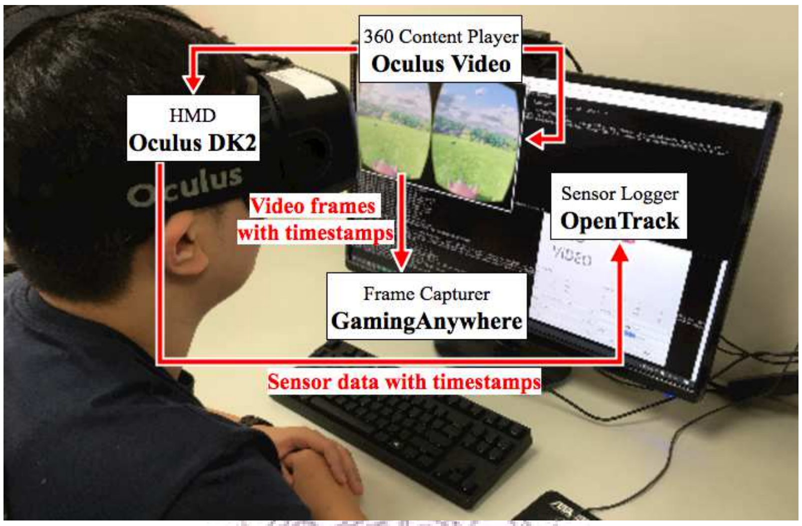


Figure 5.2: A photo of our 360° video streaming testbed. During the experiments, most subjects prefer to stand when watching videos.

HMD. We use Oculus Rift DK2 [7] to be our HMD. We follow the official installation guide from Oculus to set up the hardware and install the Software Development Kit (SDK). This is done on a PC workstation with an Intel E3 CPU, 16 GB RAM, and a NVIDIA GTX 970 GPU.

360° video player. The Oculus Video [4] is an official app from Oculus. We configure it to render 360° videos in both HMD and a mirrored screen. We note that Oculus Video supports equirectangular 360° videos (projected to sphere surface) if the filenames have a suffix of *_360*, e.g., *coaster_360.mp4*. Otherwise, the videos are played as conventional videos instead of 360° ones.

Frame capturer. We use GamingAnywhere [50] as our frame capturer, in order to record the videos rendered to the viewer. The frame capturer stamps each recorded frame with the timestamp, which will be used to align data from various sources. We configure GamingAnywhere to save YUV files at 30 fps.

Sensor logger. We use OpenTrack [73], an open-source head tracking tool, to record the viewer orientations, including yaw, pitch and roll in the range of [-180, 180] from the HMD sensors. In addition, we record and timestamp the viewer positions, including the x, y, and z coordinates. We, however, notice that the 360° video player *ignores* the viewer positions; hence, most viewers in our dataset stay at roughly the same position. Several enhancements have been added by us into OpenTrack project. For example, we enhance the code to save timestamped logs by increasing the granularity of timers to meet our needs.

When a viewer watches a 360° video as shown in Fig. 5.2, the viewer can watch at any orientation by rotating his/her head. The rendered videos on the mirrored screen are captured by frame capturer and stored to disk. The sensor logger records and stores the viewing orientations. Note that the timestamps added by frame capturer and sensor logger are from the same PC workstation. Hence, they can be readily used for alignments.

5.3 Dataset Format

In this section, we present the data format and some statistics of both content and sensor datasets. The content dataset is compressed using H.264, while the sensor dataset is stored as Comma-Separated Values (CSV) files in ASCII.

5.3.1 Content Dataset

The content dataset contains twenty H.264 videos in MP4 container, where each 360° video is analyzed for two content files: (i) the image saliency map and (ii) the motion map. Table 5.2 gives the filenames and sizes of these 20 video files.

5.3.2 Sensor Dataset

```
timestamp, raw x, raw y, raw z, raw yaw, raw pitch, raw roll
1487571103.944, 26.289, 28.063, -15.581, -5.246, -4.298, -1.315
1487571103.953, 26.291, 28.063, -15.567, -5.297, -4.287, -1.333
1487571103.957, 26.292, 28.063, -15.559, -5.323, -4.284, -1.341
1487571103.961, 26.293, 28.063, -15.552, -5.350, -4.277, -1.348
1487571103.965, 26.294, 28.063, -15.545, -5.378, -4.270, -1.354
...
```

Figure 5.3: Sample lines of a raw sensor data log file.

The sensor dataset contains 500 *raw sensor log files*, since we have 50 subjects and ten 360° videos. Fig. 5.3 gives a sample of a raw sensor log file, which contains 7 fields: (i) timestamp, (ii) raw x, (iii) raw y, (iv) raw z, (v) raw yaw, (vi) raw pitch, and (vii) raw roll. In our pilot experiments, we find that different HMD viewers tend to introduce different amount of *bias*. We then introduce a calibration procedure before each viewer

Table 5.2: Content Data Files of Ten 360° Videos

Video	Filename	Size (MB)
Mega Coaster	`coaster2_saliency.mp4`	45.43
	`coaster2_motion.mp4`	42.90
Roller Coaster	`coaster_saliency.mp4`	52.19
	`coaster_motion.mp4`	23.83
Driving with	`driving_saliency.mp4`	43.51
	`driving_motion.mp4`	71.57
Shark Shipwreck	`diving_saliency.mp4`	21.99
	`diving_motion.mp4`	7.99
Perils Panel	`panel_saliency.mp4`	27.07
	`panel_motion.mp4`	1.98
Kangaroo Island	`landscape_saliency.mp4`	60.39
	`landscape_motion.mp4`	57.62
SFR Sport	`sport_saliency.mp4`	28.58
	`sport_motion.mp4`	19.65
Hog Rider	`game_saliency.mp4`	45.94
	`game_motion.mp4`	27.70
Pac-Man	`pacman_saliency.mp4`	33.95
	`pacman_motion.mp4`	5.43
Chariot Race	`ride_saliency.mp4`	49.28
	`ride_motion.mp4`	45.17

starts watching 360° videos. More specifically, we insert a 35-sec calibration video at the beginning of each 360° video. The calibration video sequentially displays an object (cartoon sheep) at (1920, 960), (2880, 480), (2880, 1440), (3840, 960), (960, 480), (960, 1440), and (1920, 96) of coordinates. We show the object at each position for 5 seconds, and we ask the subject to rotate his/her head in order to place the object at the center of their FoV. From this, we average the bias between the captured sensor data and the ground truth from calibration video. Using the bias, we compensate the raw sensor readings (yaw, pitch, and roll) for *calibrated* (cal.) sensor readings (yaw, pitch, and roll).

We note that the sensor data are collected (by OpenTrack [73]) at 250 Hz, and the

```
no. frames, raw x, raw y, raw z, raw yaw, raw pitch, raw roll, cal. yaw, cal. pitch, cal. roll
00001, 16.458 ,30.032, -19.276, -9.661, 5.853, -3.068, -4.65473888889, 4.06641388889, -3.068
00002, 16.458 ,30.032, -19.276, -9.661, 5.853, -3.068, -4.65473888889, 4.06641388889, -3.068
00003, 16.449 ,30.02, -19.362, -9.763, 5.746, -3.184, -4.75673888889, 3.95941388889, -3.184
00004, 16.449 ,30.02, -19.362, -9.763, 5.746, -3.184, -4.75673888889, 3.95941388889, -3.184
00005, 16.433 ,30.007, -19.473, -9.676, 5.659, -3.308, -4.66973888889, 3.87241388889, -3.308
...
```

Figure 5.4: Sample lines of a view orientation log file.

```
1: no. frames, tile numbers
2: 00001, 9, 28, 29, 30, 31, 47, 48, 49, 50, 51, 67, 68, 69, 70, 71, 72, 87, 88, 89, 90, 91, 92
3: 00002, 9, 28, 29, 30, 31, 47, 48, 49, 50, 51, 67, 68, 69, 70, 71, 72, 87, 88, 89, 90, 91, 92
4: 00003, 9, 28, 29, 30, 31, 47, 48, 49, 50, 51, 67, 68, 69, 70, 71, 72, 87, 88, 89, 90, 91, 92
5: 00004, 9, 28, 29, 30, 31, 47, 48, 49, 50, 51, 67, 68, 69, 70, 71, 72, 87, 88, 89, 90, 91, 92
6: 00005, 9, 28, 29, 30, 31, 47, 48, 49, 50, 51, 67, 68, 69, 70, 71, 72, 87, 88, 89, 90, 91, 92
7: ...
```

Figure 5.5: Sample lines of a viewed tile log file.

captured video frames are saved (by GamingAnywhere [50]) at 30 Hz. Users of the raw sensor log files need to align the raw sensor log files with the captured video frames. To simplify the usage of our dataset, we generate *view orientation log files* at 30 Hz by aligning the timestamps in the raw sensor log files and the captured video frames. Moreover, we include the calibrated sensor readings derived above in the view orientation log files. Fig. 5.4 gives a simple view orientation log file, which contains 10 fields: (i) number of frames, (ii) raw x, (iii) raw y, (iv) raw z, (v) raw yaw, (vi) raw pitch, (vii) raw roll, (viii) cal. yaw, (ix) cal. pitch, and (x) cal. roll.

While view orientation log files give the *center* of viewer's FoVs, determining which tiles are needed to render the FoVs require extra calculations; thus, we assume the FoVs are modeled by 100°x100° circles. Therefore, we process the view orientation log files, and generate *viewed tile log files* to further simplify the usage of our dataset. For all 360° videos, we divide each frame, which is mapped in equirectangular model, into 192x192 tiles, so there are 200 tiles in total. Then we number the tiles from upper-left to lower-right. Fig. 5.5 gives a sample viewed tile log file, which contains 2 fields: (i) number of frames and (ii) tile numbers. Each tile number determines a unique tile of the entire 360° video, and an FoV overlaps with multiple tiles as shown in this figure.

Last, Table 5.3 gives filenames of sensor data. For each video and each user, there are three sensor data files for: (i) raw sensor, (ii) view orientation, and (iii) viewed tiles. This table also reports the total size of these log files stored in ASCII format.

Table 5.3: Sensor Data Files of 50 Subjects

Video	Sample Filename (User 0)	Total Size (MB)
Mega Coaster	coaster2_user00_raw.csv	224.93
	coaster2_user00_orientation.csv	16.22
	coaster2_user00_tile.csv	20.16
Roller Coaster	coaster_user00_raw.csv	228.66
	coaster_user00_orientation.csv	16.14
	coaster_user00_tile.csv	19.75
Driving with	drive_user00_raw.csv	226.99
	drive_user00_orientation.csv	15.89
	drive_user00_tile.csv	19.78
Shark Shipwreck	diving_user00_raw.csv	224.69
	diving_user00_orientation.csv	16.19
	diving_user00_tile.csv	19.03
Perils Panel	panel_user00_raw.csv	229.45
	panel_user00_orientation.csv	16.13
	panel_user00_tile.csv	20.42
Kangaroo Island	landscape_user00_raw.csv	228.71
	landscape_user00_orientation.csv	16.29
	landscape_user00_tile.csv	20.41
SFR Sport	sport_user00_raw.csv	225.77
	sport_user00_orientation.csv	15.86
	sport_user00_tile.csv	21.97
Hog Rider	game_user00_raw.csv	230.10
	game_user00_orientation.csv	15.79
	game_user00_tile.csv	19.78
Pac-Man	pacman_user00_raw.csv	218.60
	pacman_user00_orientation.csv	16.02
	pacman_user00_tile.csv	20.32
Chariot Race	ride_user00_raw.csv	230.65
	ride_user00_orientation.csv	15.79
	ride_user00_tile.csv	19.55

Chapter 6

Evaluations

In this chapter, we conduct extensive experiments to quantify the performance of our edge-assisted 360° streaming system described in Chapter 3.

