UNIVERSITÄT
DUISBURG
ESSEN

Offen im Denken

Institut für Informatik und Wirtschaftsinformatik (ICB)
Fachgebiet Kommunikationsnetze und -systeme
Prof. Dr.-Ing. Amr Rizk

AN EXTENDABLE DASH STREAMING SYSTEM FOR DYNAMIC POINT CLOUDS

Bachelorarbeit

vorgelegt der Fakultät für Wirtschaftswissenschaften
der Universität Duisburg-Essen (Campus Essen) von

ÖZGÜR EROL
B.Sc.

Lisztstraße 4
42549 Velbert
Matrikelnummer: 3097359

16. September 2022

Betreuung:	Michael Rudolph, M.Sc.
Erstgutachter:	Prof. Dr.-Ing. Amr Rizk
Zweitgutachter:	Prof. Dr. Stefan Schneegass
Studiengang:	Wirtschaftsinformatik (B. Sc.)
Semester:	6

ABSTRACT

Point clouds are increasingly being used for representing objects in three-dimensional space. They create high-quality models and are commonly utilized in automotive and entertainment applications. Although point clouds rise in popularity, they are characterized by large file sizes, which complicates real-time transmission. This thesis examines the current state of point cloud streaming and presents a system that supports future development. By implementing Dynamic Adaptive Streaming over HTTP (DASH), the system is able to adapt to the network condition of the user. Moreover, the proposed system combines the strengths of existing solutions and includes several evaluation methods to facilitate the creation of new functions. We performed a number of simulations, to present the functionality of the system and compare the findings of this thesis. The results suggest that point cloud streaming requires high bandwidth speeds and that a wireless connection is more prone to quality degradations, compared to a wired configuration.

ZUSAMMENFASSUNG

Punktewolken werden zunehmend zur Repräsentation von Objekten im drei-dimensionalen Raum genutzt. Sie schaffen hochqualitative Modelle und werden oftmals für Anwendungen im Automobil- und Unterhaltungsbereich verwendet. Obwohl Punktewolken an Attraktivität gewinnen, sind diese durch große Dateigrößen charakterisiert, was die Übertragung in Echtzeit erschwert. Diese Arbeit untersucht den aktuellen Stand von Punktewolken-Streaming und stellt ein System zur Unterstützung zukünftiger Entwicklungsschritte vor. Durch das Einsetzen von Dynamic Adaptive Streaming over HTTP (DASH), ist das System in der Lage auf Netzwerkänderungen des Nutzers zu reagieren. Das entwickelte System kombiniert die Stärken bestehender Lösungen und enthält mehrere Auswertungsmethoden, die das Erstellen neuer Funktionen vereinfachen. Wir haben eine Menge an Simulationen durchgeführt, um die Ergebnisse der Arbeit zu vergleichen und die Funktionalität des Systems vorzustellen. Die Ergebnisse deuten darauf hin, dass Punktewolken-Streaming hohe Bandbreite voraussetzt und dass eine kabellose Verbindung anfälliger für Qualitätseinbuße ist, im Vergleich zu einer kabelgebunden Konfigurierung.

CONTENTS

LIST OF FIGURES

LISTINGS

1 INTRODUCTION

While traditional video is still considered to be highly valuable, the idea to create immersive environments has gained large attention from both practice and research. The emergence of cost-effective and accessible head-mounted displays (HMDs) attributes to the growing interest in immersive multimedia applications [9], allowing the user to change perspective and experience content to a greater extent.

As an example, consider the use of 360-degree videos, which represent scenes that can be observed from every angle [4]. They provide the user with an enhanced interactive experience compared to traditional video and play an important role in modern virtual reality applications.

However, although 360-degree video has led to a renewed interest in immersive media, it limits the user's freedom by fixing the camera to a single location. In contrast, a volumetric approach is free from this limitation: By placing the objects in a three-dimensional scene, the user is able to detach from his position and explore the environment without any restrictions.

Within this area of investigation, point clouds seem to be the most favorable format to represent volumetric data [15].

Point clouds form an intuitive approach to the representation of three-dimensional objects. They represent high-quality scenes, where each point defines a position and may contain other additional information, like color [7].

Here, the density of a point cloud greatly impacts the overall quality. A higher amount of points results in a high-quality object, while a lower amount may lead to quality degradation. An example for a point cloud object is given in Figure 1.1, which contains about 120.000 points.

Point clouds have been extensively researched and find application in many areas that

range from the preservation of cultural heritage to new-found forms of entertainment and communication [7].

With regard to the automotive industry, point clouds are used to support autonomous driving by evaluating the environment with mobile mapping [5].

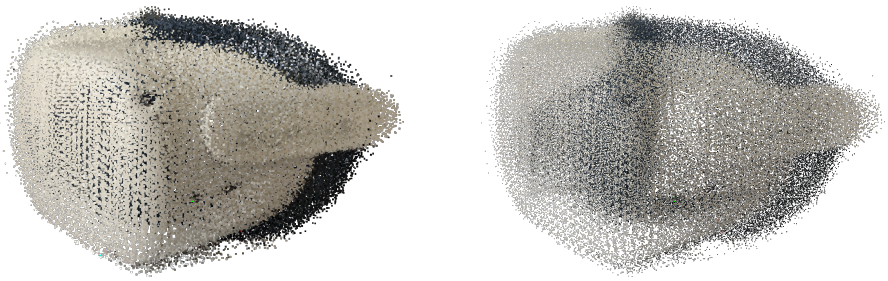

Figure 1.1: An example for a point cloud, featuring an HMD. The second object features the same point cloud with a smaller point size

For entertainment purposes, modern virtual and augmented reality (VR/AR) applications may benefit from an increased degree of freedom [9], allowing the user to experience new levels of immersion.

1.1 PROBLEM STATEMENT AND CONTRIBUTION

Research suggests that the biggest problem revolves around the size of point clouds, resulting in massive storage requirements. More notably, the transmission of the enormous amount of data seems to be a reoccurring issue. Therefore, point clouds streaming requires high connection speeds, which can cause problems with bandwidth-limited devices, such as mobile phones [11].

To illustrate, a bandwidth of 3.6 Gbps is required to stream a dynamic point cloud at 30 frames per second [7], given the amount of 1 million points per frame. These bandwidth requirements reach a substantial level, considering a connection speed of only 15 to 20 Mbps is sufficient to stream 4K videos on major video platforms [18, 30].

Accordingly, considerable research attention has been directed toward the reduction of size and the transmission of point clouds. Along with applying compression to point clouds, recent studies have explored the use of adaptive bitrate streaming.

Similar to traditional video, the user's network condition plays an essential role. If the user can't meet the bandwidth requirements of the stream consistently, he may experience long starting times or rebuffering. By applying adaptive bitrate streaming, the system is able to react to bandwidth changes and adapt the quality of the stream accordingly, to ensure a proper viewing experience for the user.

The goal of this thesis is to explore various design choices to create an extendable point cloud streaming system that supports adaptive bitrate streaming. We consider the use Dynamic Adaptive Streaming over HTTP (DASH), which is widely applied to modern video services.

The proposed system presents an efficient way to stream point cloud data and attempts to make point cloud streaming available to a wide range of consumers.

The contributions of this thesis are twofold. First, it provides a better understanding of the challenges and the contributions made for point cloud streaming. Second, it provides a framework that facilitates the comparison and evaluation of new solutions, promoting future development.

1.2 OUTLINE

The remainder of this thesis is structured as follows. Chapter 2 addresses point clouds and gives an overview over the topic of streaming. The next chapter outlines the research and progress made on point cloud streaming. The following two chapters introduce the proposed point cloud streaming system and discuss features, as well as design decisions. Chapter 6 carries out an extensive study of the system and evaluates the results. Lastly, Chapter 7 recaps the main contributions and describes limitations, prior to providing an outlook on future work.

2

BACKGROUND

This chapter establishes a general understanding of the topic of point clouds and presents their use in practice. Next, it gives an overview of the history of streaming and follows with the idea of adaptive bitrate streaming prior to discussing Dynamic Adaptive Streaming over HTTP. Finally, the concept of quality of experience is presented.

2.1 POINT CLOUDS

The introduction of point clouds enables a new form of media representation that is characterized by high-quality graphics and an increased degree of immersion. Due to the simple nature of point clouds, they differentiate themselves from other volumetric data formats by being easy to acquire and render [26].

For this reason, point clouds are commonly used to represent volumetric data [28] and can be acquired by passive or active methods, including a combination of the two [7]. Passive methods make use of a number of RGB cameras, which capture an object from multiple angles [15].

Consequently, image matching and spatial triangulation are applied to project the object onto three-dimensional space [7]. Active methods involve the use of depth scanners [11], which are able to capture millions of points within a short time frame.

The result is a set of points, where each point is defined by Cartesian coordinates (X, Y, Z). A point may contain additional attributes that are relevant for a specific scene, such as color or reflectance [7].

After a point cloud is acquired, it can be rendered by mapping every point to a three-dimensional space. In addition, a camera is added to the scene, which enables the user to navigate freely and observe the point cloud from any angle.

Point clouds can be further classified into static point clouds and dynamic point clouds [5]. While the former category represents static objects, like historical buildings or statues, the latter involves movement and consists of multiple frames. For an example of a dynamic point cloud scene, refer to Figure 2.1.

Figure 2.1: An example for a dynamic point cloud scene, featuring longdress [31]

Although the idea of point clouds sounds promising, it comes with a set of challenges that need to be addressed. Research suggests that the main problem seems to revolve around the size of point clouds. They may contain billions of points [5] and therefore, result in massive file sizes.

This makes them particularly resource intensive, which complicates streaming and rendering them for practical use [11]. With regards to dynamic scenes, the transmission of point clouds is paired with time-sensitive delivery and requires low latency [15].

As a result, the size of point clouds has to be considered to effectively transmit them to contemporary devices. This can be accomplished by employing well-established methodologies from traditional video streaming, which will be discussed in the following.

2.2 STREAMING

There are two modes of acquiring and watching video files from the internet: download mode and streaming mode [6]. In download mode, the user requests the video file and starts playing the video once it is fully downloaded on the computer.

In streaming mode, however, the user is able to start the playback while the video file is in the process of being transmitted, which is more preferable to users [13]. Additionally, the need for storing the entire video file ceases to exist as the streamed segments are deleted after the user is done watching them.

In its early stages, the Real-time Transport Protocol (RTP) was used with the User Datagram Protocol (UDP) to deliver video and audio content over the internet [13]. Although the RTP was used frequently in the early stages of video streaming, it eventually was replaced with other protocols. This can be largely attributed to the rising prevalance of content delivery networks (CDNs), which lack support for RTP streaming [24].

