ENHANCEMENT OF 360-VIDEO EXPERIENCE DURING HEAD MOTION

A Thesis

by

Burak Kara

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master's Degree

in the
Department of Computer Science

Özyeğin University
August 2022

Copyright © 2022 by Burak Kara

ENHANCEMENT OF 360-VIDEO EXPERIENCE DURING HEAD MOTION

Approved by:

Assoc. Prof. Ali Cengiz Beğen, Advisor
Dept. of Computer Science
Özyeğin University

Prof. Reha Civanlar
Dept. of Computer Science
Özyeğin University

Assoc. Prof.ı Müge Sayıt
Dept. of International Computer Institute
Ege University

Date Approved: 11 August 2022

To my precious family and friends...

ABSTRACT

Efficient use of available bandwidth is vital when streaming 360-degree videos, as users rarely have enough bandwidth for a pleasant experience. The combination of viewport-dependent streaming (VDS) and rate adaptation is a promising solution to efficient bandwidth use. In VDS, the whole spherical video can be partitioned into several tiles encoded at various quality levels. The goal of this combination is to spend most of the available bandwidth on the viewport tiles. However, the direction and amount of the head motion can change the viewport tiles, resulting in low-quality content rendering until the new tiles can be replaced with high-quality versions. Moreover, head motions can influence the perception of the content to a human eye, where users can experience unfocused (blurry) content on fast head motions. First, the concept of Head-motion-aware Viewport Margins (HMAVM) was developed as a head motion-dependent safety area around the viewport and evaluated using a human head motions dataset. Later, this thesis studies how moving the head may impact the objective viewport quality assessment at short, medium and long timescales.

ÖZETÇE

Kullanıcılar tatmin edici bir 360 derece video deneyimi için nadiren yeterli bant genişliğine sahip olduklarından, 360 derecelik videolar yayınlanırken mevcut bant genişliğinin verimli kullanımı hayati önem taşır. Kullanıcıların görüntü alanına bağlı akış ve uyarlanabilen video kalitesi birleşimi, verimli bant genişliği kullanımı için umut verici bir çözümdür. Görüntü alanına bağlı akışta, tüm küresel video, çeşitli kalite seviyelerinde kodlanmış birkaç parçaya bölünebilir. Bu kombinasyonun amacı, mevcut bant genişliğinin çoğunu görünüm alanı parçalarına harcamaktır. Bununla birlikte, baş hareketinin yönü ve miktarı, görüntü alanı parçalarını değiştirebilir, bu da yeni parçalar yüksek kaliteli versiyonlarıyla değiştirilene kadar düşük kaliteli içerik görmek ile sonuçlanır. Ayrıca, kullanıcıların hızlı baş hareketlerinde odaklanmamış (bulanık) içerik yaşayabilmesi gibi, baş hareketleri içeriğin insan gözü tarafından nasıl algılandığını da etkileyebilir. İlk olarak, Baş Hareketine Duyarlı Görüntü Alanı Kenar Boşlukları (HMAVM), görüntü alanı çevresinde baş hareketine bağlı bir güvenlik alanı olarak geliştirildi ve baş hareketlerinin etkisini en aza indirmek için gerçek insan baş hareketlerinden oluşan veri seti kullanılarak değerlendirildi. Daha sonra, bu tez, başı hareket ettirmenin kısa, orta ve uzun zaman ölçeklerinde objektif bakış açıdan kalite değerlendirmesini nasıl etkileyebileceğini inceledi.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the thesis committee members for their support and feedback. Special thanks to my supervisor, Assoc. Prof. Ali Cengiz Beğen, for his outstanding support, patience and guidance throughout this study, where he generously provided his knowledge and expertise. I am also grateful to my friend and colleague, Mehmet N. Akçay, for his support and valuable contributions. Another special thanks to Saba Ahsan, Igor D.D. Curcio and Emre Aksu from Nokia for their guidance and contributions.

To conclude, I cannot forget to thank my family and friends for all their unconditional support. I could not have undertaken this journey without them.

The author gratefully acknowledges the support of TÜBİTAK 2210/A National MSc/MA Scholarship Program for this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND RELATED WORK

The demand for immersive video has been on the rise thanks to the increasing popularity of head-mounted-display (HMD) devices since they have been more accessible for end-users. Today, users can experience immersive videos on social media and online gaming platforms. The common goal across all these platforms is to ensure a pleasant user experience, albeit with resources (mainly the available bandwidth during streaming) less than ideal.

Many standards and algorithms have been developed to deal with the demand for high resources. Omnidirectional MediA Format (OMAF) is the first worldwide virtual reality (VR) standard designed to store and distribute immersive media [1]. Dynamic Adaptive Streaming over HTTP (DASH) (aka MPEG-DASH) enables the best effort (the highest quality possible) media content streaming over Hypertext Transfer Protocol (HTTP) thanks to the rate adaptation [2]. DASH used with OMAF created packages also allows the streaming of immersive content. The one bottleneck is that immersive media streaming is more challenging than traditional (2D) content streaming. These challenges require more considerations to deal with. As mentioned, the bandwidth consumption is much higher in the streaming of immersive media compared the 2D content streaming. The 4K content is the bare minimum spatial resolution for a pleasant immersive experience since HMD devices allow watching immersive content much closer to the user's eyes.

Viewport-dependent streaming (VDS) using tiles is one of the best approaches we have today to deal with immersive media streaming challenges; it delivers the viewport (part of the video a user is viewing) at the highest possible quality and everything

else at a lower quality. Here, tiles refer to the parts produced by partitioning the immersive video. Tiles are individually encoded, segmented and stored on the server to allow the streaming client to fetch them in a rate-adaptive manner. On the average, bandwidth consumption can be reduced by 30-70% without significantly degrading viewport quality [3, 4, 5, 6, 7, 8, 9, 10]. The design-related problem in VDS arises when the viewport tiles change due to a head motion. The user can briefly experience low-quality tiles until new viewport tiles are downloaded and rendered in high quality. One solution is that the models designed to predict the future head motions can be used to download these low quality tiles. They can reduce the delays occurring during viewport tile updates [11, 12]. Similarly, other learning-based studies to predict future viewports (and tile bitrates) are presented in [13, 14, 15]. Another problem is the viewing possibilities of high quality (viewport) tiles that are not viewed by the user during some particular head motions (*e.g.*, continuous motions from one side to another side, at a faster speed). This data waste can be prevented up to 25% with accurate head trajectory predictions [16].

Ultimately, all these studies are conducted to improve a user's perceptual quality, which is defined as a user's degree of satisfaction from viewing a video. To date, various subjective and objective metrics have been defined to measure perceptual quality. Subjective metrics evaluate the quality with user opinions gathered via question and answer (Q&A) sessions, whereas objective metrics often use mathematical formulations to produce concrete results without depending on any user study. For instance, the average viewport quality is often used to quantify the quality of an immersive video, and it can be specified using a subjective or an objective metric. Another example of an objective metric is the motion to high quality delay (MTHQD), which represents the time spent on a viewport with at least one low-quality tile [17]. Subjective testing involves one or more user surveys, and potentially the collection of

physiological and objective data such as brain wave and head-eye coordination analyses. However, these tests require significantly more effort, time and budget. Moreover, they cannot be conducted in real time and scale to every user, instead only a small subset of the users can be tested. On the other hand, objective metrics are easier to compute and use for comparison.

Earlier, traditional video quality metrics (*e.g.*, peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), video multimethod assessment fusion (VMAF) and video quality metric (VQM)) were proposed to evaluate immersive and 360-degree videos [18, 19]. Further PSNR-based methods (S-PSNR and WS-PSNR) were also introduced in [20, 21] for the same purpose. In addition, 3rd Generation Partnership Project (3GPP) specified a more straightforward method in [22], where the viewport quality is calculated by multiplying the quality ranking of each tile constituting the viewport by the percentage of the viewport it is covering. A further simpler approach has been to use the quality of the tile at the center of the viewport while ignoring all the other tiles in the viewport [23].

The methods combining objective and subjective evaluations were also studied. For instance, weighted viewport peak signal-to-noise ratio (WVPSNR), taking into account the human visual system (HVS), was presented in [24], where the authors offered different weights to zones calculated by their distance from the central gaze direction and the lens' focal length. These zones were then used to compute WVPSNR. Similarly, multi-metric fusion (MMF) [25] combined objective metrics with the characteristics of HVS. In [23], Hooft *et al.* introduced a probabilistic approach called ProbGaze that used eye gaze statistics taken from [26] to construct a viewport quality metric. The authors first distributed circles in the viewport with a polynomial density function based on the gaze heatmap presented in [26]. Then, they sampled the viewport area with uniformly distributed points on these circles. Later, the quality is calculated with the weighted averages of the quality values at these points.