6.1 Implementations

We use the 360° videos from a public dataset [62] for experiments. There are 10 video sequences in this dataset, which are classified into three categories: (i) Natural Image (NI) fast-paced, (ii) NI slow-paced, and (iii) Computer-Generated (CG) fast-paced. For ease to conduct the experiments, we indexed the video sequences as follow: 1, 2, 3 is NI fast-paced; 4, 5, 6, 7 is NI slow-paced, and 8, 9, 10 is CG fast-paced. The dataset also contains video viewing data, such as users' orientation, from 50 subjects. We split the dataset into two groups: (i) training set (40 subjects' viewing data) for build the system model and (ii) test set (10 subjects viewing data) for conducting the experiments.

We leverage an Intel 40-cores workstation with 256 GB of RAM as cloud server. Cloud server contains several software components, including an HEVC encoder, an MPEG DASH content generator, and an Apache HTTP server. We use an open-source codec, Kvazaar [102], to slice and encode the videos into multiple tiles (subvideos). Then, we package our raw HEVC bitstream, cut the video into small segments with a few seconds, and generate MPD file using MP4Box [13].

We set up a PC workstation with Intel i7 CPU and 16 GB RAM as edge server. We first analyze and generate the image heatmap (as videos) using training set. The heatmap is a gray-scale image (from 0 to 255), varying from black, indicating the least viewing pixels, to white indicating the most viewing pixels. We can calculate the weighted Spherical PSNR (WS-PSNR), based on the viewing heatmap as a video quality model, which predicts the overall video quality improvement of each HMD client. With this video quality model, the edge server is able to allocate its resources to maximize the overall video quality improvement. Also, we implement the OSVR Render Manager [110] to be our viewport renderer. We then implemented our proposed algorithm using Python. The edge-assisted 360° system has been written in C and Python. At the client side, we use Samsung Gear VR tethered on an mobile phone Samsung S6 Edge+ to download and render videos. We also installed PowerTutor [115], an application for mobile devices, which measures the power consumed by major system components, on our Samsung S6 Edge+.

For comparisons, we also implemented the baseline algorithm using CPLEX [11], which is the optimal solution solver developed by IBM. CPLEX solver gives us the maximal video quality improvement in a given bandwidth and computing power. For clarity, we will refer to the current 360° video streaming approach, CPLEX-based algorithm, and proposed algorithms as CUR, OPT and PRO, respectively.

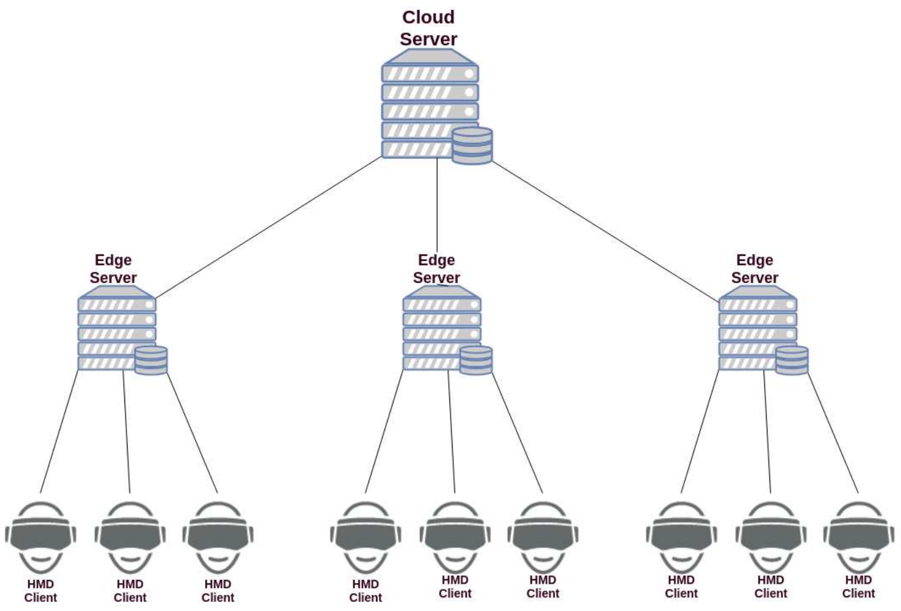


Figure 6.1: Network topology.

6.2 Setups

We randomly choose 40 viewing traces form the test set, which contains 100 viewing traces (10 subjects watched 10 video sequences), to simulate the real users watching 360° video. Fig. 6.1 is an example of the topology used in our experiments. We compare our edge-assisted solution against the current 360° video streaming platform, in which all the 360° video are non-tiled and projected in equirectangular projection. All videos are 60 seconds long, in 4K solution, and 30 fps (frame-per-second). We fix the number of tiles in 5x5, segment length at 2 sec, and the high and low encoding bitrate as 8Mbps and 1Mbps, respectively. The 360° videos are tiled and compressed by the state-of-the-art codec, such as Kvazaar [102], on cloud server and streamed to the edge server. Edge server then acts likes a rendering approach handler and determines which HMDs should be served by edge devices. We assume that the network bottleneck is between edge server and HMD client. We consider the outbound bandwidth of the networks above as 1, 2, 4 Gbps, following the AWS network benchmark reports [5].

We consider the following performance metrics in our experiments.

- **Consumed bandwidth**: the required transmission bandwidth for delivering data to clients.
- **Video quality**: We adopt the Viewport PSNR (V-PSNR) [112] as video quality metric, which calculates the Peak Signal-to-Noise Ratio (PSNR) for the viewer's FoV (instead of the whole 360° video).
- **Latency**: we measure latency in three different categories, including rendering delay (RD), propagation delay (PD), and transmission delay (TD).
- **Power consumption**: the consumed power (mW) for downloading data and render the user's viewport scenes.

We set the total number of HMD clients to be 40. We vary the maximum number of HMD clients that an edge server can serve E = 5, 10, 15 20, 25 and record their consumed bandwidth, latency, and video quality.

6.3 Results

In this section, we report the experiment results and analysis. Also, we hold a 95% confidence intervals whenever applicable.

Our proposed system results in lower bandwidth consumption. Fig. 6.2 plots the consumed bandwidth of CUR (non-tiled video) and our proposed edge-assisted streaming system. It shows that our proposed solution consumes less bandwidth for all 10 video sequences in the public dataset. This is because we only stream parts of video (both TR and VPR) to HMD clients, instead of streaming the entire 360° video. Our proposed algorithm even saves consumed bandwidth as much as OPT did. Compared to CUR, our proposed algorithm can reduce min/avg/max = 31%/64%/78% bandwidth consumption.

Next, we increase the edge server capacity, where $E = 5, 10, 15, 20, 25$. Fig. 6.3 shows the average consumed bandwidth of CUR, OPT, and PRO. The result shows that both OPT and PRO save more bandwidth consumption, when edge server capacity is increasing. As we mentioned in Sec. 3, VPR only streams user's FoV to HMD client, which leads to less bandwidth consumption than TR. The higher edge server capacity (i.e., more HMD clients that edge can serve), the more bandwidth consumption we can save.

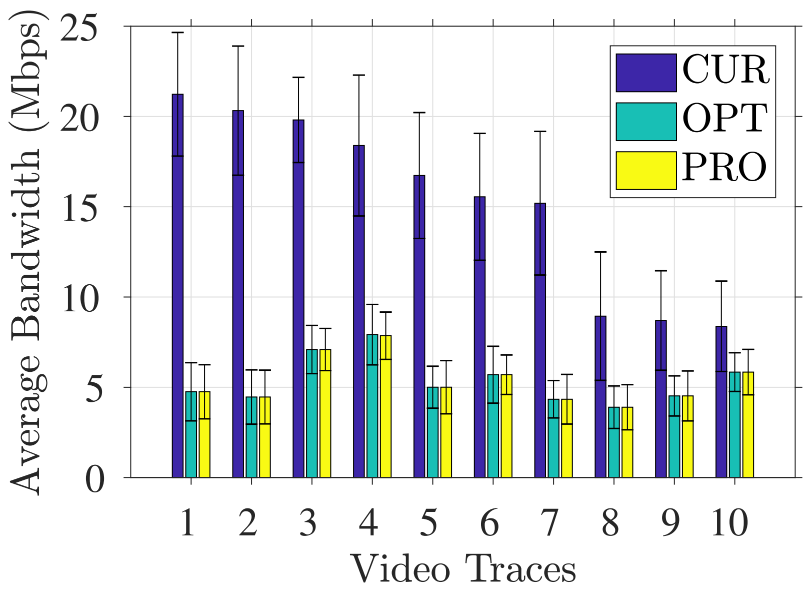


Figure 6.2: Consumed bandwidth of CUR, OPT, and PRO, when edge server capacity E = 10.

Generally speaking, both of OPT and PRO can save at least 35% (up to 62%) average bandwidth consumption, compared to CUR. Hence, only streaming tiled or viewport of 360° videos over modern networks reduces the amount of data transfer and Internet traffic amount.

Our proposed system results in higher video quality. Fig. 6.4 presents the video quality achieved by OPT and PRO, compared to CUR. This figure shows PRO outputs better video quality for all 10 video sequences. The min/avg/max of video quality improvement is 6/7.4/8.4 dB.

Fig. 6.5 reports the video quality of CUR, OPT, and PRO, when the edge server capacity is increasing, where E = (5, 10, 15, 20, 25). Both OPT (as baseline algorithm) and PRO algorithms continue delivering high video quality (V-PSNR ≥ 40) with enough outbound bandwidth of edge server in V-PSNR over time. Fig. 6.6 plots the video quality improvement of OPT and PRO algorithms. It shows that both OPT and PRO algorithm achieve at most 9.5 dB video quality gain at a given bitrate. These numbers indicate that the proposed algorithm produces the optimal video quality improvement (as proved in Chap. 4), assuming that the outbound bandwidth of an edge server is high enough ($\geq 1Gbps$).

Fig. 6.7 plots the video quality achieved by OPT and PRO algorithms, when the outbound bandwidth of an edge server is low ($\leq 0.3Gbps$). This figure shows the OPT algorithm outperforms our proposed algorithm by at most 4 dB. Fig. 6.8 presents the video quality improvement of OPT and PRO algorithms. OPT algorithm outperforms our proposed algorithm when edge server can serve more than 15 HMD clients. However, when

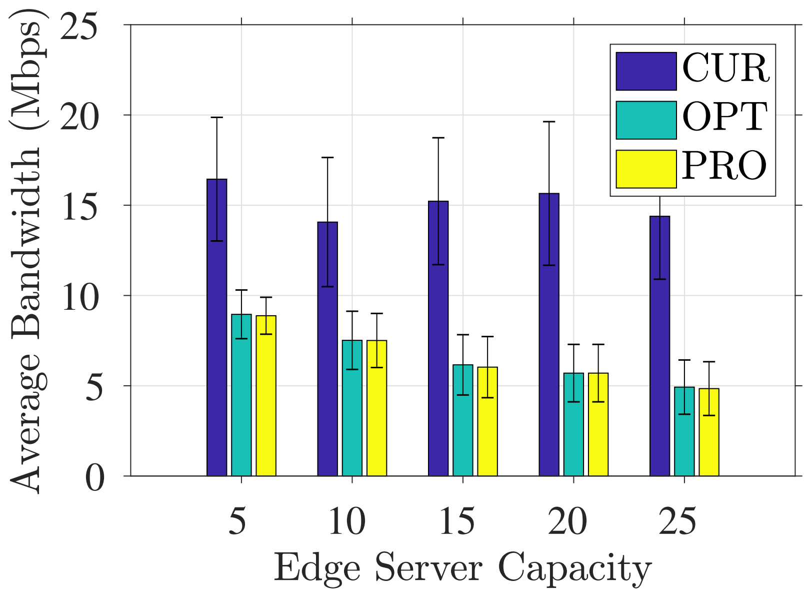


Figure 6.3: Consumed Bandwidth of OPT and PRO, compared to CUR, when edge server capacity is increasing, where E = (5, 10, 15, 20, 25).

the outbound bandwidth of edge server is not enough, the OPT algorithm suffers from exponential running time as shown in Table 6.1. It is not suitable to real-time systems like 360° video streaming platforms. Our proposed algorithm runs in polynomial time (proved in Sec. 4.3). Hence, our proposed algorithm outputs good video quality without overloading the edge server and consuming lots of time.