The interest in CDNs stems from a number of reasons: by using CDNs, service providers are able to reduce the load on origin servers and oppose latency problems [25]. As a result, the geographic distribution of servers is helpful in closing the gap between the content provider and consumer and avoids the problems that originate from long-distance communication.

Streaming over HTTP has a number of advantages that RTP can't produce. These include [24]:

1. The internet infrastructure along with CDNs provides support for HTTP
2. HTTP requests can pass through any firewall
3. The server does not store session states which would otherwise limit the maximum number of parallel clients

HTTP generally runs over TCP, which is considered as a reliable and efficient protocol [29]. By implementing TCP, the video stream is relieved from packet loss, that otherwise could lead to missing frames and result in decreased video quality [16].

Today, modern streaming platforms such as Apple's HTTP Live Streaming (HAS) or Adobe's HTTP Dynamic Streaming HDS use HTTP to efficiently deliver media content across the internet [23].

2.3 ADAPTIVE BITRATE STREAMING

As opposed to regular video streaming, adaptive bitrate streaming is used to adjust the quality of a stream to the network conditions of the user [11]. Likewise, applying adaptive bitrate streaming can be helpful in reducing transmission overload and thus, prevents buffering or long starting times for bandwidth-limited users.

To illustrate, the viewer may experience drops in bandwidth speed, as other devices in the same household could engage in bandwidth-heavy activities, which limits the available bandwidth for anyone else. By having a video player that supports adaptive bitrate streaming, the player is able to detect drops in bandwidth, and the video stream is switched to a suitable quality representation. As a result, the user can continue to watch the video without interruptions.

Storing different quality representations on the server is not only beneficial for combating bandwidth changes of the user; it can be instrumentalized to create subscription plans that include a set range of possible streaming qualities to which the user can respond based on his needs [23].

To make adaptive bitrate streaming possible, the initial video has to be encoded at multiple bitrates which result in different qualities. Depending on the level of encoding, the new video representations may result in a high amount of bandwidth savings. These representations are then stored on an HTTP server, together with a manifest file. The manifest file contains information about the different representations, such as required bandwidth or URI [11]. The client may now request said file and start downloading the appropriate quality representations. Here, the literature differentiates between server-side and client-side adaptive schemes [29].

In short, server-side schemes require the client to regularly inform the server about streaming metrics, namely the current bandwidth or buffer status. The server then decides which representations fit the user best. On the contrary, client-side schemes manage choosing and requesting quality representations autonomously.

Irrespective of the chosen scheme, the client starts the streaming process by requesting the manifest file. Since there is no information on the user bandwidth prior to starting

the video, the system can only adapt to the current network conditions after the first segments have been requested and downloaded.

A simple approach would be to select the highest quality available [24], however, an average quality or the lowest quality may be considered to reduce initial loading times. Then, the client requests the appropriate quality representations to fill the buffer and starts adapting to the user's bandwidth condition, which plays a vital role: based on its current state, the system chooses the highest quality representation possible while maintaining continuous playback. Accordingly, the streaming process continues by adapting to the network condition of the user and with the intention of freeing him from any disturbances.

In addition, while the client requests the quality representations that fit the user bandwidth, it is important to keep the buffer at a certain threshold. If the buffer runs out of video segments, the playback stops and the user experiences a stall [23]. It is only when the buffer is adequately filled that the video can continue playing.

Although all modern streaming platforms use adaptive bitrate streaming solutions, their implementations differ in terms of manifest and segment formats [24]. Moreover, it is worth noting that the idea of adaptive bitrate streaming is not limited to video content and can be extended to every medium, such as audio or subtitles.

Adaptive bitrate streaming is not only utilized in video streaming; it is highly prevalent in other multimedia applications, such as 360-degree videos and can be extended to point clouds, to overcome the massive transmission sizes that are associated with them [11].

2.4 DYNAMIC ADAPTIVE STREAMING OVER HTTP

Dynamic Adaptive Streaming over HTTP (DASH) is one of many realizations of adaptive bitrate streaming.

It was developed under MPEG, which issued a Call for Proposals in 2009, resulting in a technology that differentiates itself from other implementations by being an international standard [24].

Not only is the existence of a standard detrimental to market growth, but also effective in increasing operability and effectiveness across different devices and service providers [24].

By using DASH, a Media Presentation Description (MPD) file is stored on the server, along with the different quality representations. The streaming information is stored in the MPD in the XML format and the client can then acquire the file by performing an HTTP GET request and extracting the relevant information to start the video playback. This includes the number of periods, qualities, their file format, bandwidth requirements, and other information that are useful for the associated setting [24].

It is worth noting that DASH does not define an adaptation algorithm [29]. Additionally, any system that implements DASH can choose how the media is encoded or how the MPD file should be acquired [24]. In any case, DASH requires the client to implement its own adaptation logic [25], making it a client-side adaptation scheme. As a result, the client is responsible for analyzing the state of the user and for selecting the proper quality representations, without sending the information to the server.

Similar to adaptive bitrate streaming, the concept of DASH is not limited to video or audio files. Rather, it can be extended to any media format that can be streamed to the user, including point clouds.

2.5 QUALITY OF EXPERIENCE

To evaluate the efficiency of a streaming system, the effects on the user have to be observed. This is done by measuring the quality of experience (QoE) of the user, which can serve as a reference point to maximize satisfaction [23]. Accordingly, the measured QoE may be used to manage streaming performance and adapt to the preferences of the user [19].

For example, the duration between starting a video stream and watching the first segments is kept small at best. However, it is unknown which amount is tolerable for the user and how significant the influence is. By assessing the QoE, the streaming system is aligned with the needs of the user and disturbances of any kind can be captured.

There have been a number of contributions on the aspect of QoE. Mok *et al.* [16] performed a study with 10 subjects and conclude that rebuffering frequency can be considered

as the main factor that has a negative effect on the QoE. At the same time, initial delay and the mean rebuffering duration seemed to be tolerable and the authors suppose that viewers are accepting of long starting times, provided that the rest of the stream runs smoothly. Thus, for the purpose of creating long-lasting streaming solutions, the impressions and satisfaction of the users have to be taken into account.

In the context of point clouds, it is important to grant the user a high visual quality to improve the QoE. However, a high-quality representation of an object may contain billions of points [11], which greatly increases file size and thus, complicates transmission. In contrast, a low-quality representation may reduce disruptions whilst streaming, but decrease the perceived quality and satisfaction of the user.

Ultimately, the value of a streaming service is greatly shaped by the perceived quality of the people who interact with it [23].

2.6 SUMMARY

Since the difficulties surrounding point cloud streaming overlap with traditional video streaming, the solutions and adaptations of video streaming may enable a reference point that is helpful in making point clouds more accessible to end-users.

By combining the ideas of compression and adaptive bitrate streaming, point clouds can be optimized for the consumer market.

3

RELATED WORK

This chapter presents the current state of point cloud streaming. It discusses the progress made in reducing the size of point clouds by compression and evaluates the attempts on making point cloud streaming accessible. In addition, it introduces 360-degree video to reveal possible streaming solutions.

3.1 COMPRESSION

The immense size of point clouds complicates transmission to bandwidth-limited devices, which ultimately limits their potential use. As mentioned, the transmission of high-quality point clouds results in immense bandwidth requirements and thus, requires high connection speeds.

In recent years, several attempts have been made to decrease the file size of point clouds and thus, make real-time transmission possible. Similar to traditional media, the file sizes of point clouds can be reduced by compression.

Although there have been a number of contributions made on the topic of point cloud compression, most compression methods have focused mainly on static point clouds [22].

Previous studies concentrated on tree data structures to compress point clouds, namely octrees. An octree creates a hierarchical structure where each internal node has eight children and is used to spatially partition the point cloud [17] and effectively reduce file size. While a number of studies have focused on octrees [12, 21] for point cloud compression, considerable research attention has been directed towards the compression standard proposed by The Moving Picture Experts Group (MPEG).

Utilizing the rising interest in point cloud technology, MPEG issued a call for proposals in 2017 [1], which lead to two main compression technologies: geometry-based point cloud compression (G-PCC) and video-based point cloud compression (V-PCC) [22]. The former

method compresses the point cloud by employing octrees [5] [7]. Although the position of each point in a point cloud is generally stored in floating point values, G-PCC requires the use of integer values [7]. This conversion is considered to be beneficial since integers require less processing power and improve memory usage [2]. However, the conversion likely leads to quality degradation, due to data loss.

While the former technology is used in conjunction with static and dynamically acquired point cloud data, the latter technology addresses dynamic content [5, 22]. In the context of this thesis, we are primarily concerned with dynamic scenes and thus, use V-PCC to compress the point cloud data.

The main idea of V-PCC is to project the point cloud into two-dimensional space [22] and compress the data using existing video codecs. The concepts behind V-PCC are summarized in the following [14, 22]:

1) *Patch Generation*: 3D patches are generated by first estimating the normal for each point in a point cloud. Next, clusters are formed by examining the orientation of the underlying surface and projecting it onto one of six planes. Finally, the adjacent points in the clusters are referred to as patches and are processed in the next step.

2) *Patch Packing & Occupancy Map Generation*: The generated patches are now positioned on a 2D grid. While placing the patches on the grid, the algorithm aims at minimizing the unused space and assures that each patch is free from overlapping. In addition, an occupancy map is generated. It represents a binary map where the value 1 indicates that the block of the 2D grid is occupied and 0, when it's empty.

3) *Image Generation & Padding*: Image generation describes the process of storing the geometry and texture information of a point cloud in images. While projecting the information on 2-dimensional images, it is possible that multiple points are mapped to the same pixel. V-PCC uses a near layer and far layer for each patch and compares the depth values for each point, to identify and prevent duplicates. Lastly, padding is applied to ensure a smooth transition between patches, which is beneficial for video encoding.

4) *Video Encoding*: The resulting geometry and texture images, along with the occupancy map can now be compressed using traditional video encoders. The video encoder for

V-PCC currently uses High Efficiency Video Coding (HEVC) and by specifying different quantization steps, the point cloud can be processed into multiple quality representations.

It is worth noting that both G-PCC and V-PCC require additional software, due to the lack of dedicated hardware support [20].

Overall, the proposed compression technologies offer promising results and are effective in reducing file size. However, they come with a set of problems that will be discussed in the last section of this chapter.

3.2 POINT CLOUD STREAMING

Recent work was conducted by Hosseini & Timmerer [11] who present DASH-PC, a system that follows the idea of DASH and creates different quality representations by the use of subsampling. The authors put forward three algorithms and by removing the total number of points, they expect to reduce the file size and decrease the processing power that is involved in rendering a point cloud.

They extend their findings by examining the concept of visual acuity and come to the conclusion, that the low quality that is associated with low-density point clouds is most likely not visible to the human eye, given the distance between is at a certain threshold.