The methods that best approximated the actual user experience were the ones that took HVS into account [18] and this was confirmed by user surveys [23, 27]. While these methods considered users' gaze and eye lens characteristics, they fell short and largely ignored head motions' speed, direction and duration. This thesis argues that these motion characteristics can significantly affect the perceptual quality under certain circumstances. For instance, the content rendered in the viewport can be unfocused (or even blurry) to a human eye if the head turns too fast (that is, the user cannot focus on the content during a fast head motion). In this case, this individual viewport's quality has only a slight, if any, effect on the overall user experience since the user is naturally fine with even a low-quality video for a brief amount of time and the actual viewport quality does not matter. Another observation is that the specific region the user is paying attention to (region of interest (ROI)) inside the viewport varies depending on the head motion. As discussed in [27], the head motion direction and eye gaze distribution are correlated, meaning that the head and eye gaze move mainly in the same direction. This implies that ROI shifts from the viewport center to the side in the direction of head motion. Consequently, the quality of the tiles in the opposite direction matters less. That is, they have a negligible effect on the overall user experience.

In this thesis, first, the study conducted to improve user experience during immersive or 360-degree video streaming is presented. Second, another work designed to measure user experience more accurately is shown. (i) The first study extends an earlier work, the viewport margins that was first presented in [28] to avoid stalls caused by head motions. The proposed solution by the authors adds additional areas around the viewport area. These additional areas have a middle quality (*i.e.*, lower than the viewport tiles but higher than the background). The main goal is smoother viewport changes during head motions, giving a lower MTHQD [17]. Earlier, the concept of the margin was designed as *symmetric* where boundaries around

the viewport are symmetric. They are preset as 30 degrees to the left and right, and 20 degrees to the top and bottom. As expected, the symmetric viewport margins caused a significant bandwidth overhead. Later, *directional margins* were introduced where these extra margin areas were extended or shrunk based on the head motion (*e.g.*, a 40-degree left and 5-degree right margin upon a head motion toward the left). The bandwidth saving thanks to *directional margins* was observed since the concept shrank the margin area in the opposite direction of the head motion. On the other hand, the *directional margins* concept was not aware of the speed and duration of the head motion. The improved version of this *directional margins* concept (named as ead-motion-aware Viewport Margins (HMAVM)) were introduced in this study [29] by taking into account the head-motion speed and duration. However, a bandwidth overhead has occurred again that can affect the user experience when the available bandwidth becomes insufficient to download the margin tiles.

The one-step improved version of the HMAVM was designed in two stages to improve the user experience further and diminish the bandwidth overhead problem. The first stage is adding two new margins types: Complementary Margin (CM) and Negative Margin (NM). In this work, the number of HMAVM tiles was set as dynamic and referred to as Positive Margin (PM). CM tiles are the ones at the farthest distance from the viewport. These were previously marked as possible PM tiles; however, they could not be used as such because of the lack of available bandwidth. Now, they are used as CM tiles as a consequence of the bandwidth gain resulting from the use of the NM tiles. The NM tiles are the viewport tiles in the opposite direction of the head motion that can be delivered in a lower quality since these tiles are hardly visible during the head motion. This counterbalances the network load caused by the previous PMs. The second stage is making the margin selection part of the rate-adaptation (aka ABR) algorithm. Thanks to this, all quality levels of viewport, margin and background tiles can be controlled depending on the head-motion speed

and direction. These contributions are tested using head motion datasets on both HTTP/1.1 and H2. The average viewport quality using weighted average viewport quality (WA) [22] metric was improved up to 13% while MTHQD was improved up to 36%.

(*ii*) The second study was conducted to measure user experience more accurately. It relies on the effect of the head motion on the user experience. This work tries to answer the question of when and how head motions are significant enough to be considered in the quality evaluation. A head motion's speed, direction and duration may be critical in different ways, impacting an objective quality at short, medium and long timescales. To that effect, a new metric called Head Motion Aware Viewport Quality (HMAVQ) is developed in two steps. In the first step, the qualities of individual viewports are calculated using sampled points that are shifted and weighted based on the head-motion speed and direction. In the second step, the overall viewport quality is calculated with the weighted average of the individual viewport qualities, where the weights were assigned based on the head-motion speed. This comparative study reveals that the existing metrics cannot accurately represent the objective quality when the user moves its head.

While I believe this research findings and conclusions in this article are significant enough, there are still some work to do, especially a large-scale user study to confirm the relation between the HMAVQ metric and ground truth, and determine the most appropriate method for the weight assignments. Chapter 4 details the plans on this subject.

CHAPTER II

HEAD-MOTION-AWARE VIEWPORT MARGINS

In this chapter, Head-motion-aware Viewport Margins(HMAVM) for rate-adaptive streaming of 360-degree video is introduced. When a head motion changes the tiles within the viewport, the streaming client tries to replace the (background) tiles already buffered in low quality with their higher-quality versions. In addition to this base implementation, HMAVM try to determine the future viewport using the head motion's characteristics (*e.g.*, speed and direction). Based on this, possible future viewport tiles are requested as Positive Margin (PM) tiles. Moreover, Negative Margin (NM) tiles are determined within the viewport and downgraded to the background quality. Complementary Margin (CM) tiles are chosen from the unused PM tiles selected by the algorithm but not downloaded in high quality since they exceeded the limit for the maximum PM tiles allowed for the streaming client. The size of HMAVM varies according to the ABR decisions, as will be explained in the following sections. The quality of PM and CM tiles is between the quality of the background and viewport tiles. The quality of the NM tiles is the same as the quality of the background tiles.

The DASH manifest for the tiled video provides the tile bitrates for different quality levels. This increases the decision space compared to 2D videos, as the streaming client can individually pick the quality level for each tile. The approach in this thesis finds an association of a single bitrate value to a single quality level for the full 360-degree video instead of the individual tiles in order to reuse the existing ABR algorithms. This implies that the algorithm finds the best quality level for a 360-degree video experience based on the buffered content, required and reported throughput

levels, margin types, *etc.*

The individual tile qualities for the chosen quality level are already predetermined at the session start. This can be done by adding the bitrate values for the tiles at certain quality levels, assigning higher quality to the viewport tiles and lower quality to the rest. These predetermined tile qualities for different viewport orientations can be overwritten to adapt HMAVM with aggressive switches in the ABR. However, efficient tiling schemes, VBR encoding and the changing number of tiles in the viewport can significantly change the required bitrate for the full 360-degree video at a certain quality level based on the viewport orientation. The calculation details for the required bitrate for different representations can be found in [30].

In brief, the design in this thesis consists of the following steps: (i) calculate the viewport quality based on the bitrate estimation from [30], (ii) calculate the PM tiles for the remaining bitrate based on Algorithm 1 in [29], (iii) calculate the CM and NM tiles, and (iv) use the calculated viewport quality and HMAVM from the previous steps for rate adaptation. Different margin types are illustrated in Figure 1.

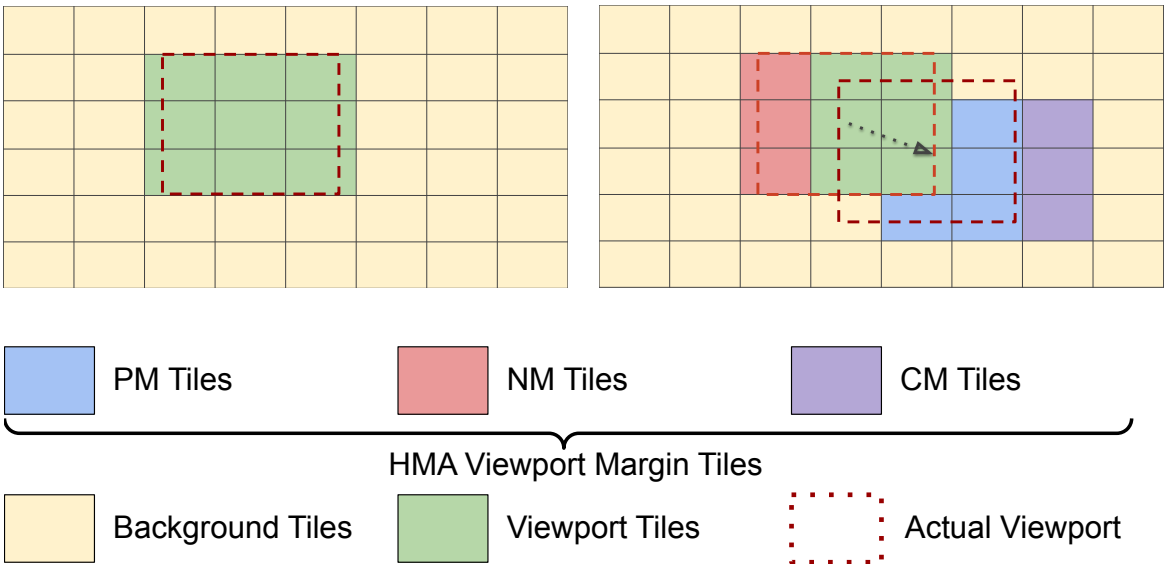


Figure 1: Tiles before a head motion (left). HMAVM after a head motion illustrated with an arrow (right).