Table 6.1: Running Time (s) of OPT and EDD, when there are 40 HMD clients and outbound bandwidth $B = 0.3$ Gbps.

Edge Server Capacity	**5**	**10**	**15**	**20**	**25**
OPT	5.41	375.54	533.57	55.29	1.02
PRO	0.19	0.20	0.36	0.41	0.52

The latency of our proposed system is lower. We run our edge-assisted 360° streaming system on Amazon AWS service. We set up the edge servers at Asia/Tokyo (TYO), Asia/Seoul (SEL), Asia/Singapore (SIN), Asia/Sydney (SYD), and Asia/Mumbai (BOM), respectively. The cloud servers are set at US/California (CA), US/Virginia (VA), CAN/Montreal (YMQ), EU/Frankfurt (FRA), and EU/London (LON). We measure the network latency between 1 a.m. to 2 a.m. every day. Network latency is consisted of three different categories, including rendering delay (RD), transmission delay (TD), and propagation delay (PD). RD is the time it takes to render user's viewport scenes and encapsulate into a MP4 container. TD is the time it takes to push packet's bits onto the network. PD is the time it takes for a packet to travel between one place and another at the speed of light. Table. 6.2

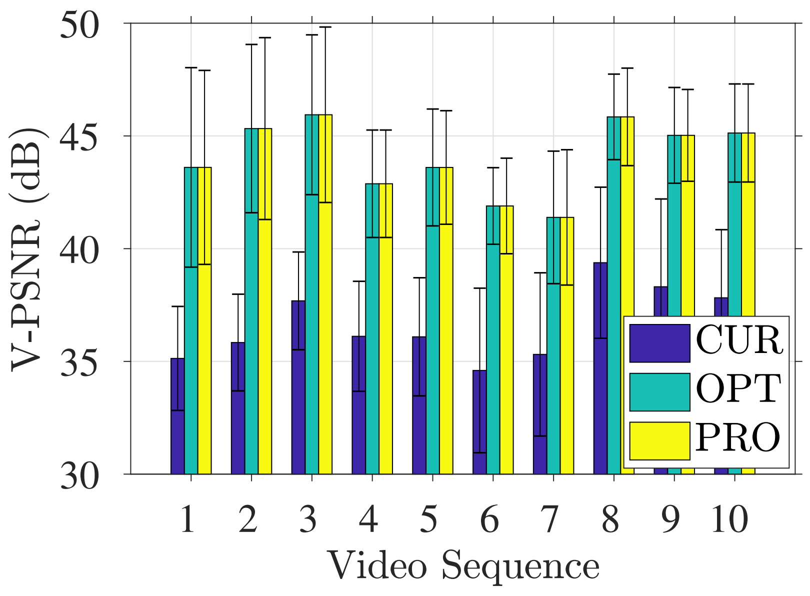


Figure 6.4: Video quality in Viewport-PSNR of OPT and PRO, compared to CUR, where edge server capacity E = 10 and available outbound bandwidth B = 1 Gbps.

reports RD of each video sequences from 360° video viewing dataset. The average FPS and rendering time per frame of our edge-assisted 360° video streaming system is 26.032 and 38.414 ms, respectively.

We plot the latency of with/without edge-assisted rendering in Fig. 6.10 and Fig 6.9. Fig. 6.10 shows that 360° videos are streamed from cloud server to HMD clients, without the assistance from edge devices. Fig. 6.9 shows that 360° videos are first delivered from cloud server to edge server, and streamed to HMD clients. These two figures reveal that lower latency is achieved with the assistance from edge devices.

Our proposed system saves the battery of HMD clients. We measure the power consumption of HMD client using PowerTutor [115]. Power consumption is consisted of three different categories, including computing, communicating, and displaying power. We denote them as CPU, Comm, and Display, respectively. Fig. 6.11 reports the power consumption of the current streaming approach, TR, and VPR. This figure shows that the screen consumes most of battery power by up to 700 mW in average. The consumed computation power of CUR, TR, and VPR is 66.1, 46.2, and 26.6 mW, respectively. Also, the consumed communication power of CUR, TR, VPR is 175.6, 135.8, and 94.2 mW, respectively. These indicate that performing VPR on edge server consumes less battery power of HMD client than CUR and TR. Because we offload the rendering works from HMD client to edge server and only deliver the user's viewport scenes to the HMD client. Fig. 6.12 plots the power consumption of CUR, OPT, and PRO, which are normalized to CUR. Compared to CUR, PRO reduces min/avg/max 7.1%/8.9%/10.5% power consumption of the HMD client. By performing user's viewport rendering and only streaming

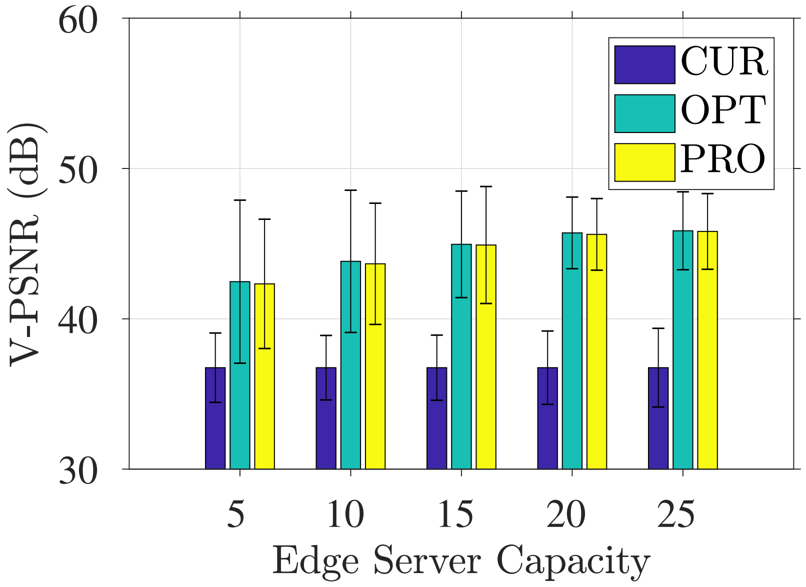


Figure 6.5: Video quality in Viewport-PSNR of OPT and PRO algorithm.

parts of 360° video (e.g., TR or VPR), our edge-assisted 360° streaming system reduces the power needed to be consumed for a HMD client.

In summary, when available outbound bandwidth of an edge server is high ($B \geq 1$ Gbps), our proposed algorithm can produce optimal video quality improvement, which is up to 8.4 dB. When available outbound bandwidth is low ($B \leq 0.3$ Gbps), our proposed still achieves a good video quality improvement without consuming lots of time. Our edge-assisted 360° video streaming system achieves lower network latency, reduces the computation/communication loading, and extends the battery life of HMD client.

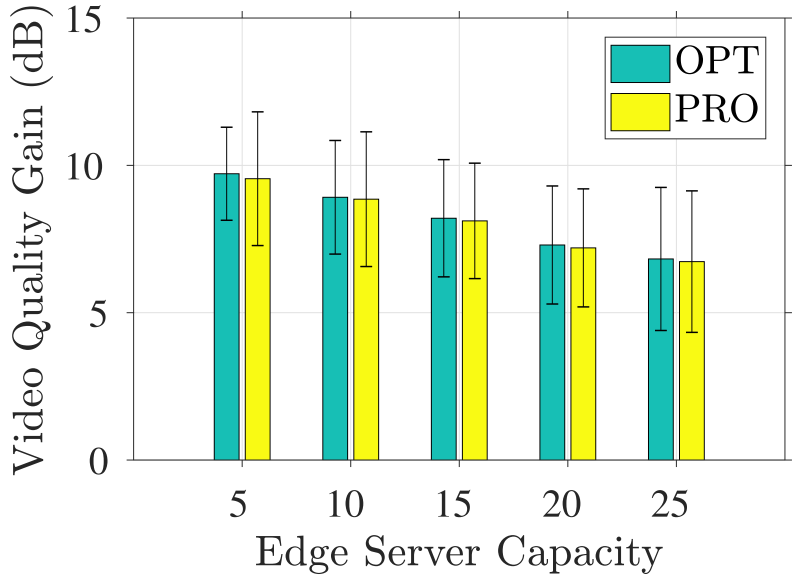


Figure 6.6: Video quality improvement of OPT and PRO algorithms.

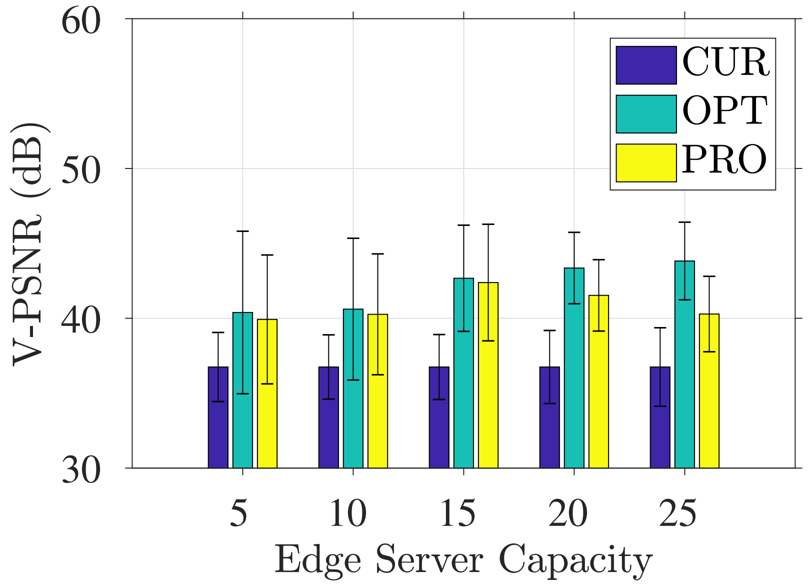


Figure 6.7: Video quality improvement of OPT and PRO algorithms.

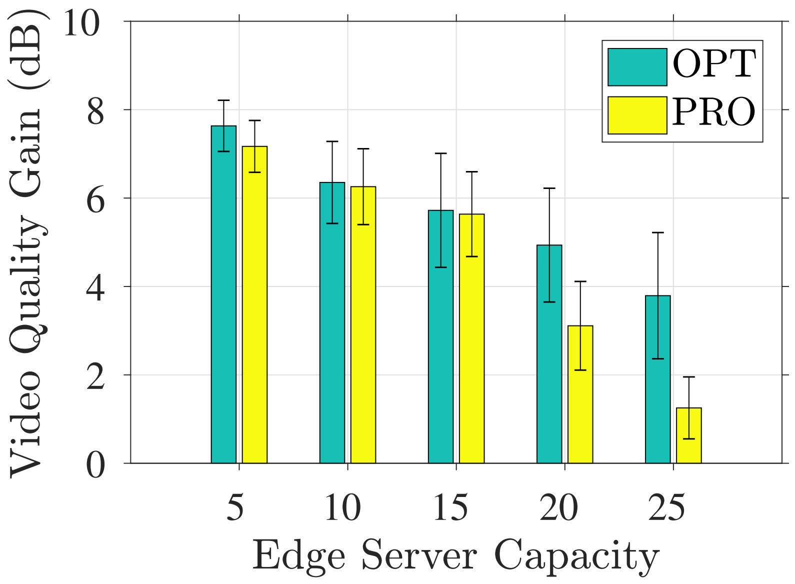


Figure 6.8: Video quality improvement of OPT and PRO algorithms.