To summarize, their results suggest by reducing the density of point clouds, considerable performance and bandwidth gains are made possible, without a significant decrease in quality.

Another body of work was conducted by van der Hooft *et al.* [9], in which they propose PCC-DASH. In contrast to DASH-PC [11], their system focuses on streaming multiple dynamic point cloud objects and takes additional metrics into consideration, like buffer size or camera position and angle.

Moreover, four rate adaptation heuristics are presented along with three bitrate allocation schemes. The adaptation heuristics rank the point cloud objects by certain metrics, such as distance to the user or visual area. Next, a bitrate allocation scheme is chosen to select the appropriate quality representations.

To illustrate, the point clouds may be ranked by the Euclidean distance to the user. Therefore, objects that are closer and hence, more relevant to the scene, are given higher importance. Next, by using the greedy allocation scheme, the available bandwidth is used to assign the highest quality possible to the closest object. The remaining bandwidth is then allocated to the next objects, repeating the same approach until all objects are assigned the appropriate quality.

Three distinct scenes were created to evaluate the system and the results indicate that each setting has to be examined individually and that an all-purpose streaming solution for point cloud streaming has yet to exist. The authors further draw attention to the importance of an accurate viewport prediction, which in conjunction with the hybrid bitrate allocation scheme, seems to outperform the other allocation schemes in terms of quality and bandwidth savings.

Additionally, they highlight the influence of the buffer size on the video quality. The findings suggest that a smaller buffer size correlates with higher video quality, while a bigger buffer is more resistant to stalls.

Lastly, Qian *et al.* [20] present Nebula, a streaming solution for point clouds that focuses on mobile phones. Their system aims at delivering point cloud data to bandwidth-limited and resource-constrained devices. In contrast to other systems, Nebula considers the network behavior in wireless networks, which are more prone to bandwidth fluctuations than wired networks [27]. At the same time, the authors extend their adaptive bitrate streaming algorithm by the user's viewport, to prevent unnecessary bandwidth constraints.

For instance, a multi-object scene can feature multiple point clouds, but the user may only be interested in a specific object or a portion of the scene. Therefore, considerable bandwidth savings are made possible by focusing on the objects that are within the user's viewport.

With regard to the processing power that is involved with decoding the point cloud data, the authors argue that mobile devices can't reproduce the performance of regular PC hardware. For this reason, they suggest relocating the decoding process by using edge computing. As a result, the client regularly sends position and viewport information to

the server, which is responsible for transcoding the point cloud into a 2-dimensional video. Finally, the client receives the transcoded video and applies traditional video decoding techniques, requiring much less processing power than applying MPEG's compression technologies.

This section has demonstrated that point cloud streaming solutions mostly consist of two parts: (1) To use adaptive bitrate streaming, multiple quality representations are required. This can be achieved by either performing compression with different quantization steps or by subsampling the point cloud using different intensities. (2) Moreover, an adaptation heuristic is required to effectively exploit the information on existing point clouds, including the position and orientation of the user. As a result, the available bandwidth is optimally utilized and a good user experience is maintained.

3.3 360-degree video streaming

Over the years, a considerable amount of research has been devoted to adaptive 360-degree video streaming. Similar to point clouds, the transmission of 360-degree video in real-time remains to be an issue, due to high bandwidth requirements and time-sensitive delivery [29]. Therefore, we intend to apply the ideas and concepts of 360-degree video streaming to point cloud streaming by exploiting the similarities between the two representation formats.

Corbillon *et al.* [4] put forward the idea to use viewport-adaptive streaming to efficiently utilize the available bandwidth. They create different quality representations of the same 360-degree video, where each representation offers a different *Quality Emphasized Region*.

In other words, they ensure that the user sees a high quality representation of the video, while the surrounding parts are depicted in a lower quality. Consequently, the available bandwidth is optimally utilized and ideally, the user is unaware of the quality differences. However, in case of a sudden change in head movement, the user is confronted with a low quality representation of the video, up until the video stream has adapted to the new situation.

In addition, they suggest that a shorter duration for video segments is helpful in supporting frequent switches. Therefore the authors hope to react swiftly to changes in network stability and viewport orientation.

A similar approach was pursued by Hosseini & Swaminathan [10]. They suggest partitioning the 360-degree video into tiles, where each tile represents a portion of the video and is assigned multiple quality representations. After the user starts the video stream, an adaptive bitrate streaming algorithm determines the appropriate quality of the tiles, based on the user's viewport.

The findings reveal a high rate of bandwidth savings, effectively reducing bandwidth usage for up to 72%. Moreover, the authors claim that their system creates only minor quality degradation and thus, provides a good QoE for the user.

3.4 ANALYSIS OF RELATED WORK

Despite the recent contributions made on the topic of point cloud streaming, important questions regarding the practicability and efficiency of the results remain unanswered. This section will investigate the advantages and drawbacks of the proposed solutions.

The findings of Hosseini & Timmerer [11] give an outlook on how subsampling is efficient in decreasing point cloud data, whilst maintaining good visual quality. However, their findings are limited, as they only consider single point cloud objects.
Therefore, it is unclear to which extent their results can be applied to multi-object scenes. Consequently, the proposed MPD file lacks important attributes that are necessary in immersive settings, namely position, rotation or size of point cloud objects.

The issue of multi-object scenes was addressed by van der Hooft *et al.* [9]. Although the proposed rate heuristics are helpful in ranking the importance of the objects, they do not entirely consider the intentions of the user.
For instance, the user may wish to traverse the scene to move to a point of interest. During his course, the objects are constantly ranked and high quality representations are requested, without being of interest to the user. As a result, a high amount of bandwidth is wasted and the user may experience long waiting times when he comes to a stop.

Likewise, while the Euclidean distance presents a simple solution to ranking the point cloud objects by importance, it does not consider the user's viewport. For instance, an occluded object may be allocated an essential amount of bandwidth, even though the user

is not capable of seeing it. Consequently, objects that may take a large portion of the user's view are given a lower quality, which ultimately degrades the QoE.

Similarly, the proposed camera traces are artificial in nature and may not realistically represent the user's behavior. As a result, quick head movements and unexpected movements may have a detrimental effect on the adaptation algorithm, leading to a poor user experience.

A number of current point cloud streaming solutions estimate the user's bandwidth by dividing the file size of the requested objects by the time taken to download the segments. However, this approach is prone to short-term fluctuations and increases the number of quality switches [27].

It is generally agreed that a buffer plays an important role in a streaming system. It is helpful in utilizing the user's bandwidth and avoiding disruptions, such as stalls. However, there seems to be no agreement on the optimal buffer size.

In discussing the buffer size, van der Hooft *et al.* [9] argue that a lower value translates to higher interactivity and better video quality. At the same time, the findings of Liu *et al.* [15] suggest that a larger buffer may be of advantage, particularly when high bandwidth fluctuations are expected.

Accordingly, it is crucial to consider the network behavior of the user to accurately define an adequate buffer. In the context of this thesis, we will look into different values for the buffer size and attempt to find a general solution.

Despite the effectiveness of MPEG's compression, the processing power that is involved in decoding and rendering the compressed point cloud objects is far too great and ultimately prohibits the possibility of streaming them in real-time [9]. Therefore, an improvement in compression performance is crucial to make real-time point cloud streaming feasible.

At the same time, an increased interest in edge computing has emerged in recent years. With regards to point cloud streaming, Liu *et al.* [15] suggest that mobile edge computing may be beneficial in decreasing the computational burden that is involved in decoding. This idea was pursued by Qian *et al.* [20], who used edge computing to relocate the

decoding process to a more powerful system.

However, it is unknown how many concurrent clients can be supported, since a high amount of traffic could quickly deplete the available resources of the edge server. Therefore, the employment of edge computing solutions have to be considered with great care.

Finally, the three presented point cloud streaming systems [9, 11, 20] do not consider the combination of ideas, such as the interplay of subsampling and compression techniques. Rather, they focus on one approach and fail to evaluate the strengths and shortcomings of the combination of solutions.

Therefore, we intend to create a system that not only investigates the effects of each decision but combines well-established principles and facilitates future development.

3.5 SUMMARY

Considering the recent interest in point cloud streaming, multiple contributions have been made that could enable real-time transmission in the future. Although the proposed systems are a driving factor and offer important results, they are unsuccessful in providing a general solution for a multitude of constellations and settings.

To improve and advance the improvement of point cloud streaming, we propose a system that supports rapid prototyping and provides extensive feedback.

4

DESIGN

To further investigate the findings of this thesis, we implemented our own point cloud streaming system. This chapter gives an overview of the proposed system and identifies key modules in the streaming process.

Moreover, it provides a better understanding of the underlying principles that enable point cloud streaming and attempts to evaluate the effectiveness of the proposed design decisions in the previous chapter.

The system aims at supporting the creation of new methods and thus, provides a framework for future development. It includes several methods to compare and evaluate each solution and gives an outlook on the QoE of the user.

By using the compressed point cloud files from the V-PCC encoder, the client is provided with a number of different quality representations. Furthermore, the client implements DASH to not only react to bandwidth changes but to provide the user with the highest quality possible.

Ultimately, the system features a client and a server, which are described in the following.

4.1 REQUIREMENTS AND ASSUMPTIONS

Although the proposed system utilizes compressed point cloud files, it is not responsible for engaging in the encoding process. In other words, the compressed point cloud files are required to be on the server and are independent from the system.

At the same time, the client does not perform decoding mechanism to decompress the point cloud files. As mentioned, the computational power is significant, which prevents the possibility of real-time decoding. Consequently, we decided to not include a rendering module for the client.

4.2 SERVER CONFIGURATION

Starting with the server, it is mainly concerned with storing the required files. This includes the MPD and the point cloud files, both in their raw and processed form. Moreover, it is responsible for creating the MPD files for the client. Figure 4.1 gives an overview of the server.

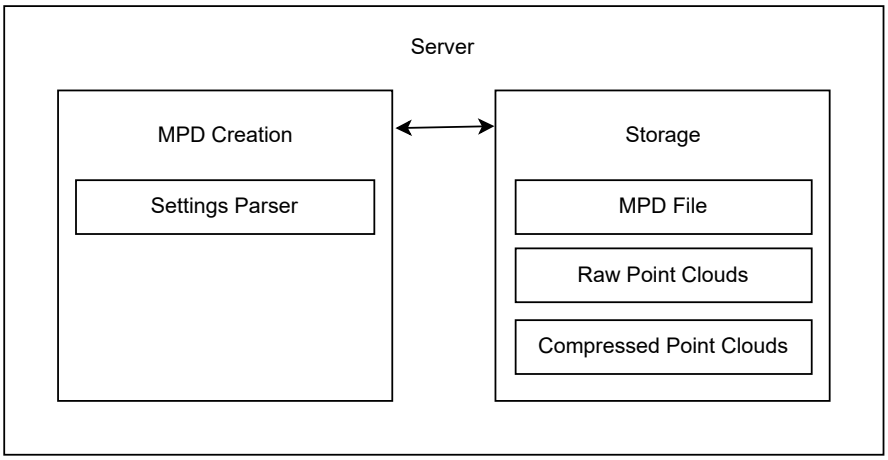

Figure 4.1: Architecture of the server

Although the raw point cloud data poses difficulties for server storage, it is needed to extract necessary metrics of the point cloud objects, such as density or minimum bounding box. Accordingly, the information is stored in the MPD file.