Next, first, the rate-adaptive approach in this work's design is explained and then the improvements on PM are presented. Later, the implementation details are described for the CM and NM, followed by the reasons for migrating from HTTP/1.1 [31] to H2 [32] and the details of the H2 implementation.

2.1 Rate-Adaptive Streaming

In [30], a bitrate estimation algorithm was introduced. The authors used the viewport decision technique to select the viewport quality. They also modified the BOLA algorithm [33] to check whether the bitrate requirement could be met with the available bandwidth. In this work, while deciding on the quality of the viewport tiles, HMAVM are taken into account. Once the representative viewport has been selected, the streaming client can determine the required rate for downloading the tiles in all available qualities. This step reduces the decision space for the ABR algorithm from individual tiles to the entire sphere.

The most basic viewport-dependent scheme uses high-quality tiles in the viewport and low-quality tiles in the background. Two groups are thus defined: a group of the tiles within the viewport and another group of the remaining tiles. The tiles in the background are not visible to the user unless the user turns its head, so they are downloaded in the lowest possible quality. This maintains continuity in the viewing experience with the maximum bitrate savings. The tiles in the viewport can be at different qualities depending on the available bandwidth. Therefore, when calculating the full sphere (FS) bitrates, the quality of the background tiles are kept constant (lowest level) and set the quality of the viewport and HMAVM tiles at different levels.

2.2 Positive Margins

The authors of [29] proposed an algorithm to find and prioritize the possible margin tiles using the head motions. This algorithm and the bitrate estimation algorithm from [30] are combined to select the quality of the margin tiles inside ABR. The

resulting algorithm is given in Algorithm 1 that takes the possible margin tiles as an argument (*i.e.*, $possibleMargins$).

Algorithm 1 Get Positive Margins

```
1: function GetPositiveMargins(possibleMargins)
2:     minBuffer ← 3 × segment_duration
3:     maxPercent ← 30%
4:     total_percent ← 0
5:     positiveMargins ← ∅
6:     q ← currentBitrateIndex
7:     q′ ← previousBitrateIndex
8:     r = getDashThroughputInBps() − B^q_FS
9:     if (currentBuffer < minBuffer) or (q < q′) then
10:        return positiveMargins
11:    end if
12:    for each margin in possibleMargins do
13:        if (total_percent < maxPercent) and (r > 0) then
14:            positiveMargins ← positiveMargins ∪ margin
15:            p ← area(margin)/area(viewport)
16:            total_percent ← total_percent + p
17:            r ← r − bitrate(margin)
18:        else
19:            break;
20:        end if
21:    end for
22:    return positiveMargins
23: end function
```

In Algorithm 1, $segment\_duration$ denotes the duration for the segments described in the DASH manifest file. q is the viewport quality index picked by the ABR and q' is the previous ABR decision. The algorithm first checks the buffer and the last ABR decision. If ABR decides to downshift ($q < q'$) or the current buffer is less than the minimum buffer length, it will not select any PM tiles to not cause any stalls or an additional quality reduction in the viewport. Once these initial conditions are satisfied, the algorithm computes the remaining bitrate r by using a throughput measurement function called $getDashThroughputInBps()$. This function calculates the throughput using the downloaded bytes for one segment duration and keeps the maximum measured throughput to get an approximate bandwidth measurement. Using

(1) in [30], the required bitrate for a quality index q is known and with the measured bandwidth, the algorithm calculates the remaining bandwidth. The PM tiles are selected as long as there is remaining bandwidth and no more bits spend for the margins by checking $maxPercent$. $maxPercent$ value (*i.e.*, 30%) is used as in [28] in this study.

2.3 Complementary and Negative Margins

The extra bandwidth needed for downloading the PM tiles in the direction of the motion can be offset by dropping the quality for some viewport tiles that are not easily noticeable during the head motion. To do this, first, the CM tiles are calculated, which are the tiles in the direction of the motion, to be downloaded and they have been marked as possible PM tiles in the previous iteration. Then, the viewport tiles are determined in the opposite direction of the motion that can be unnoticeably rendered in the background quality. These are called NM tiles.

The CM and NM tiles can be seen in Figure 1 in addition to viewport, background and PM tiles. The combination of CM and NM results in a bandwidth-neutral solution. That is, even if there is initially no available bandwidth, it is possible to increase the viewport quality and decrease MTHQD using NM. CM and NM tiles are selected using Algorithm 2, where there are two configurable parameters: $maxNM$ and max_head_speed. For the experiments in this thesis, they are set to 30% and 120 degrees per second (dps), respectively. max_head_speed is the maximum degrees that a human can turn its head in a second. Using the speed (extracted using $getHeadMotionSpeed$) from the HMD device and max_head_speed, a $speed\_factor$ is calculated to select more NM tiles when the head speed increases.

Next, the algorithm loops on $unusedPositiveTiles$ (line 13) that are the remaining possible PM tiles unused in Algorithm 1 to select the CM tiles (line 23). In this loop, the algorithm sets the next element in the $unusedPositiveTiles$ to be a CM tile

Algorithm 2 Get Negative Margins

```
1: function GetNegativeMargins(unusedPositiveTiles, viewportTiles)
2:     maxNM ← 30%
3:     max_head_speed ← 120
4:     speed ← getHeadMotionSpeed()
5:     speed_factor ← 1
6:     NM ← ∅
7:     complementary ← ∅
8:     total_percent ← 0
9:     if (speed < max_head_speed) then
10:        speed_factor ← speed / max_head_speed
11:    end if
12:    p ← speed_factor × maxNM
13:    for each unusedPositiveTile in unusedPositiveTiles do
14:        if (total_percent >= p) then
15:            break
16:        end if
17:        margin ← unusedPositiveTile
18:        f ← findFarthestViewportTile(margin, viewportTiles, NM)
19:        area ← getVPIntersection(f)
20:        if total_percent + area < p then
21:            total_percent ← total_percent + area
22:            NM ← NM ∪ f
23:            complementary ← complementary ∪ margin
24:        end if
25:    end for
26:    return {NM, complementary}
27: end function
```

and then calculates the farthest viewport tile from this CM tile (line 18) to set that tile as an NM tile (line 22). In Algorithm 3, both the circular distance and angular distance from the motion vector are used. The Haversine formula calculates the circular distance or great-circle distance between two points on a sphere. The details of these circular and angular distance calculations can be found in [29]. Algorithm 3 finds the maximum distanced viewport tile and checks whether the calculated angle between two vectors (the vectors originated at the viewport center and ending in the viewport tile and margin tile) is larger than $\frac{\pi}{2}$, which makes sure that the viewport tile is in the reverse direction of the head motion. Then, the resulting farthest viewport

Algorithm 3 Find Farthest Viewport Tile

```
1: function findFarthestViewportTile(marginTile, viewportTiles, NM)
2:     result ← ∅
3:     max_distance ← 0
4:     maxAngularDistance ← 0
5:     for each viewport_tile in viewportTiles do
6:         if viewport_tile ∈ NM then
7:             continue
8:         end if
9:         d ← getCircularDistanceFromVP(marginTile)
10:        a ← getAngularDistanceFromMotion(viewport_tile)
11:        if d > max_distance and a > maxAngularDistance and a > π/2 then
12:            max_distance ← d
13:            maxAngularDistance ← a
14:            result ← viewport_tile
15:        end if
16:    end for
17:    return result
18: end function
```

tile is returned. In Algorithm 2, the area in the viewport is computed for this farthest viewport tile f. If the percentage does not exceed p (speed_factor $\times$ maximum allowed NM percentage), then the margin tile is selected as a CM tile and farthest viewport tile f is selected as an NM tile. After running the loop, the algorithm returns a set of CM and NM tiles. The ABR algorithm then switches NM tile quality levels to the background quality and sets the quality level of the CM tiles to the PM tile quality.

2.4 Migrating to H2

The tile download strategy is similar to the one explained in [30]. However, the network module is modified in the code base for this work and implemented H2-based segment download. Thanks to this, the race conditions observed in [30] are reduced. The problem caused by the HTTP/1.1-based transport is observed especially in the 12x8 tile grid, since the number of tiles is too many to create a new download thread for each of them. To that effect, first, the HTTP/1.1 implementation is tested and analyzed, and observed that the individual HTTP/1.1 connection threads were

creating a race condition in the transport layer. Consequently, there were stalls because of the deadlocks even without any bandwidth limitations. These observations are detailed in Chapter 4.