Table 6.2: Rendering Time (ms) of each video sequences from 360° video viewing dataset.

Video Sequences	**FPS**	**Time per Frame (ms)**
Mega Coaster	21.22	47.13
Roller Coaster	21.22	47.13
Driving with	29.61	33.78
Shark Shipwreck	26.47	37.78
Perils Panel	29.99	33.33
Kangaroo Island	24.34	41.09
SFR Sport	29.80	33.56
Hog Rider	25.83	38.72
Pac-Man	29.99	33.33
Chariot Race	21.85	45.76
Avg.	26.03	38.41

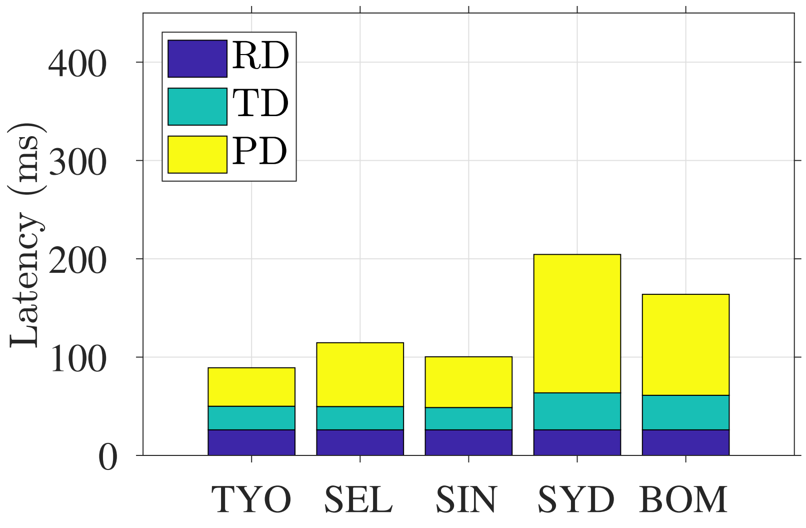


Figure 6.9: Latency of edge server and HMD client.

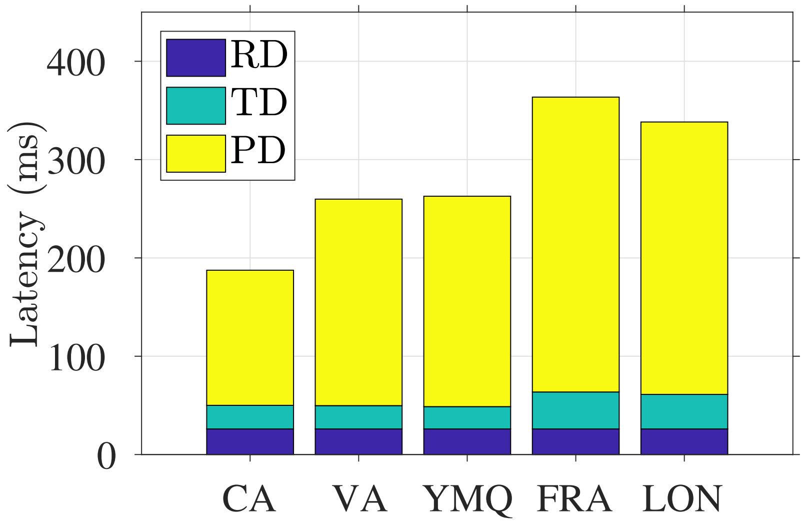


Figure 6.10: Latency of cloud server and HMD client.

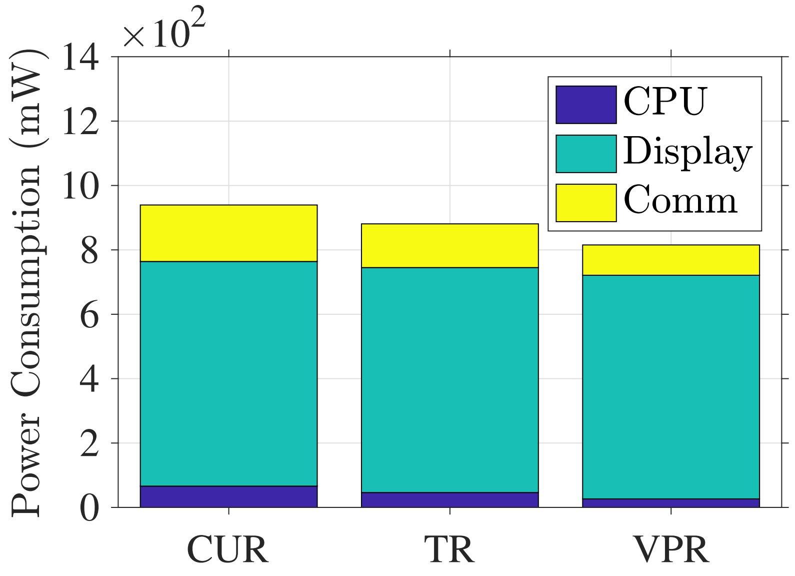


Figure 6.11: Power consumption (mW) of CUR, TR, and VPR.

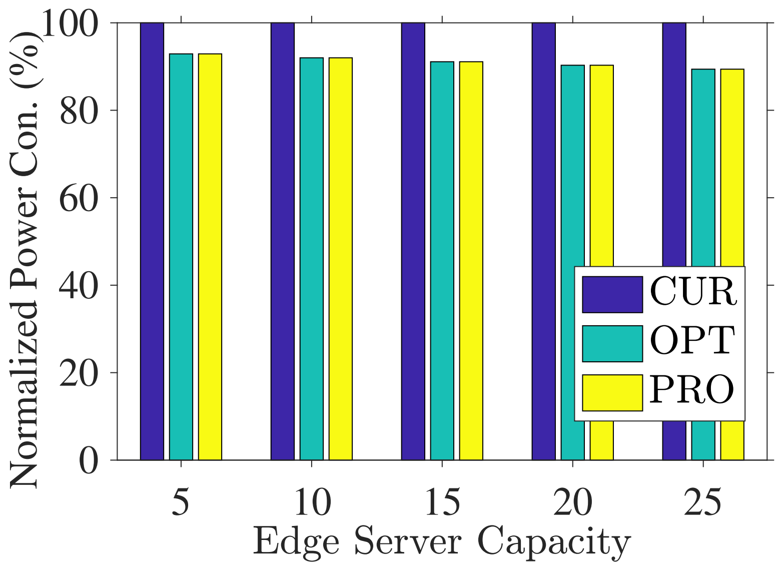


Figure 6.12: Normalized power consumption with diverse edge server capacity.

Chapter 7

Related Work

In this chapter, we present various state-of-the-art researches in the literature, from acquisition, encoding, transmission, to display. We also discuss quality assessment at the end of this section.

7.1 360° Video Acquisition

Capturing, stitching, and pre-processing images and videos from heterogeneous cameras for 360° videos are the starting point for providing immersive experience, which requires unique optimization approaches. Schreer et al. [84] discuss *format agnostic* production, which jointly leverages videos from multiple camera sensors at different temporal and spatial resolutions for 360° videos that support diverse applications from mobile devices to wide-angle displays. Different from conventional production systems, format-agnostic production systems have no fixed frame size and support virtual cameras controlled by directors or viewers. The authors built such a system using 360° cameras and image processing algorithms to capture nature scenes, and render immersive viewing (360° video) and hearing (spatial audio) experience. The techniques and experiments presented in their paper are valuable to researchers working on ultra-high resolution 360° videos.

Hardware cameras have been built for 360° videos, which could be based on either *single cameras* or *camera arrays*. Single camera approaches are less expensive and are immune to the artifacts due to stitching. For example, Krishan and Nayer [58] present a fisheye camera for 360° images, which attaches a curved mirror to a fish-eye lens. They propose an algorithm to stitch the two images from fish-eye lens and mirror into a seamless 360° image. Cotune et al. [35] present a stereoscopic 360° video capturing system using a pair of rotating commercial-graded video cameras. The authors employ full video frames for stereo motion alignment in the temporal domain. They do not stitch adjacent video frames, instead they blend the video frames, because blending seams are continuous, and thus, harder to spot. Aggarwal et al. [21] propose to use a filter mirror for stereoscopic 360° videos, where the light rays, into left and right eyes, are captured by a single camera. The filter mirror's surface is parameterized and the parameters are mathematically optimized for maximizing the quality of captured images. Compared to the: (i) multi-camera and (ii) moving camera systems, the filter mirror approach reduces the device size, simplifies the synchronization/calibration, and works on commodity cameras.

In contrast, single cameras need to support ultra-high resolution for immersive experience. Belbachir et al. [24] attach a pair of linear light sensors to a rotation platform, and generate stereoscopic 360° videos in real-time. This is achieved in three steps. First, two linear dynamic vision sensors, which are designed for capturing asynchronous images, are mounted on a high-speed rotating disk. Second, an algorithm is proposed to reconstruct intensity images using the sensor data. Last, stereoscopic 360° videos, anaglyph images, and depth images are created from the high dynamic range cameras at high frame rate.

Camera arrays often have higher total resolutions. For example, Foote and Kimber [43] build a camera array by attaching off-the-shelf cameras along the wall of a cylinder. This automatic camera system, called FlyCam, generates seamless videos by fusing data from several near-by cameras. They present an approach for stitching and anti-distortion, in order to generate 360° videos, motion analysis, and camera control algorithms. Their lightweight methods for 360° videos could be used in several applications, such as teleconferences, lectures, and meetings. Afshari et al. [19] construct a camera array by adding cameras on a sphere, which is inspired by flying insects. With FPGA, they design a system for real-time videos captured from up to 30 cameras. The cameras are put on a spherical pointing at different directions. In addition, they present a 360° video reconstruction algorithm, configurations, and a real FPGA implementation. Cogal et al. [31] present a similar camera array with 44 high-resolution cameras, achieving a total resolution of 220 Mega-Pixels (MP). The cameras are optimally arranged on a sphere for 360° $\times$ 100° FoV. The detailed hardware design of a 360° video capturing and recording system is presented in their paper. It is reported to achieve 21.6 MP at 30 frame-per-second (fps) and 82.3 MP at 9.5 fps in real-time.

Images and videos from cameras need to be pre-processed before being useful. Stitching, which merges multiple images/videos into a higher-resolution one, is probably the most commonly seen pre-process. Countless papers propose ways to improve the quality of image stitching, e.g., Zomet et al. [117], who developed optimization algorithms to maximize the stitching quality. The stitching quality is defined as a function of: (i) similarity between the output and input images and (ii) invisibility of the artifacts along the seams of stitched images. Several cost functions are introduced and tested, while the seam visibility is quantified in the gradient domain. Their proposed solution minimizes the adopted cost function, and can be used to generate 360° images and object blending, among other applications. Xiong and Pulli [109] also solve the stitching problem for images. They concentrate on minimizing the artifacts that appear on the seam of two stitched images, due to the color and luminance difference between them. This is achieved by color matching across input images with techniques like color correction and image blending. The resulting 360° images show high color consistency and smooth color transition. The proposed solution is implemented and evaluated on smartphones for 360° images with visually-appealing results. Xiao et al. [107], in contrast, propose an algorithm to generate panorama images directly from fisheye images. In particular, they formulate the projection conversion equations and map the points on fisheye images to the panorama using backward mapping approach.