With regards to regular video streaming, the MPD file contains periods that are characterized by start time and length and store a number of adaptation sets [24].

The adaptation sets contain information about different representations along with their quality. Each representation consists of one or multiple segments, that extract a small section of the original video.

We make use of this structure to create our own MPD, which is customized for point cloud streaming.

Finally, the MPD creation module features a configuration file, which specifies the proper-

ties of the MPD. It contains basic information, such as the title of the point cloud scene. In addition, the position and rotation for each point cloud object can be defined.

Decorating the MPD file with the position of point cloud objects is not only necessary to place them in the scene, but useful for an adaptive bitrate streaming solution: occluded objects may be given a lower quality representation, resulting in great bandwidth savings.

Lastly, the configuration options are organized into sections, providing a clear overview of the different settings.

After the settings file is parsed, the MPD creation module applies the defined values to the MPD. As mentioned, the MPD file contains the relevant information for the streaming process, mainly the different point cloud objects and their quality representations.

The module further defines the size of each point cloud object and specifies the bandwidth needed for each segment. Here, the quality representation with the highest bandwidth requirement produces the best visual quality.

4.3 CLIENT CONFIGURATION

Moving forward, the system's main functions are on the client side. The client communicates with the server to obtain the point cloud files and implements the adaptation algorithm.

Generally, there are three main decisions to consider when developing an ABR algorithm [23]: choosing the next segments, deciding when to request them and managing the video buffer.

Accordingly, its main responsibility is the selection of the appropriate quality representations that fit the user best, while maintaining a good QoE.

In addition, the client features multiple evaluation functions. It is capable of capturing and monitoring relevant data to not only perform simulations but evaluate the performance of the streaming process. This may include bandwidth speed or buffer history.

As a result, the proposed system assists in fast prototyping and creates log files or diagrams, to store the results of each streaming activity. An outline for the client is given in Figure 4.2.

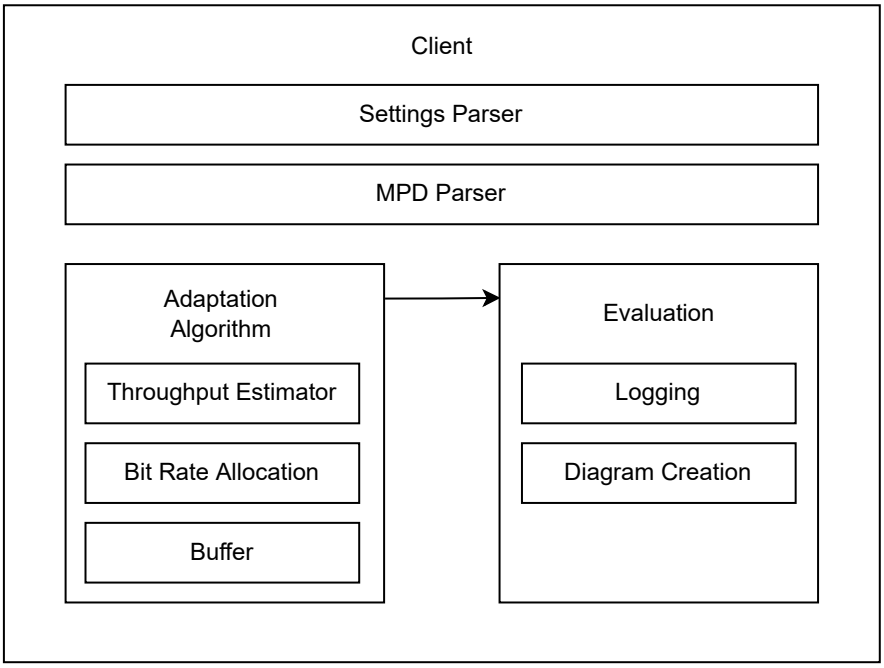

Figure 4.2: Architecture of the client

Similarly to the server, a separate configuration file is given to the client. It specifies starting options and controls the behavior of the adaptation algorithm. In addition, it assists the evaluation process by defining settings for diagram creation or logging. Lastly, it defines the necessary information for the MPD parser.

Given that the MPD file is ready for transmission, the client sends a request to the server to retrieve the necessary information. Consequently, it extracts the relevant data and maps it according to the appropriate classes.

Due to the implementation of adaptive bitrate streaming, the system is able to detect changes in bandwidth and switch the point cloud quality to a suitable quality representation.

The proposed system uses DASH to select appropriate quality representations, that are

compatible with the user's bandwidth. However, the adaptation algorithm itself is independent of DASH. Therefore, the client is expected to implement its own adaptation logic.

Regardless of the chosen adaptation scheme, the user's network condition plays an important role. Therefore, the system has to be able to capture the relevant information, namely the user's bandwidth speed.

As mentioned, stalls and buffering greatly reduce the perceived quality of the system. To prevent these disturbances, the system introduces a buffer. Accordingly, the point cloud segments are first stored in memory, given the buffer size is below the set minimum buffer size.

After the buffer is adequately filled, the system may send the segments to a renderer, which displays the point clouds to the user.

Finally, the information for each streaming session is stored in log files, enabling evaluation and comparison of performance. For instance, the client may store the estimated bandwidth speed or the buffer history. In addition, the resulting log files may be used for the diagram creation module.

The main goal of the diagram creation module is to provide the developer with a better understanding of the log files. Ultimately, it gives a descriptive representation of the system results and does not require any additional measures.

4.4 SUMMARY

The proposed system uses well-established methods from both traditional video streaming and point cloud streaming. By using both a server and a client, we are able to reproduce a real streaming situation.

In addition, the evaluation methods are helpful in assessing the quality of the system configuration and thus, support future development.

5 IMPLEMENTATION

The following chapter gives an extensive overview of the major functions of the proposed system and proposes a customized MPD for point cloud streaming. Additionally, it presents the different configuration options and evaluation methods, which were described in the previous chapter.

The client features a headless python application, which displays the relevant information in the console. Its behavior is specified in the settings file, which will be discussed in the last part of this chapter.

5.1 ADAPTATION ALGORITHM

One of the main goals of the systems is to facilitate the integration of new methods, which includes supporting the creation and evaluation of adaptation heuristics. For this purpose, the client provides several helper functions.

In addition, the client features two adaptation heuristics, namely dash basic and dash distance.

The most basic idea of bandwidth adaptation is to request the same quality level for every point cloud object, given the total bandwidth requirement of the quality level is below the user's estimated bandwidth. As a result, the user is given the best quality possible.

Although this approach follows the idea of adaptive bitrate streaming, it leaves out potential performance gains. In other words, the available bandwidth is distributed to each object evenly, although not all objects are of interest to the user.

Therefore, we use a distance adaptation method to take the user's position into account, similar to van der Hooft *et al.* [9].

To illustrate, the user's position may be used to order the point cloud objects by importance. By allocating more bandwidth to objects close to the user, the available resources are optimally utilized.

However, before the system can calculate the distance to the cloud objects, the position of the user is required. In a real scenario, the client would acquire the relevant metrics from a HMD.

The system uses the Euclidean distance to determine the distance to each object. After the objects are ranked by distance, the available bandwidth is distributed among the objects. Here, we use two of the three bitrate allocation schemes proposed by van der Hooft *et al.* [9], namely greedy and uniform bitrate allocation.

The former allocation method determines the highest quality possible for each point cloud object, given the available bandwidth. In other words, the highest ranked object is allocated the most bandwidth, while the remainder is used for the descending objects.

Although this method guarantees the best visual quality for the closest object, it creates a high contrast in quality to farther objects. Primarily, this becomes an issue when the objects have a similar distance to the user. Under those circumstances, the second bitrate scheme might prove to be more suitable.

The bitrate allocation scheme incrementally increases the quality of the point cloud objects, starting with the highest-ranked point cloud object. If the available bandwidth is not sufficient, the object either maintains or decreases its quality level.

Depending on the structure of a scene, this approach may produce a more gradual change in visual quality compared to the greedy bitrate allocation scheme.

Finally, the interplay of efficiency and good user experience has to be greatly considered. Although a more sophisticated approach may produce significant bandwidth savings, it can lead to noticeable quality degradation. Therefore, it is important to consider the structure point cloud scene, prior to choosing an adaptation scheme.

5.2 BUFFER MANAGEMENT

The client features a buffer to prevent stalls during video playback. At the beginning of a streaming session, the system populates the buffer with the lowest quality representation. Although this process increases the time between initiating and starting the stream, it is helpful in preventing stalls.

Moreover, research suggests that provided the user has the intention of watching a video in its entirety, initial loading times up to several seconds are perceived as acceptable for the user [23].

Therefore, populating the buffer with the average quality representation may be considered, to provide a better quality for the first video segments.

During video playback, the system regularly examines the buffer size. If the buffer happens to be smaller than the set minimum buffer size, the system switches the quality for each point cloud object to the lowest representations until the buffer is adequately filled.

5.3 MANIFEST PRESENTATION DESCRIPTION

The MPD contains the relevant information for the streaming session. It is stored on the server and similar to the functions of the client, it can be configured as desired. For instance, each point cloud object may be given a specific position or rotation. After the configuration file is processed, the system starts the MPD creation module.

By defining the directory of the compressed point cloud files, the MPD creation module fetches the information of all point cloud objects, namely the name of the object, the different quality representations, and the number of segments. Next, it searches for the uncompressed point cloud data.

By accessing the raw point cloud data, the system extracts relevant information for the point cloud objects, namely the size of the bounding box. As mentioned in the previous chapter, the existence of raw point cloud data on the server can lead to storage issues. We argue that the raw data can be removed from the server, once the MPD is created.

In the context of this thesis, we present a customized MPD, which was designed for point cloud streaming. An excerpt of the proposed MPD is given in Listing 5.1.

It groups the different point clouds into periods, which represent fragments of the point cloud video. Each period contains a number of point cloud objects, which are stored in adaptation sets.

An adaptation set includes information about the name and duration of an object. It is worth noting that the duration attribute is associated with the framerate of the point cloud video. For instance, a duration of two represents 60 point cloud frames, given a framerate of 30.