This problem is eliminated in this implementation thanks to H2's built-in stream multiplexing feature. With this feature, the tiles can be requested over the same H2 connection and this solves the initiation of the concurrent connections depending on the tile configuration. For instance, with the 12x8 tile configuration, there used to be at least 96 parallel threads for HTTP/1.1, whereas the thread count was reduced with H2. On the other hand, there is no the ability to abort an existing request with HTTP/1.1. Yet, H2 has the capability to abort an active request when there is an unnecessary download in progress. This allows the streaming client to apply ABR decisions faster, especially during fast head motions. `libcurl` 7.80.0 [34] is used to create a performant H2 implementation. As mentioned in `libcurl`'s documentation, the recommended way of handling concurrent multiplexed transfers in a high performance way in H2 is to use an asynchronous event framework with `libcurl` and integrate it using socket callbacks. That is why, `libcurl` is integrated with `libuv` 1.42.0.

2.5 Experimental Setup

The OMAF Player [35] and an HMD simulator running on a Windows machine (64-bit Windows 10 with 32 GB RAM, Intel ®Core™ i7-10875H CPU and NVIDIA GeForce RTX 2070 SUPER 8 GB GPU) are used to conduct the tests. This player implemented HTTP/1.1 for the segment downloads and it is enhanced by implementing H2. This way, HMAVM are analyzed on two different HTTP versions.

The implementation was evaluated using two 60-seconds long 30-fps 4K (*RollerCoaster*, *Timelapse*) video sequences from the head-motion dataset [36]. These represent the real-world scenario for this work since users view the content at slower

or faster speeds in any direction. These sequences were encoded using the Kvazaar encoder [37] at four quality levels at 20, 30, 40 and 50 Mbps, and three tiling configurations of 6x4, 8x6 and 12x8. The viewport size is chosen as 110x110 degrees.

Considering the bitrates of the generated content, two network bandwidth limits (with no delays introduced) are selected for evaluation: 40 and 60 Mbps. 40 Mbps is selected since this bandwidth may or may not be enough (depending on the tiling configuration) to download all viewport tiles in the highest quality and the remaining background tiles in the lowest quality. 60 Mbps is, on the other hand, sufficient to download all of the tiles in the highest quality. Note that any bandwidth limit less than 40 Mbps would produce an unpleasant video experience for high-quality VR content, hence, such results were omitted.

The segment duration was 300 milliseconds (ms) and each segment consisted of only one GoP (nine frames per segment). Initially the buffer duration was tested as 300 ms, one second and two seconds. The analysis showed that a one-second buffer gave the best results with the 300-ms segment duration. Shorter segment and buffer durations were chosen to allow a quick response to the viewport changes.

First, the tests were conducted with artificial head-motion files on *RollerCoaster* and *Timelapse*. These files consisted of the same head-motion pattern at exact speeds (*i.e.*, 25, 45, 60, 90, 120 dps). After that, the head-motion files recorded while the subjects were watching *RollerCoaster* and *Timelapse* from a human head-motion dataset [36] were used. The head-motion files consisted of four columns; timestamp, pitch, yaw and roll. These files were fed to the HMD simulator. The results generated using the user head-motion files were merged into one value with the same constraints (*e.g.*, *Timelapse* and 40 Mbps). Similarly, the results for the artificial ones were also combined to preserve the readability. For instance, in Figure 3, the left-most group (*i.e.*, Artificial) in the average viewport quality chart shows the combined results for *RollerCoaster* and *Timelapse* with all tile grids (*i.e.*, 6x4, 8x6 and 12x8) and

artificial head-motion files (*i.e.*, 25, 45, 60, 90, 120 dps) at 40 Mbps. The middle group (*i.e.*, *RollerCoaster*) shows the results for *RollerCoaster* with all grids and 63 human head-motion files. The right-most (*i.e.*, *Timelapse*) shows the same but for *Timelapse*.

The evaluation criteria were the same as [29, 30]. The average throughput, average viewport quality, number of viewport quality level changes and MTHQD were used to evaluate HMAVM. To preserve the thesis' readability, only the average MTHQD is provided, viewport quality and throughput results. The viewport quality is calculated as given in [22]. That is, each tile is weighted by the percentage of the viewport it is covering. The average viewport quality is calculated by summing the multiplication of the tile qualities by their weights. As given in [22], the quality levels are numbered from 1 to 4, where 1 is the highest (*i.e.*, 50 Mbps) and 4 is the lowest (*i.e.*, 20 Mbps). As explained in Chapter 4, MTHQD is the delta time from a viewport containing at least one lower quality tile to a viewport fully covered by the highest quality tiles. It is an important user experience metric since it gives insights about changes in the quality of the content rendered within the viewport.

CHAPTER III

HEAD-MOTION-AWARE VIEWPORT QUALITY METRIC

The Head-motion-aware Viewport Quality (HMAVQ) metric was designed starting with ProbGaze [23] as the baseline that takes into account the eye gaze distribution within the viewport. By incorporating the peculiarities of head motion, the goal in this study is to quantify the overall viewport quality more accurately than the previous approaches. Studies [26, 27] show that users cannot always view the entire viewport but instead focus on a specific part of it. Users also tend to view the viewport without focusing during a fast head motion and the actual quality of the viewport during this motion is not as relevant. This implies that not every viewport a user has viewed has an equal weight in determining the user's overall experience.

In what follows, first, the calculation for the quality of an individual viewport during head motion and then the calculation of the overall viewport quality are presented.

3.1 Individual Viewport Qualities

Each viewport was sampled with n circles and m points on each circle. Algorithm 4 describes the core of the head-motion-aware sampling model for the viewport with the height (h) and width (w) of 110-degrees. In this model, the circles are indexed from 1 to n, and each circle's diameter (d_i) polynomially grows from inward to outward as in (1) in line 8 of Algorithm 4 where r_i denotes the radius. When the smallest circle's diameter (the distance between the viewport center and the first circle) was calculated using the equations in [23], the result was 10.53 degrees, which was approximately

one-tenth of the viewport's one dimension. Therefore, the largest circle's diameter (d_n) equals the viewport's one dimension, while the smallest circle's diameter (d_1) is one tenth of the largest. The centers of these circles are shifted from the viewport center with the formula given in (2) in line 9 of Algorithm 4, where $speed_x$ and $speed_y$ represent the head-motion speeds in the horizontal and vertical axes, respectively. So, $circle_i$ is shifted by x_i degrees horizontally and y_i vertically. In (2) in line 9 of Algorithm 4, $speed\_threshold$ specifies the speed threshold beyond which the content rendered in the viewport starts becoming unfocused (blurry). This parameter had been determined based on a small-scale user study that was conducted for an earlier work in [38].

The circle weights are assigned linearly in descending order from inward to outward per (3) in line 10 of Algorithm 4. The weight of $circle_i$ (w_i) is calculated by dividing its reversed order by the sum of the circle indexes. Subsequently, the points on each circle are distributed with an angular distance of $2 \times \pi/m$ between them. The coordinates of each point are calculated with parametric equations of the circle. These coordinates are then used inside the function $getPointQuality$ to find the tile T with the projection of the point. The quality of the tile T is recorded as $p.q$, which is then divided by the number of points (m) to find the average quality of $circle_i$. The resulting value is then multiplied by w_i and added to the variable $quality$, which finally indicates the individual viewport quality.

Example sampled viewports at different speeds can be seen in Figure 2. Each example represents a viewport, while the vertical and horizontal lines are the boundaries of the tiles named as T_i within that viewport. Figure 2a shows the case where the viewport is stationary (no head motion). Figure 2b and Figure 2c demonstrate the shifted circles for a viewport viewed at a relatively slow and fast speed, respectively. Figure 2d shows the case where the head moves faster than the speed threshold in the horizontal axis.