Video stitching is more challenging, and receives increasingly more attentions. For example, Lin et al. [61] present a framework to stabilize and stitch videos captured by freely moving cameras. Each stitched video is generated by first identifying the camera paths, and constructing the 3D scene. Next, a new camera path is built by smoothening all the input camera paths. This new camera path is then employed to warp the input videos into a stitched one. Their proposed warping process is optimized for both stability and stitching quality. Their framework has various applications, such as stitching 360° videos, social-media content creation, and multi-robot vision. Jiang and Gu [53] design a spatial-temporal content preserving stitching approach for videos. Their proposed algorithm adopts warping to stitch imaging and stabilize videos, but with fixed camera positions. The algorithm consists of two steps: (i) aligning frames from multiple videos and (ii) finding spatial-temporal seams. The first step aligns frames from different videos in a temporally consistent manner. The second step is transformed into a 3D graphcut problem, where the weights are functions of objects and motion to maximize the stitching quality. Perazzi et al. [77] capitalize local warping to remove parallax from multiple unstructured cameras for 360° videos. Their proposed algorithm is unique for three reasons. First, they propose a patch-based error measure as a function of image gradients, which is used to maintain content similarity between input videos and the resulting 360° videos. Second, they design a method to analyze the relative camera positions, scene content, and order of pairwise warping, to improve the warping quality. Last, they introduce a weighted warping procedure for the final 360° videos, which mitigates the temporal artifacts. Because stitching videos is computationally intensive, they proposed to employ GPU to accelerate video stitching. Calagari et al. [28] build a similar system, but with videos from regular cameras that have been installed in the sports fields/courts for sports 360° videos. They first generate a static 360° image, which serves as the background image when some areas are not covered by any camera. Then, the authors derive player motions from the main (center) camera, and apply various techniques such as warping, to remove the parallax and align videos from all cameras. Last, the resulting video is blended with the background image. The resulting system has been tested in basketball, hockey, and volleyball games and compared against a GoPro Omni 360 camera rig. Silva et al. [88] connect multiple (4 or 6) GoPros to a computer via HDMI cables and capture cards. The captured video frames are loaded to GPU card memory. The last 4 (or 6) video frames in the queue are then stitched with vertex and fragment shaders. The resulting video frames are encoded by GPU and streamed by CPU. Lee et al. [59] propose approaches to solve issues of creating 360° videos from structured camera array. These issues include the misalignment between two adjacent cameras and the relative low resolution of the final 360° video. First, they leverage a moving checkerboard to perform calibration for estimating various settings of individual cameras. The depth disparities are then computed through feature extraction to recover 3D points for minimizing the parallax artifacts. Second, they propose to sample more important regions at higher frequencies, and less important regions at lower frequencies. Their results show that their proposed approach achieves higher rendering quality and preserves more content details than the equirectangular projection. Huang et al. [51] take a step further: they reconstruct 3D scenes in order to support 6 DoF in HMDs, where viewers can not only rotate their heads but also move freely in 3D scenes. Their approach contains two phases: offline and online. In the offline phase, they analyze the inputs of a 360° camera with the Structure-from-Motion (SfM) algorithm for camera parameters and scene geometry. When a viewer interacts with 360° videos using an HMD, i.e., in the online phase, the 6 DoF motions reported by the HMD sensors are used to warp the stereoscopic views corresponding to the motions. The warping algorithm runs on the unit sphere to avoid shape distortion. Their solution is accelerated by GPUs for high frame rate (> 120 fps).

7.2 360° Video Encoding

Typical 360° videos are spherical videos projected to rectangle videos in high resolutions, while viewers access random viewports, which are subsets of individual video frames. Encoding 360° videos, therefore, is very challenging and requires advances supports from video codecs. Heymann et al. [46] extend existing MPEG-4 to divide the 360° video into independently-encoded subvideos for the support of decoding and rendering parts of videos. Rerabek et al. [80] propose to encode 360° images into 360° JPEG files that can be decoded using the legacy JPEG decoder for backward compatibility. More specifically, their encoder first estimates the viewer viewports using saliency maps and encodes the viewports of 360° images using the regular JPEG encoder. The encoder then compress the entire 360° image also using JPEG and embeds the resulting bitstream as the metadata. By doing so, 360°-capable decoders and projectors can render the 360° images while legacy JPEG decoders render the viewport images. High Efficiency Video Coding (HEVC) [71] supports tiles, which are disjoint rectangular video regions that can be independently decoded. Tiles allow: (i) parallel decoding for 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. More details on the HEVC standard are given in Sullivan et al. [95], while the details about the tile supports in HEVC can be found in Misra et al. [66]. In general, HEVC results in 50% rate cut at similar quality [95] than AVC. In addition, tiles lead to up to 5.5% luminance bitrate reduction, compared to slices [66]. Due to their superior coding efficiency, HEVC and its tile 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 [102] 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 is reported that Kvazaar achieves real-time 4K encoding using a 14-core Intel CPU. Kvazaar supports three parallel processing approaches, including tiled encoding, and thus can be leveraged by 360° video testbeds.

Several types of optimization on video codecs have also been studied, which can be classified into three groups: (i) parameter selection, (ii) stream rewriting, and (iii) coding efficiency optimization. In parameter selection, Sanchez et al. [81] consider the problem of optimizing the tile dimension, in order to minimize the bitrate of the viewports. They propose a model using spatio-temporal activity metrics to achieve optimal tiling of the 360° videos for streaming. Their evaluations show that the proposed method resulted to higher coding efficiency, compared to static tile size, and lower complexity. Khiem et al. [56] adapt the encoding parameters of different regions in zoomable videos, based on the historical viewer access patterns, and propose two ways to dynamically crop viewports. The first way is to merge the tiles that fall into a single viewport, which is suitable for tiled encoding. The second way is to limit the motion search range of the referenced macroblocks, which is suitable for conventional encoding. Although the proposed solution improves compression efficiency, it does not achieve real-time streaming nor support diverse viewer access patterns.

In stream rewriting, Sanchez et al. [96] propose a compressed domain algorithm to rewrite multiple HEVC tiles into a single bitstream of the current viewport, which can be decoded by a single hardware decoder, and a solution to reduce the bandwidth consumption when users switch their viewports. This is a critical challenge for 360° video systems as high bandwidth consumption may lead to playout interruptions. Their core idea is to insert redundant reference pictures to compensate the temporal tiles that may not be streamed based on the viewers' viewports. Skupin et al. [92] propose the technique of dynamic tiling, which aims to adapt the resolution on-the-fly according to the viewer's viewports. They encode each video into high and low resolutions. Then, they vary the ratio of high- and low-resolution tiles over time, so that view's viewports are sent in high-resolution while other portions are sent in low-resolution tiles. Sanchez et al. [82] apply a similar approach on SHVC, which is the scalable extension of HEVC, in order to support multiple resolutions. The rewritten bitstream can be decoded by a single hardware decoder. Their proposed solution reduces the bitrate when switching among viewers' viewports. They utilize the concept of open GoP (Group-of-Picture) for better compression efficiency, because closed GoP may suffer from frame loss during decoding. Their solution supports seamless playback with and without the enhancement layers.

Several studies aim to optimize the coding efficiency for 360° videos and images. Sauer et al. [83] consider the convex polytopes projection, and propose a solution to compensate the geometric distortion for better motion compensation performance. Their work is motivated by the observation that straight lines are bended at the border of two adjacent faces when 360° videos are projected to a polytope. Such shape distortion results in suboptimal motion compensation when some motion happens across the borders. To cope with this issue, the authors propose to extend each face by projecting the adjacent faces to it using homographies. This is to ensure straight lines remain straight after projection, which increases the motion compensation performance and reduces the bitrate. Li et al. [60] extend each face of the cube projection to maintain texture continuity across face boundaries, so as to enhance motion compensation. They propose a padding method that projects the reference and current pixels on the same surface, before HEVC encoders perform motion estimation. Compared with the existing methods, the proposed solution offers an average (maximum) of 1.1% (3.4%) bitrate saving. Compression algorithms of non-rectangular images and videos have also been investigated. Tosic and Frossard [98] propose a 360° image compression algorithm that takes the geometric proprieties into considerations. The crux of their work is a redundant storage of basic geometric shapes on a sphere. This 360° image is projected on a sphere, and then passed into an iterative algorithm for a suboptimal solution of the weighted sum of a series of atoms. This is followed by an adaptive quantizer before being sent to the decoder. The resulting codec achieves superior performance at lower bitrates, where image geometry dominates image texture in terms of entropy. Youvaluri et al. [111] consider compressing 360° videos that are pseudo-cylindrically projected. However, using pseudo-cylindrical projection leads to some problems, such as coding inefficiency and coding artifacts at its borders. They propose intra- and inter-frame coding tools for higher coding efficiency and mitigate coding artifacts. Ozcinar et al. [74] select the tile bitrates in 360° videos. They formulate it as an Integer Linear Programming (ILP) problem, which chooses a subset of bitrates for each tile, in order to maximize a weighted sum of video quality and resource (storage and network) cost. The weights are heuristically chosen by the service providers, and the resulting problem is solved using a general ILP solver. Yu et al. [112] also optimize the bitrate of the tiles, while jointly considering the sampling densities of different part of the 360° videos. The rationale is that some projection models, like equirectangular model, oversample the sphere videos close to north and south poles. Because the resulting problem is a variant of knapsack problems, the authors propose a suboptimal algorithm to first determine sampling density, followed by deciding the bitrates. Xie et al. [108], in contrast, consider a more general bit-rate allocation problem, where tiles have diverse viewing probabilities. They formulate the problem as a mathematical optimization problem, with a weight sum of: (i) overall viewport distortion and (ii) inter-tile distortion variance. The distortion and bitrates of 360° video titles coded with diverse codec settings, such as the Quantization Parameters (QPs), are empirically derived, and their optimization problem is an ILP problem, which are solved with general solvers.

7.3 360° Video Transmission

Modern multimedia transport standards can be roughly classified into MPEG Media Transport (MMT), for broadcast services, and DASH, for unicast services.

Broadcasting 360° videos over the Internet has been studied. For example, Hu et al. [49] propose a 360° video broadcast system using MPEG Media Transport (MMT) for broadcasting over the Internet. The authors divide and encode the 360° videos into multiple tiles. The encoded tiles are encapsulated into multiple MMT assets, which can be individually received by receivers. The authors employ MMT signaling messages to describe the spatial relationship of MMT assets. This allows receivers to subscribe the tiles in their viewports at high bitrate, and other tiles at low bitrate.

To support 360° video streaming, MPEG DASH standard has included an amendment on Spatial Representation Description (SRD), which enables clients to request viewports of whole videos with 2D coordinates. The SRD standard is presented in Niamut et al. [70], along with several use cases of tiled videos. SRD expands Media Presentation Description (MPD) to define the relative spatial positions of tiles. It provides attributes like x- and y-axis coordinates, as well as width and height, DASH clients can determine what tiles to request. Like MPD, the SRD only provides the spatial organization of content, without dictating how DASH clients leverage such information. Several use cases of SRD has been proposed and discussed, such as zoomable, mobile, and TV-wall displays, where tiled streaming provides additional flexibility. Concolato et al. [32] further discuss the High-Efficiency Video Codec (HEVC) and ISO Base Media File Format (ISOBMFF) standards, for encoding and encapsulating tiled videos for transmission. Combining SRD, HEVC, and ISOBMFF, a client may merge several tiles into a video stream, which is decodable by a decoder. Their evaluations show that the proposed approach incurs a minor streaming overhead when delivering the tiled videos compared to the non-tiled ones. Therefore, standard DASH clients can then request and decode some or all tiles; in other words, a subset of tiles in different quality levels can be selected based on the available bandwidth in dynamic networks.