Listing 5.1: An example for a MPD, featuring longdress and loot [31]

```
<?xml version='1.0' encoding='UTF-8'?>
<MPD format="bin" type="dynamic" mediaPresentationDuration="1" framerate="30">
  <RepresentationURL>point-clouds/$adaptationset_name$/$representation_id$/
      segment_$period_id$.bin</RepresentationURL>
  <Period id="0">
    <AdaptationSet id="0" name="longdress" duration="1"
                pos="1500 0 0" rot="0 0 0"
                size="103 9 22 459 1012 318">
      <Representation id="1" bandwidth="4511264" />
      <Representation id="2" bandwidth="7704904" />
    </AdaptationSet>
    <AdaptationSet id="1" name="loot" duration="1"
                pos="-1500 0 0" rot="0 0 0"
                size="28 7 119 380 999 473">
      <Representation id="1" bandwidth="2176080" />
      <Representation id="2" bandwidth="3405168" />
    </AdaptationSet>
  </Period>
</MPD>
```

The concept of adaptive bitrate streaming requires multiple quality representations, in order to adapt to bandwidth changes. We use representation elements to store the different quality representations for each point cloud object. Each quality representation contains information about file size, which is given in bits.

While creating the quality representations, it is of advantage to incorporate an adequate amount of quality representations for the purpose of covering a broad range of bandwidth requirements. Additionally, it may prove to be helpful in reducing the visual contrast between representations.

Considering network stability and changes in head movement, a shorter duration for video segments is helpful in supporting frequent switches [4] and thus, ensures reliable adaptation. Accordingly, we choose a duration of 1 second for all point cloud objects.

Lastly, the MPD specifies the position and orientation of each point cloud object. To store the information on the minimum bounding box, we define a size attribute for every adaptation set.

By parsing the content of the MPD, the client maps each period along with the point cloud objects and their representations to the appropriate classes.

In contrast to other proposed MPD for point cloud systems [9, 11], our approach reduces the amount of information to a minimum. For instance, we define a set directory for the point cloud segments and thus, reduce redundant information in the MPD file.

At the same time, we include additional information, such as minimum bounding size. Finally, the proposed MPD combines the strengths of the other solutions, resulting in a more powerful approach.

5.4 EVALUATION METHODS

During each streaming session, the client captures information on the requested point cloud segments along with the bandwidth estimation, given the configuration file is set accordingly. Moreover, it defines important system information, such as streaming duration or buffer size during video playback.

The resulting log files are stored in different directories and ordered chronologically, by assigning a timestamp to the filename.

While the textual representation of the information is helpful for processing activities, it creates problems in terms of readability. In an attempt to address this issue, the system features visualization methods that transform the log files into more understandable diagrams.

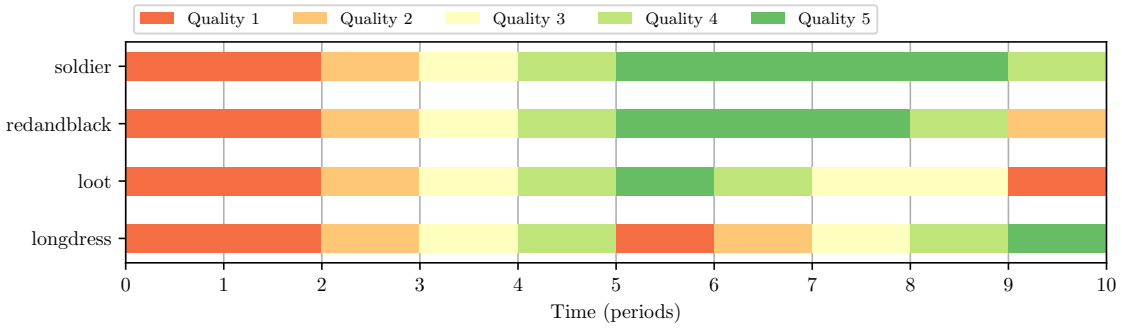

Figure 5.1: Example for a diagram, illustrating the quality of the requested point cloud objects

For an example for the diagram creation, consider Figure 5.1. It represents the quality level for the point cloud objects for each period, giving insight over the performance of the system.

In this particular example, we used a buffer size of 2 seconds. Upon closer inspection, the first two segments for each point cloud object are requested in the lowest quality. As mentioned previously, this mechanism is helpful in preventing stalls.

Moreover, the incremental increase in quality suggests that the uniform bitrate allocation scheme was employed. In the upcoming chapter, we will provide a detailed overview on the performance of both allocation schemes.

5.5 CONFIGURATION

Prior to starting the streaming process, the client fetches starting options from a configuration file. The most important settings are the server address and MPD file name, which are used to communicate with the server.

Moreover, the configuration file is mainly concerned with the system's behavior. It allows

the developer to control the streaming process, namely by defining buffer size and adaptation algorithm.

Regarding the evaluation function of the system, it defines the option to start a simulation mode. In that case, the client uses a predefined bandwidth trace to simulate the time taken for fetching the point cloud segments instead of performing real GET requests to the server. Accordingly, the client divides the segment size by the set bandwidth.

Here, the developer can choose between three bandwidth configurations:

1. Fixed bandwidth trace, simulating a constant bandwidth over time
2. Wired bandwidth trace, simulating more realistic network behavior
3. Wireless bandwidth trace, to investigate the effects of a mobile connection

To generate new bandwidth traces for wired and wireless configurations, the system uses existing bandwidth traces to apply the necessary modifications.

Regardless of the chosen bandwidth configuration, the developer is asked to specify the bandwidth speed in Mbps. While this value is constant for the first bandwidth option, it sets the mean value for the other configurations.

During the execution of the system, the client regularly prints the status of each action. For instance, it may add the estimated bandwidth speed after each request to the console. However, to prevent congestion and guarantee visibility, the printing options are specified in the configuration file.

Similarly, the same idea is applied to the diagram and logging module of the system. The developer may choose to see the results of the system after each simulation or use it solely for practical reasons and disable the function altogether.

5.6 SUMMARY

Ultimately, the proposed system was designed for point cloud streaming and focuses on evaluation. To summarize, we present a basic overview of a typical system session.

By starting the video stream, the client requests the MPD to retrieve the relevant information. Next, it downloads point cloud segments to fill the buffer. After the buffer is adequately filled, the system focuses on maintaining continuous playback.

Simultaneously, it considers the user's bandwidth and applies adaptive bitrate streaming to ensure a good QoE. During each streaming session, the client stores relevant information on the requested point cloud files and the system's behavior.

Finally, the developer is able to review the streaming process with the help of the evaluation modules.

6 EVALUATION

The main goal of the proposed point cloud streaming system is to provide future collaborators with a framework, that facilitates the evaluation and comparison of different methods. Therefore, we conducted a number of simulations to not only demonstrate system behavior under different settings but compare the significance of each design decision.

The following chapter will describe the evaluation process, prior to discussing the results.

6.1 EVALUATION SETUP

The client is powered by a Raspberry Pi 4 Model B with 4GB of RAM, running Ubuntu 22.04 LTS. We use a second Raspberry Pi for the server, using the same configuration. The server is an NGINX HTTP/1.1 web server and is responsible for storing and transmitting the point cloud files.

We used the full-body point clouds from the JPEG Pleno Database [31] to perform the simulations. They feature four dynamic sequences, namely longdress, loot, redandblack and soldier. All objects were captured by 42 RGB cameras over a duration of 10 seconds at 30 frames per second.

In their unprocessed form, the file size for a single frame ranges from 13MB to 24MB, depending on the complexity of the scene. We used MPEG's TMC2 encoder to compress the point cloud files, which uses V-PCC as its underlying technology.

In addition to the point cloud files, we use two different MPD to differentiate between single point cloud scenes and multi point cloud scenes.

While the first MPD features only longdress, the second MPD includes all four point cloud objects. Regardless of the number of objects, both MPD contain exactly 10 periods.

Moreover, each point cloud object has five quality representations, where quality level 5 represents the highest quality possible.

With regard to the wired bandwidth trace, we used Wireshark[1] to record the network activity over a period of 10 minutes. Concerning the wireless bandwidth trace, we used a predefined 4G/LTE trace from [8]. An illustration of the two bandwidth traces is given in Figure 6.1.

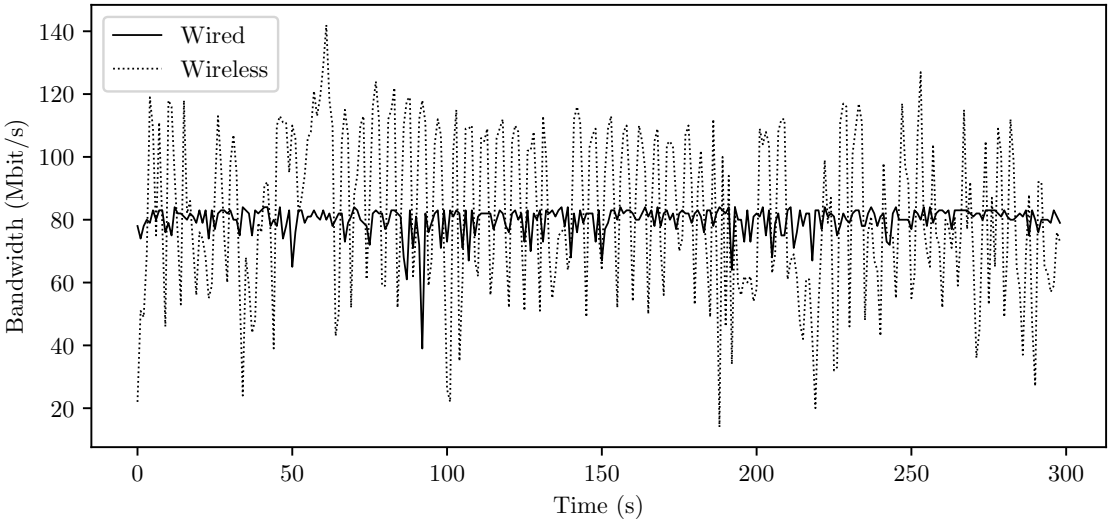

Figure 6.1: Wired and wireless bandwidth traces with a mean bandwidth of 80 Mbps

6.2 PROOF OF CONCEPT

This section will demonstrate the working system while providing a better understanding on the importance of user bandwidth. Starting with Figure 6.2, we focus on system behavior by examining the average quality of one point cloud object, based on connectivity type. We set the adaptation algorithm to dash basic and define a buffer size of zero, to focus solely on the quality of the video stream.

To examine the effects of the connectivity type, we used the three bandwidth configuration options that were presented in the previous chapter. By creating multiple bandwidth traces

1 https://www.wireshark.org/

with different mean values, we are able to study the effects on the average quality.