Algorithm 4 Individual Viewport Quality

1: **function** CALCULATEVIEWPORTQUALITY(viewport)
2: $speed_x \leftarrow viewport.speed_x$
3: $speed_y \leftarrow viewport.speed_y$
4: $tiles \leftarrow viewport.tiles$
5: $index\_sum \leftarrow \sum_{i=1}^{n} i$
6: $quality \leftarrow 0$
7: **for** $i := 1$ **to** n **step** 1 **do**
8:

$$d_i = 2 \cdot r_i = 0.88 \cdot i^2 + 1.38 \cdot i + 8.75 \tag{1}$$

9:

$$x_i = \frac{(\frac{w}{2} - r_i) * speed_x}{speed\_threshold}$$
$$y_i = \frac{(\frac{h}{2} - r_i) * speed_y}{speed\_threshold} \tag{2}$$

10:

$$w_i = \frac{n - i + 1}{index\_sum} \tag{3}$$

11: $average \leftarrow 0$
12: **for** $angle := 0$ **to** 2π **step** $2\pi/m$ **do**
13: $p \leftarrow new\ Point()$
14: $p.x \leftarrow r_i \cdot \cos(angle) + x_i$
15: $p.y \leftarrow r_i \cdot \sin(angle) + y_i$
16: $p.q \leftarrow getPointQuality(p.x, p.y, tiles)$
17: $average \leftarrow average + \frac{p.q}{m}$
18: **end for**
19: $quality \leftarrow quality + w_i \cdot average$
20: **end for**
21: $viewport.quality \leftarrow quality$
22: return $viewport$
23: **end function**

3.2 Overall Viewport Quality

The second step is to calculate the overall viewport quality using Algorithm 5 over a period of time from the individual viewport qualities calculated by Algorithm 4. First a normalized weight (weight is divided by the number of viewports) was assigned to each viewport in lines 7-11, and then add its weighted quality to calculate the overall viewport quality. The idea is to give less weight to viewports viewed at faster head motions, whereas those viewed in stationary head positions or at slower head motions

are assigned a larger weight.

By design, the function *assignWeight* gives the lowest weight for the viewports where the head-motion speed is equal to or higher than *speed_threshold*. However, how should it assign the weights for the stationary viewports or where the head motion is not too fast? This is unfortunately not a trivial task and the best one can do is to find a good method empirically. To do this, first, a linear assignment was tried and then logarithmic weights were tested. The latter turned out to be a better approach and was also more consistent with the guidance received from an optometrist. At the end, the implementation was further simplified, so the assignment of the logarithmic weights was approximated by adopting a threshold approach. After substantial testing, the use a weight of one for the viewports where the *speed_threshold* was reached or exceeded, and two for the other viewports were determined. More sophisticated *assignWeight* functions might perform better and this further discussed in Section 4.1.

Algorithm 5 Overall Viewport Quality

```
1: function calculateOverallQuality(viewports)
2:     quality ← 0
3:     vp_count ← length(viewports)
4:     for each viewport in viewports do
5:         vp_quality ← viewport.quality
6:         vp_speed ← viewport.speed
7:         if speed ≥ speed_threshold then
8:             weight ← 1 / vp_count
9:         else
10:            weight ← 2 / vp_count
11:        end if
12:        weighted_quality ← vp_quality · weight
13:        quality ← quality + weighted_quality
14:    end for
15:    return quality
16: end function
```

3.3 Experiments and Analysis

3.3.1 Experimental Setup

For the experiments, Nokia's public Omnidirectional Media Application Format (OMAF, ISO/IEC 23090-2)-based DASH player [35] enabled for using the Head-motion-aware Viewport Margins (HMAVM) algorithm [29, 38] was used. As the setup, a buffer duration of one second, segment size of 300 milliseconds (ms), encoding bitrates of 20, 30, 40 and 50 Mbps, frame rate of 30 frames per second, bandwidth limits of 40 and 60 Mbps and grid configurations of 6x4, 8x6 and 12x8 were used. Also, the runs using only H2 since this avoided the thread race conditions and lock problems associated with using HTTP/1.1 were completed. Details on this particular setup and set of values are given in [29, 38].

In the experiments, a public dataset used in [27] was benefited. Three videos with 4K resolution (3840x2160 pixels) were picked along with their gaze traces and head-motion data. The videos down to 60 seconds using the starting point of the segment information mentioned in the dataset were trimmed. In total, there were 24 head-motion files for each video since six users had viewed each with four different tasks. However, only two tasks (Task 1 and Task 3) were used since Tasks 2 and 4 had head motions at slower speeds and they were not an interesting scenario to show the impact of head motions on viewport quality. Thus, they were omitted from the results. The statistical analysis of the speeds observed in these tasks can be seen in Table 1.

Table 1: Statistics of the head motions (first (Q_1), second (Q_2) and third (Q_3) quartiles, and mean) as degrees per second (dps).

Task Name	Q_1	Q_2	Q_3	$Mean$
Task 1 (free viewing)	5.39	14.14	26.02	23.92
Task 2 (visual search)	2.24	4.24	8.06	6.87
Task 3 (saliency)	13.42	26.02	40.0	45.56
Task 4 (track)	1.41	2.24	5.39	7.02

In addition to this head-motion dataset, also artificial head motions at constant speeds (25, 45, 60, 90 and 120 dps) were utilized while developing the HMAVQ metric. In the experiments, *speed_threshold* in Algorithm 4 was fixed to 60 dps based on the pre-processing phase of this work with a limited number of users.

Above the surface, the setup and testing might look trivial. However, it is a considerably complicated workflow to properly collect all the data needed for a detailed analysis. For this purpose, the player logged the entire viewport information (*e.g.*, timestamp, coordinates, speed in horizontal and vertical axes, tiles) at every 30 ms. In this work, these logged viewports are referred to as the individual viewports (110x110 degrees). This viewport information was used with the corresponding head-motion data and gaze traces to construct and quantify the quality metrics listed below. In the end, a total of 1224 iterations (864 with human and 360 with artificial head motions) were completed, which added up to about 25 hours of viewed content.

3.3.2 Compared Metrics

For the comparisons, the following quality metrics were used:

- Weighted Average Viewport Quality (WA): Each tile is weighted by the percentage of the viewport it is covering. The viewport quality is calculated by summing the multiplication of the tile qualities by their weights [22].

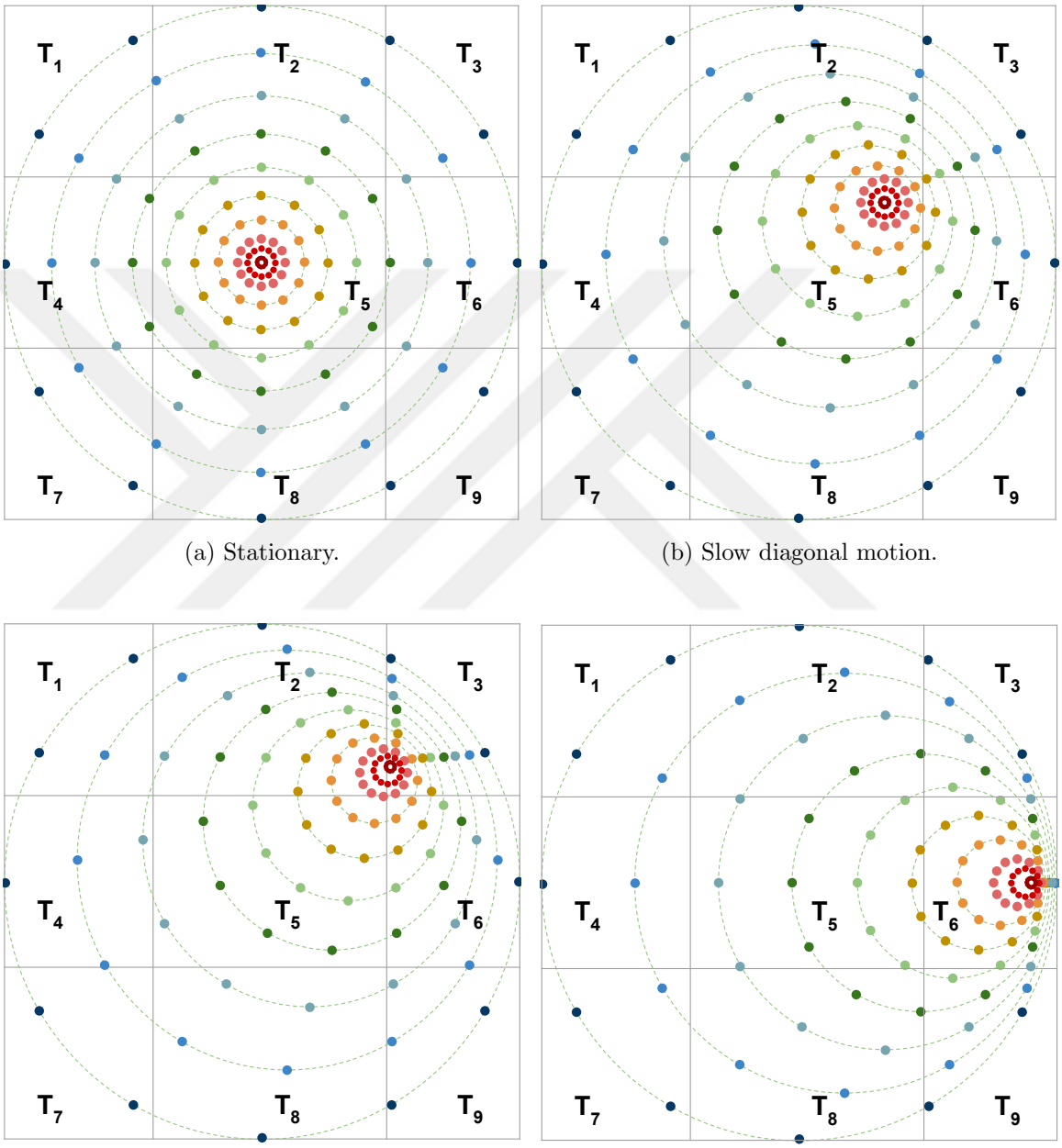
- Center Tile Quality (CT): The viewport quality is determined by the quality of the tile at the center of the viewport.