Several papers [36, 42, 45] share their experience of realizing standard-based 360° video transmission. In particular D'Acunto et al. [36] provide guidelines on realizing navigable video transmission using SRD. They summarize the design choices that allow players to render SRD-enabled DASH content: including (i) selecting the best resolution layers for the current viewport and (ii) enabling a seamless switches among tiled videos. In addition, they give examples on how a player may use SRD to support zoomable and navigable videos by extending dash.js [72], which is an MPEG DASH reference client. Feuvre and Concolato [42] employ several open source projects, such as Kvazaar [102], MP4Box [13], and MP4Client [76], to realize tile-based adaptive transmission using MPEG-DASH and SRD. Furthermore, they discuss on different adaptation policies of tiled 360° videos, where the tiles are either compressed independently or with tile-constrained motion vectors. Graf et al. [45] present a tile-based 360° streaming system and implement tools to evaluate the pros and cons of using different encoding and streaming strategies. They explore various options enabling the bandwidth-efficient 360° video adaptation over HTTP. They find that 6x4 tiles provide the best tradeoff between tiling overhead and bandwidth consumption. The trace-driven evaluations has a confirm bitrate saving up to 40% at similar video quality compared to the existing solutions.

Conducting measurement study and carrying out reverse engineering on the commercial 360° video services, such as YouTube and Facebook, is another research direction. For example, Afzal et al. [20] analyze the characteristics of online 360° videos. They collect thousands of 360° videos from YouTube and classify them into several genre. They further analyze the variability in videos resolution, and bitrate, and the possible underlying causes of these variabilities compared to transmitting non-360° videos. Zhou et al. [116] perform detailed measurements on Oculus 360° videos from Facebook and describe the offset cubic projection implemented by Oculus. Furthermore, they calculate the angles of users' viewports by their mathematical formula in order to give-high quality videos in the current viewports. Several experiments with different conditions are done to test how many segments are unwatched and wasted. The results show that offset cubic projection model saves more bandwidth than the traditional model, which sends all tiles of the same quality to clients. These studies [20, 116] shed some lights on how to optimize 360° video transmission, although they may not directly achieve that.

Most 360° video transmission is optimized through sending tiles at different quality levels. More precisely, the tiles in the viewer's current viewport are sent at higher quality, and others are sent at lower quality, to cut the bandwidth usage without viewing quality degradation. Thus, Zare et al. [114] propose to encode each 360° video into tiles in two representations: high- and low-resolutions. Then, they would transmit the viewport tiles at high resolution, and other tiles at low resolution. They adopt motion-constrained HEVC tiles and propose three heuristic schemes for 360° video transmission to HMDs. Even though no intelligent adaptation is done with their tiling schemes, experiment results reveal that their solution leverages the common patterns of head movements and achieves better coding efficiency. Nguyen et al. [69] also stream 360° videos in two regions: the center region, which is rectangular and covers the most-probable viewports, and the residue region. They solve an optimization problem by deciding the size and encoding bitrate of the center region, so as to maximize video quality under the bandwidth constraint. Their solution assumes the mapping between video quality and bitrate is empirically derived, and that only one receiver is considered in the optimization problem. Ju et al. [54] encode each 360° video into two representations and transmit the low-resolution full 360° videos along with high-resolution viewports. They propose to consider the heat map of viewer's attentions, for live transmitting the high-viewing probability portion. Corbillon et al. [33] encode 360° videos into two representations and take diverse projection models into considerations. In addition to varying the bitrates in different representations, they also consider the viewports of 360° videos in HMDs. That is, each 360° video is divided into segments, where each segment is compressed multiple times with combinations of viewports and bitrates. Each user then requests for the proper representation via the DASH protocol. The same authors extend their work to include some theoretical models for the viewport-adaptive 360° video streaming [33]. These models are then simplified by the following assumptions: (i) uniform coding complexity, (ii) two representations, (iii) maximum quality gap between the representations, and (iv) rectangular viewports. They then propose a viewport-adaptive streaming algorithm to exercise the trade-off between the viewport size and the tile bitrates.

Duanmu et al. [37] generalize the previously mentioned two-representation approach, and encode each 360° video into a base and multiple enhancement representations. Then, they create different buffers for these representations, and present their prioritized buffer control mechanisms. They give the highest priority to the playback continuity by first guaranteeing the transmission of the base representation. The residual bandwidth is then used to download enhancement representations. Nasrabadi et al. [67] consider a very similar problem, but explicitly use Scalable Video Coding (SVC) tiles for 360° videos. The core idea is to prefetch and buffer the base-layer tiles of the whole 360° videos earlier, in order to avoid playout interruption and a long rebuffering time. For tiles in viewports, enhancement-layer tiles are transmitted with residue bandwidth. Adaptations based on viewer orientations and network conditions are both considered in their work. Petrangeli et al. [79] propose to capitalize the push-based HTTP/2 protocol (instead of HTTP/1.1) for multiple representation transmission to reduce the network overhead, and also an algorithm to predict the tiles that may be watched in the future. Then, they use HTTP/2 to save time on request and delivery of the predicted tiles. The same authors extend the work into a complete system [78], e.g., dead-reckoning is adopted to predict future viewports, and thorough evaluation results are reported. Ozcinar et al. [75] propose to send the viewport at the highest-possible bitrate, and gradually reduce the bitrates that are proportional to the distance to the viewport. They build an end-to-end transmission system for 8K resolution 360° videos watched by HMDs. The experiments show that their proposed system provides better viewing quality than the baselines, when PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural SIMilarity) of viewports are considered. Hosseini and Swaminathan [47] also consider multiple representations, but they propose a new projection model for better viewing quality. They leverage peripheral vision, and further reduce the quality of some tiles. With viewport tracking, the dynamically delivered tiles within user's viewports are at a higher bitrate. 72% of bandwidth saving is reported without clear quality drops. These studies [114, 54, 33, 37, 67, 79, 75, 47, 69, 33] strive to save bandwidth by reducing the quality of unwatched or less-noticed tiles, which is agnostic to the network and computation infrastructures.

7.4 360° Video Quality Assessment

Video QoE refers to the human perceived video quality, which can only be quantified using rigorously designed testbeds and procedures. The QoE metrics are either: (i) *subjective* or (ii) *objective metrics*. The subjective metrics are from user inputs, most likely through some questionnaires, while the objective metrics are from computer algorithms. The subjective metrics are close with actual human perception, but require more efforts to design, conduct, and analyze. In contrast, the objective metrics are easier to derive, but may deviate from the real human perception. QoE evaluations and optimization have been crucial for multimedia applications, even before 360° videos are popularized. For example, Tan et al. [97] present the standardized evaluation procedures for quantifying the subjective and objective QoE levels achieved by the latest H.265/HEVC video coding standard. The authors adopt PSNR as the objective quality metric. In terms of subjective quality assessment, they follow the procedure suggested by ITU-Rec. P910 [86] and ITU-Rec. BT500 [85], which are for multimedia applications and television pictures, respectively. The subjects were asked to view and rate a random series of basic test cells, where each cell contains two video clips: the original video clip followed by the reconstructed one. Their evaluation procedure is not suitable for 360° videos, which are often transmitted in tiles for selectively transmitting tiles that are more likely to be watched to save the bandwidth consumption. Wang et al. [105] consider the QoE of zoomable video tiles, which may have diverse resolutions. Their evaluations reveal that users may not notice some tiles that are transmitted at lower resolutions. In particular, they conduct user studies to measure two thresholds for just noticeable/unacceptable differences. These two thresholds are then used to drive the resolution selection among tiles under the restricted bandwidth. Their experiment results demonstrate a 14%–20% bandwidth saving using their proposed method. In addition, the results also reveal that the two thresholds are related to the characteristics of video content, such as the motion levels. However, their work focuses on conventional flat displays, rather than modern HMDs.

QoE evaluation testbeds and procedures have gradually received more attentions. A testbed is built [100] for evaluations of QoE metrics of 360° videos. They demonstrate the applicability of their testbed by using it to collect Mean Opinion Score (MOS) of 360° images and videos encoded at different quality levels. Their testbed allows subjects to view images and videos using HMDs based on mobile devices, e.g., Google Cardboard. During each assessment session, their proposed testbed tracks the subject's scores, orientation, and consumed time. Singla et al. [91] assess sickness caused by watching 360° videos in HMDs via subjective evaluations. They consider two commercial HMDs and two resolutions in their experiments. That is, each video content is viewed 4 times with different combinations of HMDs and resolutions. 28 subjects are recruited to rate the 360° videos downloaded from YouTube, where 6 videos in total are used. Their results show that both the resolution and content have significant impacts on the subjects' experience, while the HMDs only impose a slight influence on it. Example observations include that HTC Vive provides a little better overall quality compared to Oculus Rift. In addition, average female users suffer more on sickness, especially for disorientation. Singla et al. [90] quantify the implication of different bitrates and resolutions through subjective evaluations. The authors propose a modified subjective evaluation procedure, which: (i) allows the subjects to view the test sequence twice for more reliable rating and (ii) asks the subjects to rate the test sequence through speech to prevent the interruption of wearing or taking off the HMD. Bessa et al. [25] study whether 3D (stereoscopic) view will improve the subjective QoE levels, compared to 2D view. They recruit 63 participants to view a single video. Half of the participants watch the video in 2D version while the other half watch the video in 3D version. Surprisingly, their results show that 3D 360° videos brings no benefit to the viewers compared to 2D 360° videos. This may cause by the subjects' have limited 360 video viewing experience, especially in 3D. QoE evaluations on specific applications are also possible, e.g., Hupont et al. [52] propose procedures to study the gaming QoE with HMDs. The authors conduct experiments to evaluate: (i) the perceived presence scores and (ii) the usability scores on both conventional 2D displays and HMDs. Their results show that wearing HMDs provides better experience compared to 2D displays in various aspects, such as realism, possibility to act, and willingness to use, while also leading to higher complexities and steeper learning curves. While the studies [100, 91, 90, 25, 52] shed some lights on fine-tuning the QoE of 360° videos, they do not actively optimize the QoE of 360° video systems.

Some work moves a bit further and optimizes 360° video systems in terms of user QoE levels. Hsu et al. [48] carry out QoE evaluations on foveated rendering systems, where the objects in the foveal region are encoded at higher quality than the objects in the peripheral region. In their study, they vary the resolution of the foveal region and peripheral region on 2D displays and consider four types of subjective quality assessment methods. They find that most of the viewers do not perceive the distortion if the size of the foveal region is larger than 7.5°. Furthermore, they evaluate the consistency and efficiency of the considered assessment methods. Based on the results from the considered four methods, they model the perceptual ratio on foveated rendering using regression analysis. Steed et al. [93] study how user interface and conditions may affect the VR QoE of HMD users. The considered scenario is a virtual singer singing on the stage. The authors consider 8 conditions in this scenario using Unity in their user studies. These conditions come from the combinations of 3 settings: (i) with or without self-avatar, (ii) with or without the singer asking the user to tap along to the beats, and (iii) with or without the singer looking at the user. Their results reveal that self-avatar makes clear impact on the user experience. The user experience is degraded due to synchronization issues between the subject and his/her self-avatar, when tapping along the beats. However, with the singer looking at the user, no negative effect on viewing experience is observed. The authors argue that this is because the user has no expectation on the singer to engage with him or her. Fernandes and Feiner [41] propose an evaluation design to understand the relations between viewport size, sickness, and perceived quality. In particular, they vary the viewport degree between 80° and 90°. Each subject is asked to wear an HMD and walk along a set of waypoints in virtual environments. They rate the discomfort levels every 5 waypoints. Their experiment results indicate that restricting viewport size helps subjects adapt to the virtual environments and reduce the discomfort level, as long as the restricted viewport size is acceptable to the subjects. These papers [48, 93, 41] concentrate on subjective evaluations.