The average quality is determined by first dividing the sum of the quality levels by the total number of periods. First, consider the course of the fixed and wired graph.

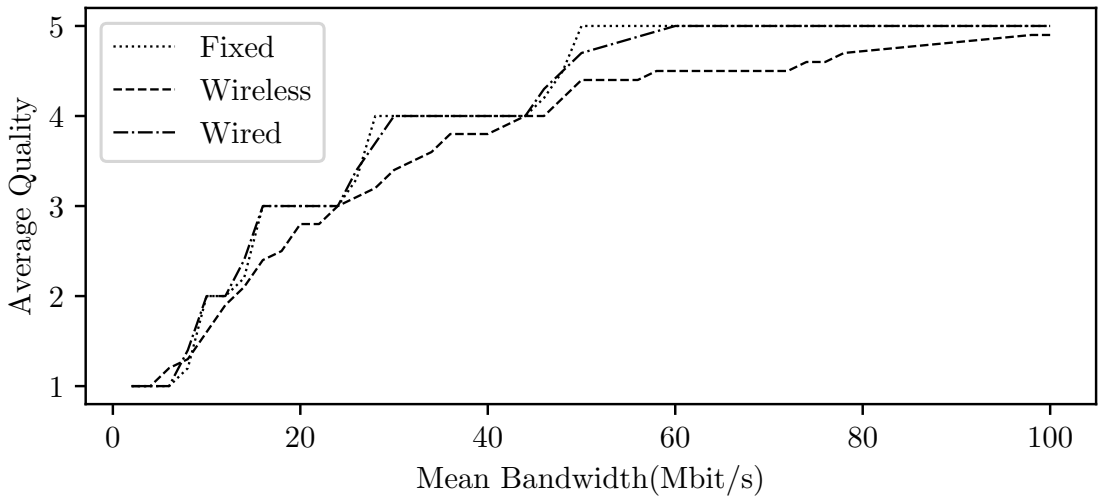

Figure 6.2: Average quality of longdress under different bandwidth requirements with three bandwidth configurations

Naturally, the wired configuration is less prone to bandwidth fluctuations and thus, similar to a constant constant function.
Although the fixed graph generally results in a higher quality representation, both wired and wireless connection types can produce better quality, depending on the degree of variation. For instance, the wireless configuration with a mean bandwidth of 6 Mbps produces slightly higher quality than the rest.

Upon closer inspection, the constant bandwidth of 50 Mbps is sufficient to stream longdress at the highest quality level consistently. The reason for this value is the fourth segment of longdress, which requires the highest amount of bandwidth.
Considering its bandwidth requirement of 49.6 Mbit, the stream upholds the best quality possible, if the estimated user bandwidth exceeds this threshold when requesting the point cloud files.

Regarding the variable graphs, the bandwidth could be lower than this threshold, re-

sulting in a lower average quality. Consequently, both wired and wireless bandwidth configurations need slightly higher bandwidth to accommodate for their bandwidth fluctuations.

While all three graphs follow a similar course at lower values for the mean bandwidth, they tend to differentiate themselves more strongly for higher values. For instance, the wireless configuration generally requires more bandwidth to reach the next quality level.

While the difference for the first two levels is small, it starts to increase rapidly. To reach the fifth quality level, the wireless configuration demands almost twice the mean bandwidth of the fixed and wired configurations.

This can be attributed to the high fluctuations of the wireless bandwidth trace. To illustrate, the first segment is requested at about 25% of the mean bandwidth. As a result, the mean bandwidth has to be increased significantly, to increase the value for this first entry. For reference, consider the wireless trace in Figure 6.1.

Next, we used the second MPD to focus on multiple point cloud objects. Figure 6.3 demonstrates the average quality for the four point cloud objects. Seemingly, the second Figure shows barely any noticeable difference from the previous Figure. Here, the graphs for the different bandwidth configuration options share a similar shape.

However, the bandwidth requirements almost triple for the multi point cloud scene. While a user bandwidth of 50 Mbps was sufficient to stream one point cloud object at the highest quality, it reaches 140 Mbps, given the wired bandwidth configuration.

Ultimately, the two figures illustrate the functionality of the proposed system. By defining different values for the mean bandwidth, different user groups are covered in terms of bandwidth speed. We notice high bandwidth requirements for multi-object scenes, especially in wireless networks.

Next, we will examine the performance of two different bitrate allocation schemes prior to discussing the impacts on QoE.

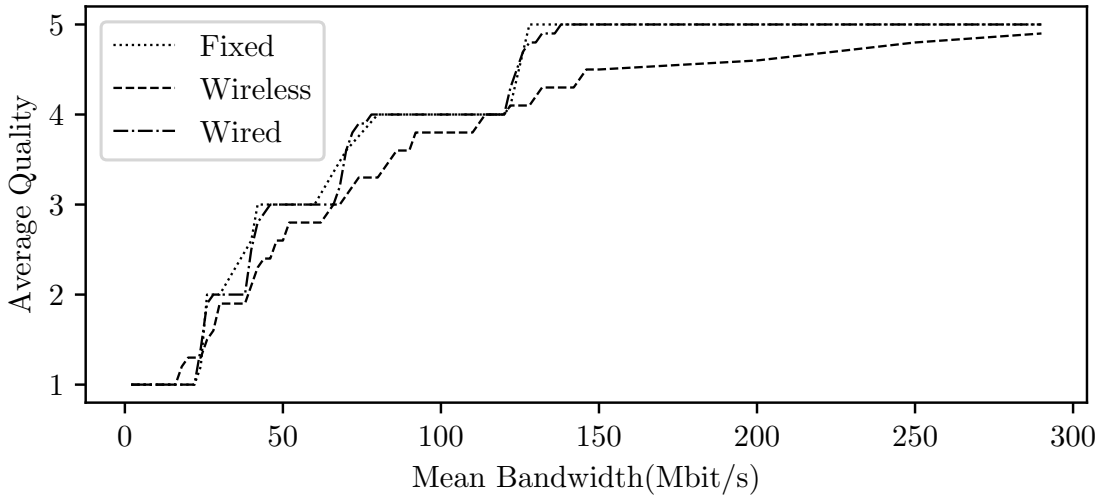

Figure 6.3: Average quality of the four point cloud objects longdress, loot, redandblack and soldier [31] under different bandwidth requirements with three bandwidth configurations

6.3 EVALUATION RESULTS

In the previous section, we used a basic configuration to run the simulations. In the following, we will compare the performance of the two bitrate allocation schemes, prior to discussing the effects of bandwidth speeds on stall occurance and duration.

6.3.1 *Bitrate Allocation Schemes*

For the next evaluation step, we compare the greedy bitrate allocation scheme with the uniform bitrate allocation scheme. Accordingly, the distance from the point cloud objects to the user is used to determine the importance of each object.

We use the scene defined in second MPD file, featuring four point cloud objects that are positioned on a straight line. The placement of the objects along with the camera trace is taken from the second scene proposed in [9]. However, we modified the position of the objects and reduced the distance between each object to accommodate for the shorter duration of the MPD file.

Due to the placement of the objects and the moving speed of the camera, the soldier

object is more visible to the camera than the other objects, which will be important for the results of the simulations.

We use a buffer size of 2 seconds to simulate more realistic streaming sessions. Moreover, the following two figures will focus on the quality of each point cloud object individually. By doing so, we use the wired bandwidth configuration to simulate an optimized environment for the system.

Figure 6.4 features the greedy bitrate allocation scheme, presenting the quality of each object under different bandwidth speeds. Due to the buffer size of 2, the first two segments are requested in the lowest quality. Therefore, the highest point cloud quality can contain only 8 segments with a quality level of 5, resulting in a maximum average quality of 4.2.

At the same time, the maximum average quality is 3.6 for the uniform bitrate allocation scheme, illustrated in Figure 6.5. The difference is caused by the incremental increase in quality. As the first two segments are requested by the first quality level, the system reaches the fifth quality level after four periods. As a result, the change in video quality ensues gradually, at the expense of a lower average quality.

Moving on, consider the beginning of the graphs illustrated in Figure 6.4 and 6.5. In this particular section, both bandwidth schemes feature a higher average quality for the loot point cloud object. This can be attributed to the bandwidth requirement, as loot requires the lowest amount of bandwidth out of all point cloud objects. Consequently, it has the highest average quality for lower values of mean bandwidth, irrespective of the distance to the user.

To illustrate, another point cloud object may be given higher importance but the available bandwidth may not be sufficient to request a higher quality representation. As a result, the next object is taken into consideration and may require less bandwidth for an increasement in quality and thus, result in a higher average quality.

As mentioned, this effect is exclusive to low bandwidth speeds. For higher values for the mean bandwidth, the differences in file size become less of an issue. As a result, the adaptation algorithm focuses on the ranking of the point cloud objects and thus, allocates more bandwidth to the objects close to the user.

At the same time, the soldier point cloud object generally has a higher quality compared to the other objects, due to being exposed more frequently to the camera. In contrast, the other point cloud objects share a similar duration in front of the camera.

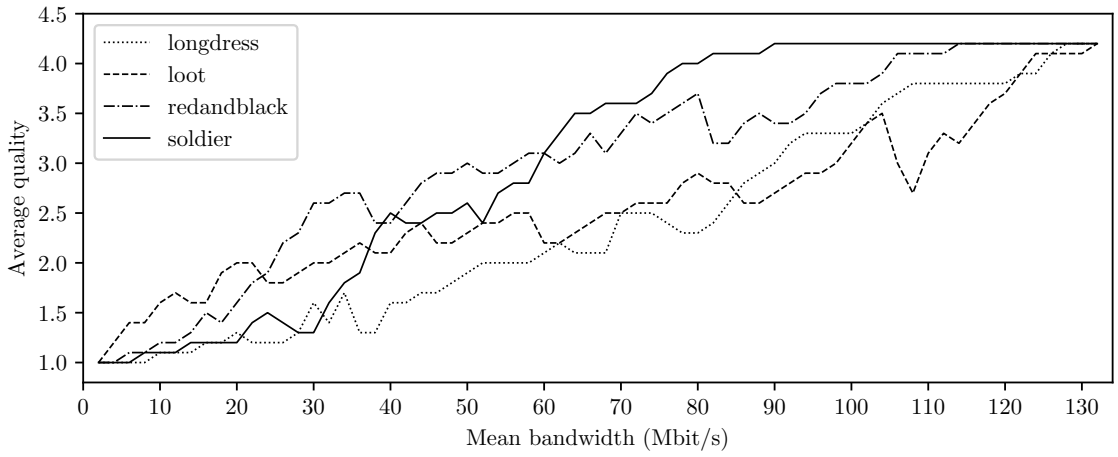

Figure 6.4: Distance-based adaptation algorithm, using the **greedy** bitrate allocation scheme