- ProbGaze (PG): This takes into account the distribution of eye movements within the viewport sampled by uniformly placed points on the polynomially distributed circles. The quality is the weighted average of the qualities projected at the points on each circle [23].

- HMAVQ (HM): The overall viewport quality is the weighted average of the individual viewport qualities calculated by Algorithm 4 as described in Section 3.2.

For all these quality metrics, the quality is defined as given in [22]. That is, the quality levels are labeled 1 through 4, where 1 indicates the highest (*i.e.*, 50 Mbps) and 4 indicates the lowest quality (*i.e.*, 20 Mbps).

The primary goal in this comparison is not to compare the absolute quality values produced by each metric (even though these values are provide). It is rather to show how the existing metrics (WA, CT and PG) ignored the effect of the head motions when computing the objective viewport quality. The amount of potential inaccuracy introduced due to this omission can only be quantified through a comprehensive user study, which is the next goal in this project.



(a) Stationary.

(b) Slow diagonal motion.

(c) Fast diagonal motion.

(d) Faster horizontal motion.

Figure 2: Sampled viewports at different head-motion speeds.

CHAPTER IV

RESULTS AND CONCLUSIONS

4.1 HMA Viewport Margins

In this section, the experimental results are presented. First, the results and their respective analysis for H2 using the constraints mentioned in Section 3.3 are provided. Then, the findings for HTTP/1.1 with the same constraints are shared. The results in this section are grouped together as explained in Section 3.3. They show the results generated using *RollerCoaster* and *Timelapse* since these sequences are taken from the dataset and the results of human and artificial head motions can be compared. In all the charts in this section, the left-most (blue) bars represent the results for the no-margin case. The middle (magenta) bars show the results for using the PM tiles only. The right-most (orange) bars represent the results for using the HMAVM tiles that include PM, NM and CM tiles. Since none of the viewport quality values in the results exceeds 1.60, the upper boundary is fixed at 1.60 for the viewport quality charts in this section.

4.1.1 H2 Results

4.1.1.1 Average Viewport Quality at 40 Mbps

In Figure 3, the viewport quality (left) and MTHQD (right) results using H2 for 40 Mbps bandwidth are shown for the human and artificial head motions. Overall, using HMAVM improved the average viewport quality by 6%, 5% and 5% on the artificial, *RollerCoaster* and *Timelapse* head motions, respectively, over the no-margin case. Since this chart indicates the combined results for all tile configurations, the improvements seem relatively minor. Checking the individual runs, however, the significant improvements are observed up to 13%, 18% and 17% for the artificial,

RollerCoaster and *Timelapse* head motions, respectively, all in the 8x6 tile grid configuration.

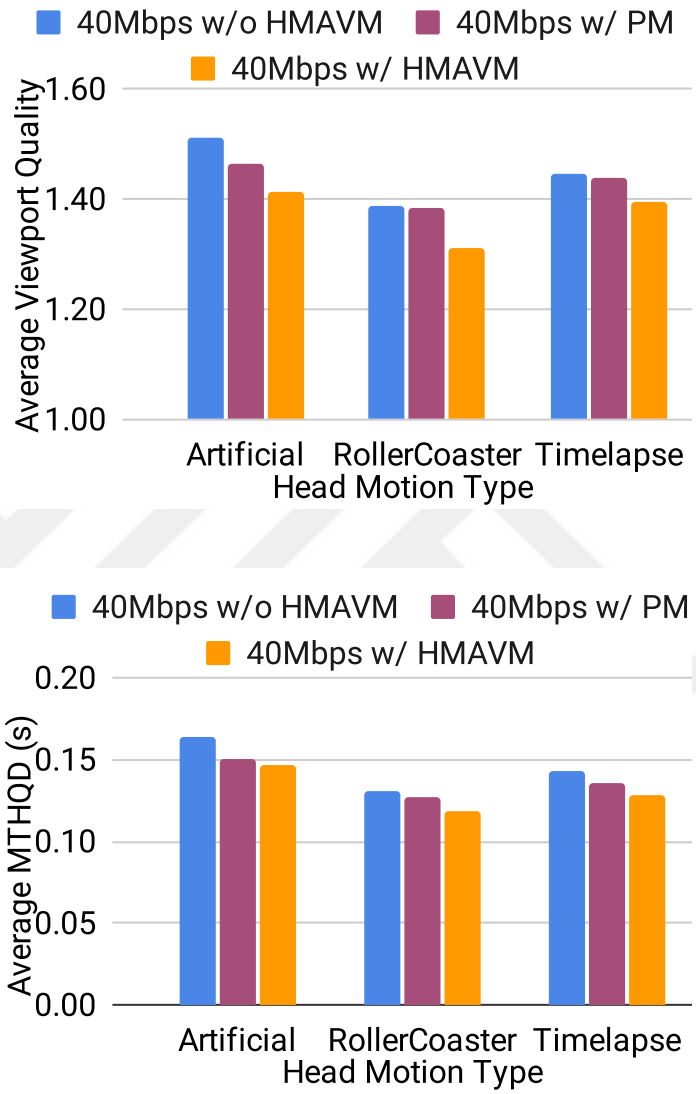


Figure 3: Average viewport quality (top) and MTHQD (bottom) results for 40 Mbps.

In these experiments, the best viewport quality is achieved with the HMAVM, except for the 25 dps case in the 6x4 configuration, since tile sizes are bigger in this configuration compared to the others. For the 6x4 configuration with 25 dps slow motion, even using the PM tiles is not improving the viewport quality as the head motion is too slow to hit an extra downloaded margin. However, for the other

configurations, tile sizes are smaller and even at slow motion, the margin hits are likely got by improving the viewport quality.

The reason for the higher viewport quality is due to the great harmony of the CM and NM tiles. The NM tiles (within the viewport) start downloading the next segment in low quality and subsequently, additional high-quality CM tiles can be downloaded in the head-motion direction without consuming additional bandwidth. However, if a high-quality segment for an NM tile has already been downloaded, there is no need to replace it. On the other hand, if a low-quality segment for a CM tile has already been downloaded and replaced with its higher quality version. This results in an immediate quality improvement for the CM tiles. For the same reason, also substantial reductions are observed in MTHQD (as discussed below) without increasing the bandwidth consumption. As the number of tiles increases and the size of the tiles decreases, HMAVM provide larger improvements. HMAVM' improvements also get more sizeable with the increasing speed of the head motion. These observations are also valid for the 60 Mbps scenario.

4.1.1.2 Average MTHQD at 40 Mbps

The right chart in Figure 3 shows the MTHQD results. Thinking of the MTHQD metric as explained in Section 3.3, using NM increases the MTHQD where in return, using CM reduces it when the viewport hits a CM tile. To reduce the MTHQD, there should be more margin hits where the PM and CM tiles are downloaded in the motion direction. In the experiments, the viewport quality is improved and the MTHQD is reduced with HMAVM for both the human and artificial head motions. Looking at the same chart, there are 10%, 9% and 11% MTHQD reduction in the artificial, *RollerCoaster* and *Timelapse* head motions, respectively. Since 40 Mbps is generally insufficient to download extra margin tiles, a smaller reductions in the MTHQD were already expected. In the artificial head-motion experiments, the most

significant reduction for the MTHQD is 50 ms for the 8x6 tile configuration and 60 dps, which is a 26% reduction compared to the no-margin case.

4.1.1.3 Average Viewport Quality at 60 Mbps

In the left chart in Figure 4, the average viewport quality results are shown. Recall that 60 Mbps is sufficient to stream the highest quality segments in the setup of this study. However, the background tiles are not downloaded in the highest quality as this naturally wastes the bandwidth. For a proper comparison, the use of PM tiles were limited (see $maxPercent$ in Algorithm 1) and also the background tiles were enforced to be in the lowest quality. Compared to using the PM only, HMAVM slightly improved the average viewport quality even with a reduced bandwidth consumption. Compared to the no-margin case, an average improvement of 9% (up to 13%), 9% (up to 20%) and 11% (up to 19%) was observed in the average viewport quality for the artificial, *RollerCoaster* and *Timelapse* head motions, respectively.