Recruiting viewers for subjective evaluations is costly, error-prone, and tedious. Therefore, several papers [101, 99, 38] discuss how to estimate the subjective results using the objective results, so as to reduce the overhead of subjective evaluations. Upenik et al. [101] conduct subjective evaluations with 45 subjects, and try to analyze the correlation between the subjective MOS values and objective quality levels. Their considered objective quality metrics include PSNR, SSIM, M-SSIM, VIFP, S-PSNR, WS-PSNR [112], and CPP-PSNR [113]. Their analysis on the subjective scores and objective metrics show that the existing objective quality metrics designed for 360° videos (e.g., S-PSNR and WS-PSNR) do not have higher correlation to the subjective scores than the original metrics (e.g., PSNR). Therefore, the authors conclude that there are still open issues in this research direction, e.g., a better objective quality metric specifically designed for 360° videos is needed. Tran et al. [99] conduct similar evaluations on 18 subjects. Their findings are more promising compared to Upenik et al. [101], e.g., all their considered objective metrics have high correlation to the subjective results. The sources of distortion are due to: (i) changing video content format and (ii) transmitting content over networks. Last, Egan et al. [38] predict the QoE scores based on the biosensors. It is shown that the electrodermal activity has significant contribution to the QoE scores. On the contrary, the heart rate has no effect for the subjective scores. Their results confirm that exploring VR using HMDs leads to more immersive experience than using 2D displays. While the community is making progress at the front of modeling subjective evaluation results with a function of objective measurement, there are still many open issues, as pointed out by Upenik et al. [101].

Chapter 8

Conclusion and Future Work

In this thesis, we study the streaming pipeline of 360° video to HMD viewers. We found to fulfill a real immersive AR/VR experience, there would be several challenges, including high bandwidth consumption, latency-sensitive, and heterogeneous HMD devices. To conquer the challenges, we proposed an edge-assisted rendering system, which leverage edge devices to perform the tile rewriting and viewport rendering. Then, we formulate the optimization problem to determine which HMD should be served without overloading the edge devices, and design an algorithm to solve the aforementioned problem properly, and a real testbed is implemented to prove the concept. The resulting edge-assisted 360° video streaming system is evaluated through extensive experiments with an open-sourced 360° viewing dataset. Compared to current 360° video streaming platforms, like YouTube, our edge-assisted rendering platform can: (i) save up to 62% in bandwidth consumption, (ii) achieve higher viewing video quality up to 7.7 dB at a given bitrate, and (iii) reduce the computation workload for those lightweight HMDs.

This work, however, can be extended in several directions:

- **Leverage GPUs to fulfill real-time computing.** Rendering user's viewport needs lots of computing power, so to fulfill real-time computing, we should leverage GPU toolkit and parallel computing to accelerate the viewport rendering. These enhancements will further improve the performance of our edge-assisted 360° streaming platform.
- **Modeling the computing cost of VPR running on an edge server.** Each edge server however has limited computing power. Like we mentioned before, rendering the user's viewport needs lots of computing power. To know the exactly number of HMD clients that edge server can serve, we should model the computing cost of VPR and develop a computing model when serving heterogeneous HMD clients.
- **More intelligence algorithm.** A better algorithm could be developed to carefully choose the best ways to assist individual HMDs for maximizing the overall video

quality improvement without overloading: (i) networks, (ii) edge servers, and (iii) HMDs.

By differentiating the pros and cons of edge-assisted streaming to current approach, we open up new opportunities for researchers and engineers to further optimize 360° streaming platforms in terms of user experience.

Bibliography

[1] The OpenCV Library. http://opencv.org, 2000. Accessed May 2018.

[2] After mixed year, mobile AR to drive $108 billion VR/AR market by 2021. https://goo.gl/P9N0z0, 2017. Accessed May 2018.

[3] Facebook Spaces. https://www.facebook.com/spaces, 2017. Accessed April 2018.

[4] Oculus Video. ://www.oculus.com/experiences/rift/926562347437041/, 2017. Accessed May 2018.

[5] EC2 Network Benchmark Results. https://cloudonaut.io/behind-the-scences-ec2-network-performance-benchmark/, 2018. Accessed July 2018.

[6] Facebook. https://www.facebook.com/, 2018. Accessed May 2018.

[7] Facebook Oculus Rift. https://www.oculus.com, 2018. Accessed May 2018.

[8] Google Cardboard. https://vr.google.com/cardboard/, 2018. Accessed May 2018.

[9] HTC Vive. https://www.htcvive.com, 2018. Accessed May 2018.

[10] HTC Vive Focus. https://www.vive.com/cn/product/vive-focus-en/, 2018. Accessed May 2018.

[11] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, 2018. Accessed July 2018.

[12] Luna 360 VR. http://luna.camera/, 2018. Accessed May 2018.

[13] MP4Box. https://gpac.wp.imt.fr/mp4box/, 2018. Accessed May 2018.

[14] Richo Theta S. https://theta360.com, 2018. Accessed May 2018.

[15] Samsung Gear 360. http://www.samsung.com/global/galaxy/gear-360/, 2018. Accessed May 2018.

[16] Samsung Gear VR. http://www.samsung.com/global/galaxy/gear-vr, 2018. Accessed May 2018.

[17] Sony Playstation VR. https://www.playstation.com/en-au/explore/playstation-vr/, 2018. Accessed May 2018.

[18] YouTube. https://www.youtube.com/, 2018. Accessed May 2018.

[19] H. Afshari, V. Popovic, T. Tasci, A. Schmid, and Y. Leblebici. A spherical multi-camera system with real-time omnidirectional video acquisition capability. *IEEE Transactions on Consumer Electronics*, 58(4):1110–1118, 2012.

[20] S. Afzal, J. Chen, and K. Ramakrishnan. Characterization of 360-degree videos. In *Proceedings of the 2017 ACM SIGCOMM Workshop on Virtual Reality and Augmented Reality Network*, VR/AR Network'17, pages 1–6, 2017.

[21] R. Aggarwal, A. Vohra, and A. Namboodiri. Panoramic stereo videos with a single camera. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'16, pages 3755–3763, 2016.

[22] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon. Transcodclouding live adaptive video streams at a massive scale in the cloud. In *Proceedings of the 2015 ACM Multimedia Systems Conference*, MMSys '15, pages 49–60, 2015.

[23] I. Bauermann, M. Mielke, and E. Steinbach. H.264 based coding. In *Proceedings of the 2004 International Conference Computer Vision and Graphics*, ICCVG'04, pages 209–215, 2004.

[24] A. Belbachir, S. Schraml, M. Mayerhofer, and M. Hofstatter. A novel HDR depth camera for real-time 3D 360° panoramic vision. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'14, pages 425–432, 2014.

[25] 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 *Proceedings of the 2016 International Conference on Human Computer Interaction*, HCI'16, pages 16:1–16:4, 2016.

[26] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the Internet of Things. In *Proceedings of the 2012 MCC Workshop on Mobile Cloud Computing*, MCC'12, pages 13–16, 2012.

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

[28] K. Calagari, M. Elgharib, S. Shirmohammadi, and M. Hefeeda. Sports VR content generation from regular camera feeds. In *Proceedings of the 2017 ACM Multimedia Conference*, MM '17, pages 699–707, 2017.

[29] J. Carmack. Latency Mitigation Strategies. `https://www.twentymilliseconds.com/post/latency-mitigation-strategies/`, 2018. Accessed April 2018.

[30] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2018. Accessed May 2018.

[31] O. Cogal, A. Akin, K. Seyid, V. Popovic, A. Schmid, B. Ott, P. Wellig, and Y. Leblebici. A new omni-directional multi-camera system for high resolution surveillance. *Proceedings of SPIE, Mobile Multimedia/Image Processing, Security, and Applications*, 9120(9):9120–9120, 2014.

[32] C. Concolato, J. 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, 2017.

[33] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *Proceedings of the 2017 IEEE International Conference on Communications*, ICC'17, pages 1–7, 2017.

[34] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara. A deep multi-level network for saliency prediction. In *Proceedings of the 2016 International Conference on Pattern Recognition*, ICPR'16, pages 3488–3493, 2016.

[35] V. Couture, M. Langer, and S. Roy. Panoramic stereo video textures. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV'11, pages 1251–1258, 2011.

[36] L. D'Acunto, J. Berg, E. Thomas, and O. Niamut. Using MPEG DASH SRD for zoomable and navigable video. In *Proceedings of the 2016 ACM Multimedia Systems Conference*, MMSys '16, page 34:1–34:4, 2016.

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

[38] 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 *Proceedings of the 2016 International Conference on Quality of Multimedia Experience*, QoMEX'16, pages 1–6, 2016.

[39] 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 *Proceedings of the 2017 Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV'17, pages 67–72, 2017.

[40] C. Feldmann, C. Bulla, and B. Cellarius. Efficient stream-reassembling for video conferencing applications using tiles in HEVC. In *Proceedings of the 2013 International Conferences on Advances in Multimedia*, MMEDIA'13, pages 130–135, 2013.

[41] A. Fernandes and S. Feiner. Combating VR sickness through subtle dynamic Field-of-View modification. In *Proceedings of the 2016 IEEE Symposium on 3D User Interfaces*, 3DUI'16, pages 201–210, 2016.

[42] J. Feuvre and C. Concolato. Tiled-based adaptive streaming using MPEG-DASH. In *Proceedings of the 2016 ACM Multimedia Systems Conference*, MMSys '16, page 41, 2016.

[43] J. Foote and D. Kimber. FlyCam: practical panoramic video and automatic camera control. In *Proceedings of the 2000 IEEE International Conference on Multimedia and Expo.*, ICME'00, pages 1419–1422, 2000.

[44] 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, 2009.

[45] M. Graf, C. Timmerer, and C. Mueller. Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP. In *Proceedings of the 2017 ACM Multimedia Systems Conference*, MMSys'17, pages 261–271, 2017.

[46] S. Heymann, A. Smolic, K. Muller, Y. Guo, J. Rurainsky, P. Eisert, and T. Wiegand. Representation, coding and interactive rendering of high-resolution panoramic images and video using MPEG-4. In *Proceedings of the 2005 Panoramic Photogrammetry Workshop*, PPW'05, 2005.

[47] M. Hosseini and V. Swaminathan. Adaptive 360 VR video streaming: Divide and conquer. In *Proceedings of the 2016 IEEE International Symposium on Multimedia*, ISM'16, pages 107–110, 2016.

[48] 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 *Proceedings of the 2017 ACM Multimedia Conference*, MM'17, pages 55–63, 2017.

[49] Y. Hu, S. Xie, Y. Xu, and J. Sun. Dynamic VR live streaming over MMT. In *Proceedings of the 2017 International Symposium on Broadband Multimedia Systems and Broadcasting*, BMSB'17, pages 1–4, 2017.

[50] C. Huang, C. Hsu, Y. Chang, and K. Chen. GamingAnywhere: An open cloud gaming system. In *Proceedings of the 2013 ACM Multimedia Systems Conference*, MMSys'13, pages 36–47, 2013.

[51] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-DOF VR videos with a single 360-camera. In *Proceedings of the 2017 IEEE Virtual Reality*, VR'17, pages 37–44, 2017.

[52] I. Hupont, J. Gracia, L. Sanagustin, and M. Gracia. How do new visual immersive systems influence gaming QoE: A use case of serious gaming with Oculus Rift. In *Proceedings of the 2015 International Conference on Quality of Multimedia Experience*, QoMEX'15, pages 1–6, 2015.