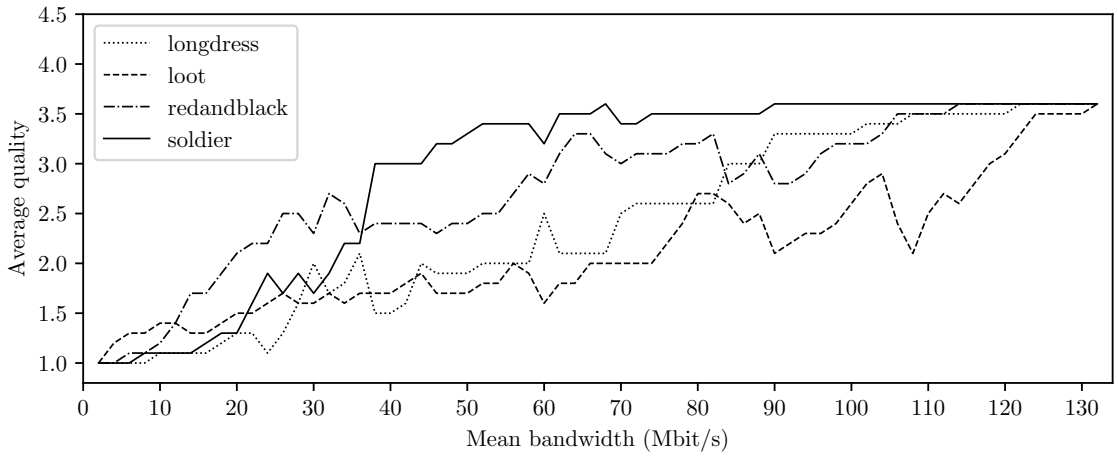

Figure 6.5: Distance-based adaptation algorithm, using the **uniform** bitrate allocation scheme

Due to the nature of the greedy bitrate allocation scheme, the change in quality between the objects is generally stark. Although the increasement of the average quality of all objects is fairly similar for bandwidth speeds up to 53 Mbps, the difference between objects tends to increase with higher speeds.

In contrast, the graphs for the average quality using the uniform bitrate allocation scheme have a more stable course. For bandwidth speeds between 35 Mbps and 60 Mbps, the quality of the objects changes homogeneously.

6.3.2 *Stalls and Stall Duration*

The remainder of this section will concentrate on the QoE. As mentioned, stalls and rebuffering are perceived as negative and decrease the satisfaction of the user [23]. To examine the occurrences of these issues, we measured the number of stalls and the average stall duration under different point cloud scenes.

For this evaluation step, we use a fixed bandwidth configuration to focus on the bandwidth requirements for continuous video playback. Similar to the previous simulation, we define a buffer size of 2 seconds. Figure 6.6 presents the findings of the evaluation.

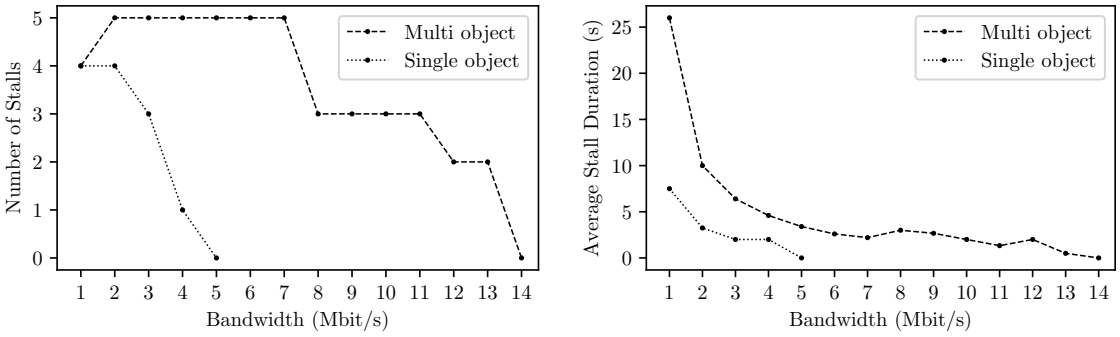

Figure 6.6: Number of stalls and their average duration for single and multi point cloud scenes under different bandwidth speeds

As expected, the number of stalls is smaller for single object scenes. Interestingly, the number of stalls declines rapidly: Due to the low bandwidth requirements, a mean bandwidth speed of 5 Mbps is sufficient to ensure continuous playback.

With regard to multi object scenes, there is little change for bandwidth speeds up to 7 Mbps on the number of stalls. However, the average stall duration is strongly impacted. Starting with over 10 seconds, the average duration stall reaches 3 seconds for bandwidth

speeds at 6 - 12 Mbps. For higher speeds, there is little improvement for the average stall duration, despite the decrease in occurrences.

Finally, a mean bandwidth speed of 14 Mbps results in a stall-free experience. Similar to the findings in (subsection), a multi object scene with four point clouds requires about three times the bandwidth compared to single objects scenes.

6.4 ANALYSIS OF RESULTS

In this chapter, we used the proposed point cloud system to determine bandwidth requirements for different point cloud scenes and study the influence of various design decisions, such as connection type and bitrate allocation scheme.

As indicated, point clouds require great bandwidth speeds. While the bandwidth requirements for single object scenes is rather low, the mean bandwidth speed has to be thrice as great to support multiple point clouds. To stream four objects simultaneously, an average speed of 140 Mbps is required. To illustrate, the global average bandwidth speed for wired networks is expected to reach 110 Mbps by 2023 [3].

In addition, the results suggest that a wireless connection is more prone to low quality video, requiring a much higher value for mean bandwidth to compensate for network fluctuations. In contrast, a wired configuration results in a higher quality for point clouds, irrespective of the number of objects.

With regard to HMD and other mobile devices, this issue could be detrimental. It would require the user to operate the system through a wired connection, which could reduce the overall QoE.

We argue that this issue is partly related to the throughput estimation module of the system. It disregards the relationship between the different bandwidth estimations and consequently, is susceptible to bandwidth changes. An example of a more robust estimation could be the average value for the bandwidth estimation.

Regarding the bitrate allocation schemes, we find that the greedy implementation results in a higher average quality, while the uniform allocation scheme produces a more

balanced quality representation. According to Seufert *et al.* [23], the QoE is greatly affected by the number of quality changes and their duration. Moreover, the authors argue that quality switches reduce the perceived quality of a video service and their extent is increased if the switches happen abruptly.

Ultimately, a point cloud streaming system requires a balance between streaming quality and QoE.

7 CONCLUSIONS

The last chapter will summarize the main findings of the thesis and state its contributions. In addition, it demonstrates the limitations of the proposed system and presents future work.

7.1 SUMMARY

This thesis addressed the idea of point clouds and examined the challenges involved in transmitting them in real-time.

While point clouds are commonly used to represent three-dimensional data, their practical use is greatly limited by storage and bandwidth constraints. Accordingly, recent attempts have been made on the compression of point clouds, resulting in a significant decrease in file size.

At the same time, the streaming of point cloud content is considered. Similar to traditional video, a streaming system has to take the user's network condition into consideration. Due to limited bandwidth capacities and fluctuating bandwidth speeds, the idea of adaptive bitrate streaming is applied to point cloud streaming.

Adaptive bitrate streaming adjusts the quality of the video to the bandwidth of the user during video playback. As a result, the user can experience the best visual quality possible, without experiencing issues such as stalls or rebuffering.

In recent years, researchers have frequently used DASH to apply adaptive bitrate streaming to point clouds. Although there are a number of adaptive bitrate streaming solutions, DASH differentiates itself from other implementations by being an international standard.

Based on these findings, the second part of the thesis presented an approach for an extendable point cloud streaming system. In addition to streaming point cloud data, the

proposed system offers several evaluation methods that enable the assessment and comparison of point cloud streaming methods.

In the evaluation chapter, we use the functions of the system to compare different design decisions. The results suggest that high bandwidth speeds result in higher average quality, especially if multiple point cloud objects are considered.
Moreover, we found that a wireless connection requires much higher bandwidth speeds compared to a wired configuration, to compensate for bandwidth fluctuations.

With regard to the bitrate allocation heuristics, the interplay of performance and QoE has to be considered. While the greedy allocation scheme results in higher video quality, the uniform scheme produces a more gradual change between quality representations.

7.2 CONTRIBUTIONS

The main contribution of the thesis is the proposed system, which serves as a practical framework that supports the development of point cloud streaming solutions. It features several evaluation methods that examine system behavior and performance.

In addition, it considers the involvement of the user by assessing the quality of the stream. By considering the number of quality switches and playback stalls, the system aims at ensuring a proper QoE.

Finally, the thesis provides an overview of the contributions made on point cloud streaming. It identifies shortcomings of existing solutions and provides suggestions for future development.

7.3 LIMITATIONS

The proposed system offers several functions that assist the development of point cloud streaming. However, its current state is limited in its execution, which will be discussed in the following.

First, the throughput estimation module is not able to capture the user's bandwidth speed accurately. As a result, the system may assume a low bandwidth speed and request

lower quality representations, which can negatively influence the QoE. To provide better estimations, the solution presented by Thang *et al.* [27] should be considered.

Second, the scene used in the evaluation chapter does not represent a real user trajectory. The findings of the evaluation may not be applied to real-world scenarios, featuring real human behavior.

Lastly, the state-of-the-art in point cloud compression is unable to make real-time transmission possible, due to the processing power involved in decoding the point cloud files. Consequently, a point cloud streaming solution for contemporary devices has yet to exist.

7.4 FUTURE WORK

As indicated, the findings of this thesis lack subjective evaluations. Therefore, we propose the idea to conduct a number of tests where participants are asked to evaluate the quality of the system. As a result, the proposed concepts of QoE are tested in real scenarios and help to make further adjustments according to the user's needs.

Furthermore, the system may be extended with movement and viewport prediction modules, increasing the amount of bandwidth savings.

BIBLIOGRAPHY

[1] "Call for Proposals for Point Cloud Compression V2." In: *document ISO/IEC JTC1/SC29 /WG11 MPEG, N 16763, 3D Graphics* (2017). URL: https://mpeg.chiariglione.org/standards/mpeg-i/point-cloud-compression/call-proposals-point-cloud-compression-v2 (visited on 09/15/2022).

[2] Chao Cao, Marius Preda, and Titus Zaharia. "3D Point Cloud Compression: A Survey." In: *The 24th International Conference on 3D Web Technology*. Web3D '19. New York, NY, USA: Association for Computing Machinery, July 2019, pp. 1–9. ISBN: 978-1-4503-6798-1. DOI: 10.1145/3329714.3338130. URL: https://doi.org/10.1145/3329714.3338130 (visited on 08/23/2022).