4.1.1.4 Average MTHQD at 60 Mbps

The right chart in Figure 4 shows the MTHQD results, where the majority of the MTHQD reduction comes from the PM tiles since there is enough bandwidth to download more PM tiles at 60 Mbps. In all the tests, the MTHQD is reduced compared to the no-margin case. In addition, HMAVM provide a further reduction in the MTHQD compared to using the PM only due to the use of CM. The reduction is observed as 29%, 39% and 44% on average for the artificial, *RollerCoaster* and *Timelapse* head motions, respectively. The most significant reduction at 60 Mbps is 36% (at 45 dps with the 8x6 tile grid) for the artificial head motion. Similarly, it is 50% in the *Timelapse* head motion with the 8x6 tile grid. These results show us that HMAVM are beneficial at both low and high-bandwidth conditions.

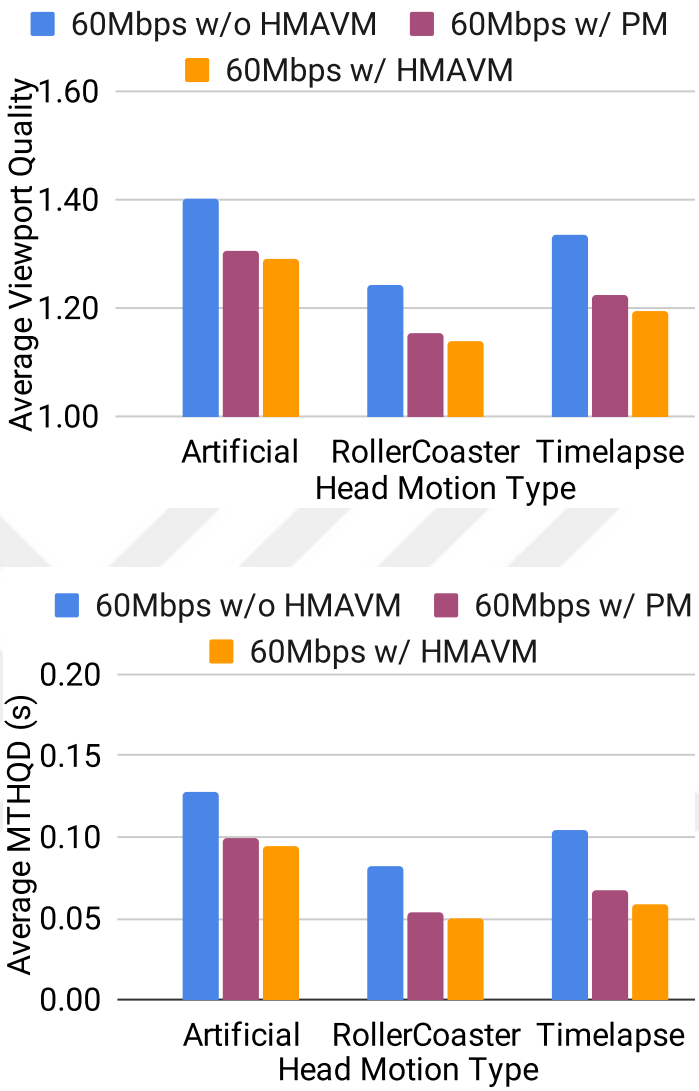


Figure 4: Average viewport quality (top) and MTHQD (bottom) results for 60 Mbps.

4.1.1.5 Average Throughput

Figure 5 shows the throughput averaged across all the experiments. At 40 Mbps, PM and HMAVM consume a bandwidth similar to the no-margin case, but they still provide improvements (Figure 3). At 60 Mbps, PM and HMAVM use more bandwidth and provide even larger improvements (Figure 4).

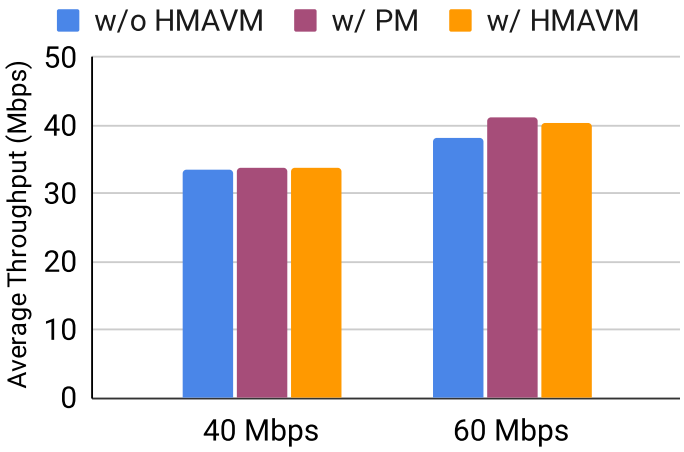


Figure 5: Average throughput.

4.1.2 HTTP/1.1 Results

The experiments above were repeated with HTTP/1.1. In Figure 6, which shows the results for the 40 Mbps scenario, where the observation is the MTHQD is not significantly reduced (*i.e.*, 5% for the artificial and 0% for the *RollerCoaster* and *Timelapse* head motions). When the average viewport quality is observed, HMAVM' output is still slightly better (5%, 2% and 3% for the respective head motions) than using no margins. Yet, the improvement with H2 was larger. The reason is that with HTTP/1.1, there is no the capability to abort an existing request as there is with H2, which is an essential feature with the PM and CM tiles as the system may frequently need to cancel an existing request and make a new one. In Figure 7, the average viewport quality and MTHQD for HTTP/1.1 are shown for the 60 Mbps scenario. Although the viewport quality improvements (9%, 7% and 9%) are close to what is observed with H2 (9%, 9% and 11%), the average MTHQD reductions (29%, 37% and 36%) with HTTP/1.1 are slightly lower than the ones with H2 (29%, 39% and 44%). Thus, using H2 is recommended for implementing the HMAVM.

4.2 HMA Viewport Quality Metric

In this section, first, the comparison for the individual viewport qualities was presented for WA, CT, PG and Algorithm 4, then the comparison for the overall viewport quality results for all these metrics plus HM. This two-part analysis is important to understand the contribution of each step in the development of HMAVQ and provides us useful insights.

4.2.1 Individual Viewport Quality Comparisons

In Figure 8a, individual viewport quality values are plotted as a time series (60 seconds) for WA, CT, PG and Algorithm 4 for one of the users viewing the *Timelapse* sequence (using a 8x6 tile grid). This particular output corresponds to a user running Task 1 (free viewing), which involves mostly casual head motions.

Examining each metric individually, it can be observed that WA often indicates worse quality than the other metrics do. This is expected since WA considers all the tiles within the viewport irrespective of whether the user actually focuses on them. Due to the same reason, WA also indicates better quality for the viewports the other metrics indicate the worst quality (*i.e.*, 4). On the other hand, CT almost always indicates the best quality (*i.e.*, 1) except for a few viewports where the quality drops to 4. Recall that CT ignores every tile in the viewport except the center one and is clearly not a true representative metric. These observations align well with the initial motivations for developing the HMAVQ metric.

Looking at PG, it can be seen that all the quality drops indicated by CT are also captured by PG, although PG indicates worse quality in the majority of the time than CT as it considers where the user is focusing on. Finally, Algorithm 4's values are quite close to PG's. This is expected due to the limited amount of head motion involved in this particular run. The main difference between the two is after 51 seconds into the playback, where the head-motion data reveals that there is a

head motion for several seconds. Consequently, Algorithm 4 shifts the circles in the direction of the head motion and some of the circles end up in low-quality tiles, resulting in a worse quality than PG measures.

In Figure 8b, individual viewport quality values are plotted only for PG and Algorithm 4 (WA and CT are omitted for brevity) for another user viewing the *Timelapse* sequence. Differently from the previous time series, this user runs Task 3 (saliency), which involves the fastest head motions within the dataset. The periods where the head-motion speed was equal to or higher than *speed_threshold* are shown in gray background. At such periods, the differences between PG and Algorithm 4 become more evident as the faster head motions cause significant shifts for the circles used in the calculation of the viewport quality. These differences by two specific cases are exemplified:

Algorithm 4 indicating worse quality than PG: Referring to Figure 8b, at 10.857 seconds, it can be seen that Algorithm 4 and PG indicate a viewport quality of 2.79 and 1.06, respectively. The corresponding (instantaneous) viewport (horizontal speed is 79.0 dps and vertical speed is 4.3 dps) is overlaid with the circles used by PG and Algorithm 4 in Figure 9, where the viewport tiles along with their actual qualities are also provided. At this time instant, the user turns right and at the right edge of the viewport, tile qualities are of the lowest quality. Since Algorithm 4's circles put more weight on these lowest-quality tiles, the resulting individual viewport quality turns out to be worse than what PG measures.