[53] W. Jiang and J. Gu. Video stitching with spatial-temporal content-preserving warping. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'15, pages 42–48, 2015.

[54] R. Ju, J. He, F. Sun, J. Li, F. Li, J. Zhu, and L. Han. Ultra wide view based panoramic VR streaming. In *Proceedings of the 2017 ACM SIGCOMM Workshop on Virtual Reality and Augmented Reality Network*, VR/AR Network'17, pages 19–23, 2017.

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

[56] N. Khiem, G. Ravindra, and W. Ooi. Adaptive encoding of zoomable video streams based on user access pattern. *Signal Processing: Image Communication*, 27(4):360–377, 2012.

[57] H. Kimata, S. Shimizu, Y. Kunita, M. Isogai, and Y. Ohtani. Panorama video coding for user-driven interactive video application. In *Proceedings of the 2009 IEEE International Symposium on Consumer Electronics*, ISCE'09, pages 112–114, 2009.

[58] G. Krishnan and S. Nayar. Cata-fisheye camera for panoramic imaging. In *Proceedings of the 2008 IEEE Workshop on Applications of Computer Vision*, WACV'08, pages 1–8, 2008.

[59] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh. Rich360: Optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics*, 35(4):63:1–63:11, 2016.

[60] L. Li, Z. Li, X. Ma, H. Yang, and H. Li. Co-projection-plane based 3-D padding for polyhedron projection for 360-degree video. In *Proceedings of the 2017 IEEE International Conference on Multimedia and Expo.*, ICME'17, pages 55–60, 2017.

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

[62] W. Lo, C. Fan, J. Lee, C. Huang, K. Chen, and C. Hsu. 360° video viewing dataset in head-mounted virtual reality. In *Proceedings of the 2017 ACM Multimedia Systems Conference*, MMSys'17, pages 211–216, 2017.

[63] 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 *Proceedings of the 2017 Asia-Pacific Network Operations and Management Symposium*, APNOMS '17, pages 205–210, 2017.

[64] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 1981 International Joint Conference on Artificial Intelligence*, IJCAI'81, pages 674–679, 1981.

[65] K. Mania, B. Adelstein, S. Ellis, and M. Hill. Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *Proceedings of the 2004 Symposium on Applied Perception in Graphics and Visualization*, APGV'04, pages 39–47, 2004.

[66] 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, 2013.

[67] A. Nasrabadi, A. Mahzari, J. Beshay, and R. Prakash. Adaptive 360-degree video streaming using scalable video coding. In *Proceedings of the 2017 ACM Multimedia Conference*, MM'17, pages 1689–1697, 2017.

[68] K. Ng, S. Chan, and H. Shum. Data compression and transmission aspects of panoramic videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(1):82–95, 2005.

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

[70] O. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Lim. MPEG DASH SRD: spatial relationship description. In *Proceedings of the 2016 ACM Multimedia Systems Conference*, MMSys '16, page 5, 2016.

[71] 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, 2013.

[72] J. Oliva. A reference client implementation for the playback of MPEG DASH via javascript and compliant browsers. `https://github.com/Dash-Industry-Forum/dash.js/`, 2017. Accessed April 2018.

[73] OpenTrack: head tracking software. `https://github.com/opentrack/opentrack`, 2018. Accessed May 2018.

[74] C. Ozcinar, A. Abreu, S. Knorr, and A. Smolic. Estimation of optimal encoding ladders for tiled 360° VR video in adaptive streaming systems. In *Proceedings of the 2017 IEEE International Symposium on Multimedia*, ISM'17, pages 45–52, 2017.

[75] C. Ozcinar, A. Abreu, and A. Smolic. Viewport-aware adaptive 360° video streaming using tiles for virtual reality. In *Proceedings of the 2017 IEEE International Conference on Image Processing*, ICIP'17, pages 2174–2178, 2017.

[76] T. ParisTech. MP4Client. `https://gpac.wp.imt.fr/player/`, 2017. Accessed April 2018.

[77] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross. Panoramic video from unstructured camera arrays. *Computer Graphics Forum*, 34(2):57–68, 2015.

[78] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. Turck. An HTTP/2-Based adaptive streaming framework for 360° virtual reality videos. In *Proceedings of the 2017 ACM Multimedia Conference*, MM'17, pages 306–314, 2017.

[79] S. Petrangeli, F. Turck, V. Swaminathan, and M. Hosseini. Improving virtual reality streaming using HTTP/2. In *Proceedings of the 2017 ACM Multimedia Systems Conference*, MMSys'17, pages 225–228, 2017.

[80] M. Rerabek, E. Upenik, and T. Ebrahimi. JPEG backward compatible coding of omnidirectional images. *Applications of Digital Image Processing XXXIX*, 9971(10):1–12, 2016.

[81] Y. Sanchez, R. Skupin, C. Hellge, and T. Schierl. Spatio-temporal activity based tiling for panorama streaming. In *Proceedings of the 2017 Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV'17, pages 61–66, 2017.

[82] Y. Sanchez, R. Skupin, and T. Schierl. Compressed domain video processing for tile based panoramic streaming using SHVC. In *Proceedings of the 2015 International Workshop on Immersive Media Experiences*, ImmersiveME'15, pages 13–18, 2015.

[83] J. Sauer, J. Schneider, and M. Wien. Improved motion compensation for 360° video projected to polytopes. In *Proceedings of the 2017 IEEE International Conference on Multimedia and Expo*, ICME'17, pages 61–66, 2017.

[84] O. Schreer, I. Feldmann, C. Weissig, P. Kauff, and R. Schafer. Ultrahigh-resolution panoramic imaging for format-agnostic video production. *Proceedings of the IEEE*, 101(1):99–114, 2013.

[85] I. R. Sector. Methodology for the subjective assessment of the quality of television picture. *ITU-R Recommendation*, BT.500(13), January 2012.

[86] I. T. S. Sector. Subjective video quality assessment methods for multimedia applications. *ITU-T Recommendation*, P.910, April 2008.

[87] M. Shirer and S. Murray. IDC Sees the Dawn of the DX Economy and the Rise of the Digital-Native Enterprise. `https://www.businesswire.com/news/home/20161101005193/en/IDC-Sees-Dawn-DX-Economy-Rise-Digital-Native`, 2016. Accessed April 2018.

[88] R. Silva, B. Feijó, P. Gomes, T. Frensh, and D. Monteiro. Real time 360° video stitching and streaming. In *Proceedings of the 2016 ACM Special Interest Group on Computer GRAPHics and Interactive Techniques Conference*, SIGGRAPH'16, pages 70:1–70:2, 2016.

[89] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[90] A. Singla, S. Fremerey, W. Robitza, P. Lebreton, and A. Raake. Comparison of subjective quality evaluation for HEVC encoded omnidirectional videos at different bitrates for UHD and FHD resolution. In *Proceedings of the 2017 ACM Multimedia Thematic Workshops*, Thematic Workshops'17, pages 511–519, 2017.

[91] 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 *Proceedings of the 2017 International Conference on Quality of Multimedia Experience*, QoMEX'17, pages 1–6, 2017.

[92] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl. Tile based HEVC video for head mounted displays. In *Proceedings of the 2016 IEEE International Symposium on Multimedia*, ISM'16, pages 399–400, 2016.

[93] A. Steed, S. Frlston, M. Lopez, J. Drummond, Y. Pan, and D. Swapp. An 'in the wild' experiment on presence and embodiment using consumer virtual reality equipment. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1406–1414, 2016.

[94] T. Stockhammer. Dynamic adaptive streaming over HTTP: Standards and design principles. In *Proceedings of the 2011 ACM Multimedia Systems Conference*, MMSys'11, pages 133–144, 2011.

[95] 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, 2012.

[96] Y. Sánchez, R. Skupin, and T. Schierl. Compressed domain video processing for tile based panoramic streaming using HEVC. In *Proceedings of the 2015 IEEE International Conference on Image Processing*, ICIP'15, pages 2244–2248, 2015.

[97] T. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J. Ohm, and G. Sullivan. Video quality evaluation methodology and verification testing of HEVC compression performance. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):76–90, 2016.

[98] I. Tosic and P. Frossard. Low bit-rate compression of omnidirectional images. In *Proceedings of the 2009 Picture Coding Symposium*, PCS'09, pages 1–4, 2009.

[99] H. Tran, N. Ngoc, C. Bui, M. Pham, and T. Thang. An evaluation of quality metrics for 360 videos. In *Proceedings of the 2017 International Conference on Ubiquitous and Future Networks*, ICUFN'17, pages 7–11, 2017.

[100] E. Upenik, M. Rerabek, and T. Ebrahimi. Testbed for subjective evaluation of omnidirectional visual content. In *Proceedings of the 2016 Picture Coding Symposium*, PCS'16, pages 1–5, 2016.

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

[102] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen. Kvazaar: Open-source HEVC/H.265 encoder. In *Proceedings of the 2016 ACM Multimedia Conference*, MM'16, pages 1179–1182, 2016.

[103] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. In *Proceedings of the 2010 Conference on Virtual Reality Conference*, VR '10, pages 211–218, 2010.

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

[105] H. Wang, V. Nguyen, W. Ooi, and M. Chan. Mixing tile resolutions in tiled video: A perceptual quality assessment. In *Proceedings of the 2014 Workshop on Network*

and Operating Systems Support for Digital Audio and Video, NOSSDAV'14, pages 25:25–25:30, 2014.

[106] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[107] S. Xiao and F. Wang. Generation of panoramic view from 360 degree fisheye images based on angular fisheye projection. In *Proceedings of the 2011 International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, DCABES'11, pages 187–191, 2011.

[108] 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 *Proceedings of the 2017 ACM Multimedia Conference*, MM'17, pages 315–323, 2017.

[109] Y. Xiong and K. Pulli. Color matching for high-quality panoramic images on mobile phones. *IEEE Transactions on Consumer Electronics*, 56(4):2592–2600, 2010.

[110] S. Yao. Modeling Quality-of-Experience of 360° videos in head-mounted virtual reality. Master's thesis, National Tsing Hua University, 2018.

[111] R. Youvalari, A. Aminlou, M. Hannuksela, and M. Gabbouj. Efficient coding of 360-degree pseudo-cylindrical panoramic video for virtual reality applications. In *Proceedings of the 2016 IEEE International Symposium on Multimedia*, ISM'16, pages 525–528, 2016.

[112] M. Yu, H. Lakshman, and B. Girod. Content adaptive representations of omnidirectional videos for cinematic virtual reality. In *Proceedings of the 2015 International Workshop on Immersive Media Experiences*, ImmersiveMe'15, pages 1–6, 2015.

[113] V. Zakharchenko, K. Choi, and J. Park. Quality metric for spherical panoramic video. In *Proceedings of the 2016 SPIE, Optics, Photonics: Optical Engineering, and Applications*, OP'16, pages 9970–9979, 2016.

[114] A. Zare, A. Aminlou, M. Hannuksela, and M. Gabbouj. HEVC-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the 2016 ACM Multimedia Conference*, MM'16, pages 601–605, 2016.

[115] L. Zhang, B. Tiwana, R. Dick, Z. Qian, Z. Mao, Z. Wang, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the 2010 IEEE/ACM/IFIP International*

Conference on Hardware/Software Codesign and System Synthesis, CODES ’10, pages 105–114, 2010.

[116] C. Zhou, Z. Li, and Y. Liu. A measurement study of Oculus 360 degree video streaming. In *Proceedings of the 2017 ACM Multimedia Systems Conference*, MM-Sys’17, pages 27–37, 2017.

[117] A. Zomet, A. Levin, S. Peleg, and Y. Weiss. Seamless image stitching by minimizing false edges. *IEEE Transactions on Image Processing*, 15(4):969–977, 2006.