[3] "Cisco Annual Internet Report (2018–2023) White Paper." en. In: *Cisco: San Jose, CA, USA* (). URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (visited on 09/16/2022).

[4] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. "Viewport-adaptive navigable 360-degree video delivery." In: *2017 IEEE International Conference on Communications (ICC)*. ISSN: 1938-1883. May 2017, pp. 1–7. DOI: 10.1109/ICC.2017.7996611.

[5] Li Cui, Rufael Mekuria, Marius Preda, and Euee S. Jang. "Point-Cloud Compression: Moving Picture Experts Group's New Standard in 2020." In: *IEEE Consumer Electronics Magazine* 8.4 (July 2019), pp. 17–21. ISSN: 2162-2248, 2162-2256. DOI: 10.1109/MCE.2019.2905483. URL: https://ieeexplore.ieee.org/document/8732728/ (visited on 08/15/2022).

[6] Dapeng Wu, Y.T. Hou, Wenwu Zhu, Ya-Qin Zhang, and J.M. Peha. "Streaming video over the Internet: approaches and directions." In: *IEEE Transactions on Circuits and Systems for Video Technology* 11.3 (Mar. 2001), pp. 282–300. ISSN: 10518215. DOI: 10.1109/76.911156. URL: http://ieeexplore.ieee.org/document/911156/ (visited on 08/15/2022).

[7] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. "An overview of ongoing point cloud compression standardization activities: video-based

(V-PCC) and geometry-based (G-PCC).” en. In: *APSIPA Transactions on Signal and Information Processing* 9.1 (2020). ISSN: 2048-7703, 2048-7703. DOI: 10.1017/ATSIP.2020.12. URL: http://www.nowpublishers.com/article/Details/SIP-149 (visited on 08/15/2022).

[8] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. “HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks.” In: *IEEE Communications Letters* 20.11 (2016). Publisher: IEEE, pp. 2177–2180.

[9] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. “Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression.” en. In: *Proceedings of the 27th ACM International Conference on Multimedia*. Nice France: ACM, Oct. 2019, pp. 2405–2413. ISBN: 978-1-4503-6889-6. DOI: 10.1145/3343031.3350917. URL: https://dl.acm.org/doi/10.1145/3343031.3350917 (visited on 08/15/2022).

[10] Mohammad Hosseini and Viswanathan Swaminathan. “Adaptive 360 VR Video Streaming: Divide and Conquer.” In: *2016 IEEE International Symposium on Multimedia (ISM)*. San Jose, CA, USA: IEEE, Dec. 2016, pp. 107–110. ISBN: 978-1-5090-4571-6. DOI: 10.1109/ISM.2016.0028. URL: http://ieeexplore.ieee.org/document/7823595/ (visited on 08/15/2022).

[11] Mohammad Hosseini and Christian Timmerer. “Dynamic Adaptive Point Cloud Streaming.” en. In: *Proceedings of the 23rd Packet Video Workshop*. Amsterdam Netherlands: ACM, June 2018, pp. 25–30. ISBN: 978-1-4503-5773-9. DOI: 10.1145/3210424.3210429. URL: https://dl.acm.org/doi/10.1145/3210424.3210429 (visited on 08/15/2022).

[12] Yan Huang, Jingliang Peng, C.-C. Jay Kuo, and M. Gopi. “Octree-based progressive geometry coding of point clouds.” In: *Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics*. SPBG’06. Goslar, DEU: Eurographics Association, July 2006, pp. 103–110. ISBN: 978-3-905673-32-6. URL: https://dl.acm.org/doi/10.5555/2386388.2386403 (visited on 09/15/2022).

[13] Baochun Li, Zhi Wang, Jiangchuan Liu, and Wenwu Zhu. “Two decades of internet video streaming: A retrospective view.” en. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 9.1s (Oct. 2013), pp. 1–20. ISSN: 1551-6857, 1551-6865. DOI: 10.1145/2505805. URL: https://dl.acm.org/doi/10.1145/2505805 (visited on 08/15/2022).

[14] Hao Liu, Hui Yuan, Qi Liu, Junhui Hou, and Ju Liu. "A comprehensive study and comparison of core technologies for MPEG 3-D point cloud compression." In: *IEEE Transactions on Broadcasting* 66.3 (2019). Publisher: IEEE, pp. 701–717. ISSN: 0018-9316.

[15] Zhi Liu, Qiyue Li, Xianfu Chen, Celimuge Wu, Susumu Ishihara, Jie Li, and Yusheng Ji. "Point Cloud Video Streaming: Challenges and Solutions." In: *IEEE Network* 35.5 (Sept. 2021), pp. 202–209. ISSN: 0890-8044, 1558-156X. DOI: 10.1109/MNET.101.2000364. URL: https://ieeexplore.ieee.org/document/9537928/ (visited on 08/15/2022).

[16] Ricky K. P. Mok, Edmond W. W. Chan, and Rocky K. C. Chang. "Measuring the quality of experience of HTTP video streaming." In: *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. ISSN: 1573-0077. May 2011, pp. 485–492. DOI: 10.1109/INM.2011.5990550.

[17] Carlos Moreno, Yilin Chen, and Ming Li. "A dynamic compression technique for streaming kinect-based point cloud data." In: IEEE, 2017, pp. 550–555. ISBN: 1-5090-4588-0. URL: https://ieeexplore.ieee.org/document/7876188.

[18] *Netflix System Requirements*. URL: https://help.netflix.com/en/node/13444 (visited on 09/15/2022).

[19] Ozgur Oyman and Sarabjot Singh. "Quality of experience for HTTP adaptive streaming services." In: *IEEE Communications Magazine* 50.4 (Apr. 2012). Conference Name: IEEE Communications Magazine, pp. 20–27. ISSN: 1558-1896. DOI: 10.1109/MCOM.2012.6178830.

[20] Feng Qian, Bo Han, Jarrell Pair, and Vijay Gopalakrishnan. "Toward Practical Volumetric Video Streaming on Commodity Smartphones." In: *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. HotMobile '19. New York, NY, USA: Association for Computing Machinery, Feb. 2019, pp. 135–140. ISBN: 978-1-4503-6273-3. DOI: 10.1145/3301293.3302358. URL: https://doi.org/10.1145/3301293.3302358 (visited on 08/25/2022).

[21] Ruwen Schnabel and Reinhard Klein. "Octree-based point-cloud compression." In: *Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics*. SPBG'06. Goslar, DEU: Eurographics Association, July 2006, pp. 111–121. ISBN: 978-3-905673-32-6. URL: https://dl.acm.org/doi/10.5555/2386388.2386404 (visited on 08/26/2022).

[22] Sebastian Schwarz et al. "Emerging MPEG Standards for Point Cloud Compression." In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.1 (Mar. 2019). Conference Name: IEEE Journal on Emerging and Selected Topics in Circuits and Systems, pp. 133–148. ISSN: 2156-3365. DOI: 10.1109/JETCAS.2018.2885981.

[23] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hobfeld, and Phuoc Tran-Gia. "A Survey on Quality of Experience of HTTP Adaptive Streaming." In: *IEEE Communications Surveys & Tutorials* 17.1 (2015), pp. 469–492. ISSN: 1553-877X, 2373-745X. DOI: 10.1109/COMST.2014.2360940. URL: https://ieeexplore.ieee.org/document/6913491/ (visited on 08/15/2022).

[24] Iraj Sodagar. "The MPEG-DASH Standard for Multimedia Streaming Over the Internet." In: *IEEE Multimedia* 18.4 (Apr. 2011), pp. 62–67. ISSN: 1070-986X. DOI: 10.1109/MMUL.2011.71. URL: http://ieeexplore.ieee.org/document/6077864/ (visited on 08/15/2022).

[25] Thomas Stockhammer. "Dynamic adaptive streaming over HTTP –: standards and design principles." In: *Proceedings of the second annual ACM conference on Multimedia systems*. MMSys '11. New York, NY, USA: Association for Computing Machinery, Feb. 2011, pp. 133–144. ISBN: 978-1-4503-0518-1. DOI: 10.1145/1943552.1943572. URL: https://doi.org/10.1145/1943552.1943572 (visited on 05/20/2022).

[26] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. "User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling." en. In: *Proceedings of the 28th ACM International Conference on Multimedia*. Seattle WA USA: ACM, Oct. 2020, pp. 3669–3677. ISBN: 978-1-4503-7988-5. DOI: 10.1145/3394171.3413535. URL: https://dl.acm.org/doi/10.1145/3394171.3413535 (visited on 05/30/2022).

[27] Truong Thang, Quang-Dung Ho, Jung Kang, and Anh Pham. "Adaptive streaming of audiovisual content using MPEG DASH." In: *IEEE Transactions on Consumer Electronics* 58.1 (Feb. 2012), pp. 78–85. ISSN: 0098-3063. DOI: 10.1109/TCE.2012.6170058. URL: http://ieeexplore.ieee.org/document/6170058/ (visited on 08/15/2022).

[28] Lisha Wang, Chenglin Li, Wenrui Dai, Shaohui Li, Junni Zou, and Hongkai Xiong. "QoE-Driven Adaptive Streaming for Point Clouds." In: *IEEE Transactions on Multimedia* (2022), pp. 1–1. ISSN: 1520-9210, 1941-0077. DOI: 10.1109/TMM.2022.3148585. URL: https://ieeexplore.ieee.org/document/9706255/ (visited on 08/15/2022).

[29] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. "A Survey on Adaptive 360° Video Streaming: Solutions, Challenges and Opportunities." In: *IEEE Communications Surveys & Tutorials* 22.4 (2020), pp. 2801–2838. ISSN: 1553-877X, 2373-745X. DOI: 10.1109/COMST.2020.3006999. URL: https://ieeexplore.ieee.org/document/9133103/ (visited on 08/15/2022).

[30] *Youtube System Requirements*. URL: https://support.google.com/youtube/answer/78358?hl=en (visited on 09/15/2022).

[31] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. "8i voxelized full bodies-a voxelized point cloud dataset." In: *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006* 7.8 (2017), p. 11. URL: http://plenodb.jpeg.org/pc/8ilabs/.

EIDESSTATTLICHE ERKLÄRUNG

Ich, *Özgür Erol (3097359)*,

versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe.

Ich versichere an Eides Statt, dass ich die vorgenannten Angaben nach bestem Wissen und Gewissen gemacht habe und dass die Angaben der Wahrheit entsprechen und ich nichts verschwiegen habe.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 StGB bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 161 Abs.1 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

Essen, 16. September 2022

Özgür Erol

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Final Version as of September 16, 2022 (classicthesis v4.6).