Algorithm 4 indicating better quality than PG: Referring to Figure 8b, at 48.939 seconds, it can be seen that Algorithm 4 and PG indicate a viewport quality of 1.29 and 3.26, respectively. The corresponding (instantaneous) viewport (horizontal speed is 31.4 dps and vertical speed is 4.2 dps) is overlaid with the circles used by PG and Algorithm 4 in Figure 10, where the viewport tiles along with their actual qualities are also provided. At this time instant, the user turns left and a bit downward and at the

left-bottom edge of the viewport, tile qualities are already high whereas the tiles in the opposite direction of the head motion are of low quality. Since Algorithm 4's circles put more weight on these highest-quality tiles, it calculates an individual viewport quality higher than what PG does.

4.2.2 Overall Viewport Quality Comparisons

Figure 11 shows the overall viewport qualities for WA, CT, PG and HM along with the basic arithmetic average of the individual viewport qualities produced by Algorithm 4, denoted by Avg(Algorithm 4). The left and right graphs are for Task 1 and Task 3, respectively. In both graphs, WA measures the worst overall viewport quality. This is not surprising since WA considers all the tiles within the viewport regardless of where the user focuses on. In contrast, CT is quite optimistic and always returns the best quality as it is not affected by the low-quality tiles at the edge of the viewport. On the other hand, PG outputs an overall viewport quality between WA and CT.

Looking at the graphs, HM indicates a lower overall viewport quality than PG for both tasks. This tells us that the tiles HM put more weight on (*i.e.*, the ones in the direction of the head motion) had (on the average) a lower quality than the ones PG weighted more. Also, the difference between PG and HM grows as the head-motion speed increases (from Task 1 to Task 3). This is expected since when the head-motion speed increases, the possibility of having a low-quality tile in the head motion direction increases, too. Yet, as the head-motion prediction schemes improve, the rate-adaptation algorithm within the player may do a better job in terms of fetching higher-quality tiles in the direction of the head motion, resulting in an increased overall viewport quality. Such quality increases can be captured accurately only by HM, though.

Finally, the graphs show that the difference between Avg(Algorithm 4) and HM is not significant in Task 1. This, however, does not mean that the second step in

HMAVQ, which calculates the overall viewport quality from the individual viewport qualities using head-motion-speed-based weights instead of equal weights, is not essential as it can be seen in Task 3. Moreover, Figure 11 averages the results across 1224 runs and this averaging process cancels out the differences between Avg(Algorithm 4) and HM. Examining the individual runs, it can be observed as much as 13% difference between the two, which emphasizes the need for the second step in HMAVQ.

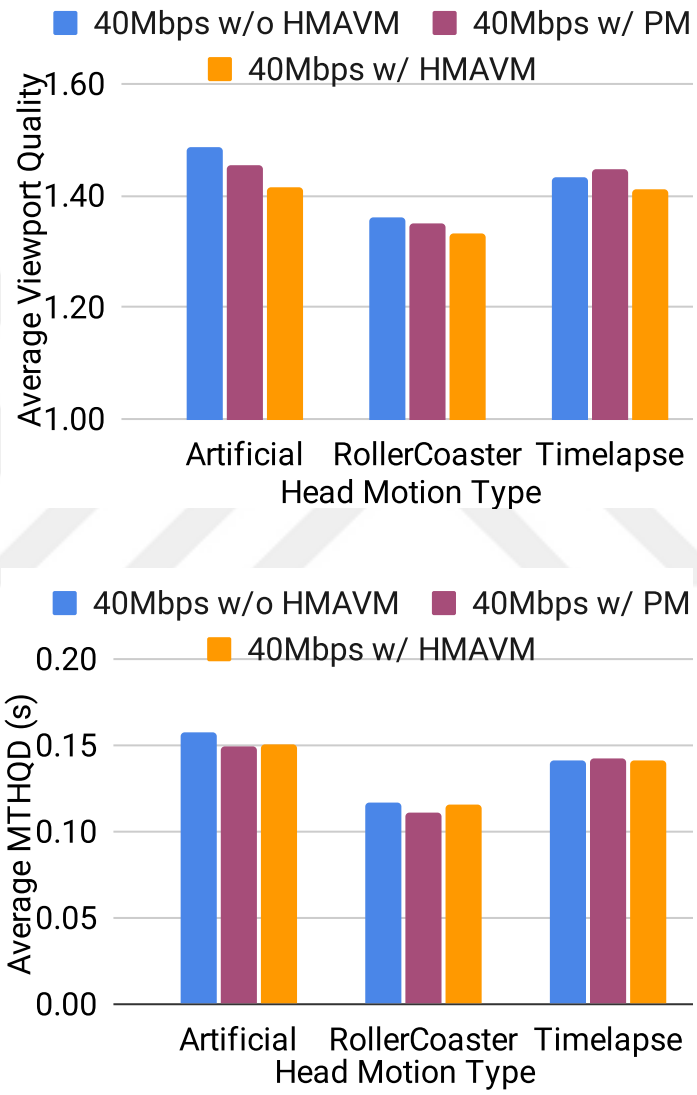


Figure 6: Average viewport quality (top) and MTHQD (bottom) results for 40 Mbps using HTTP/1.1.

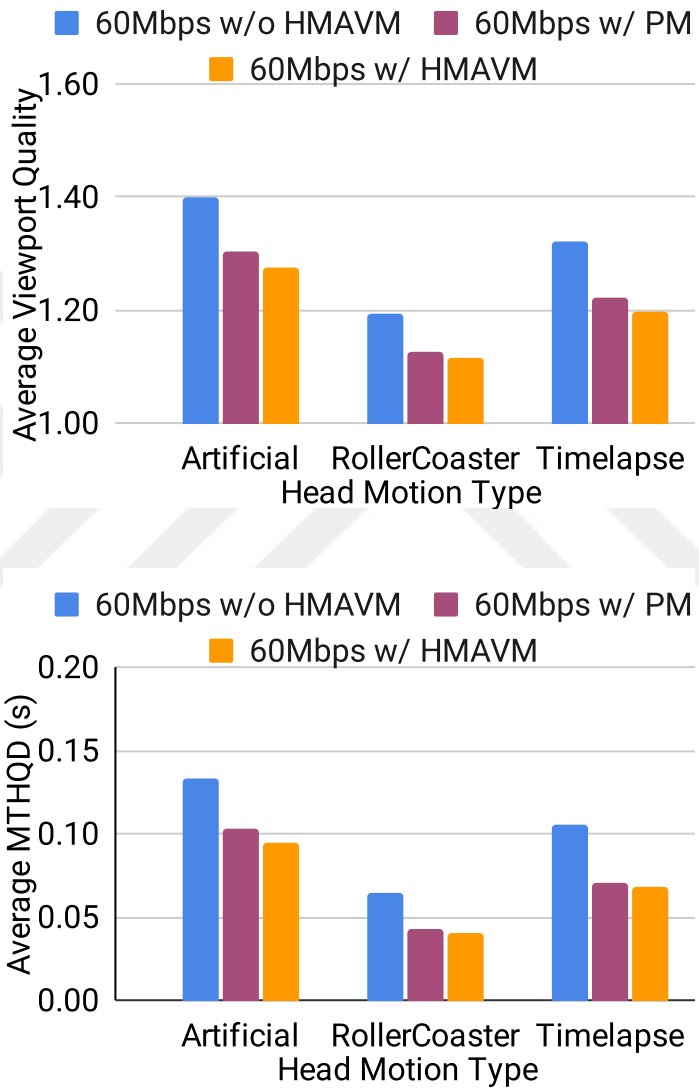
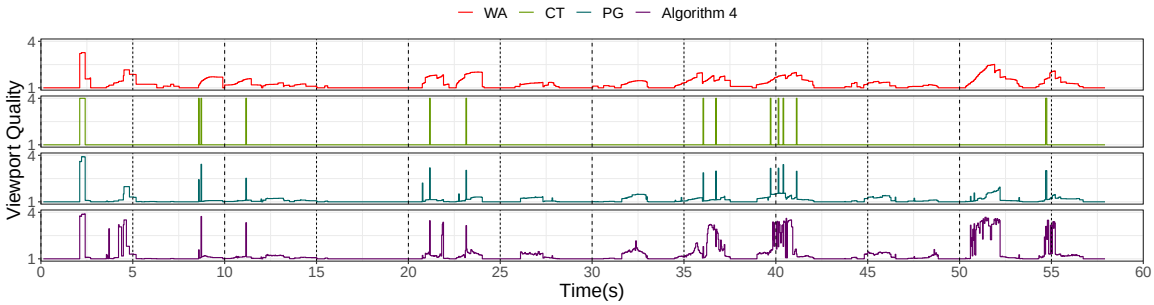
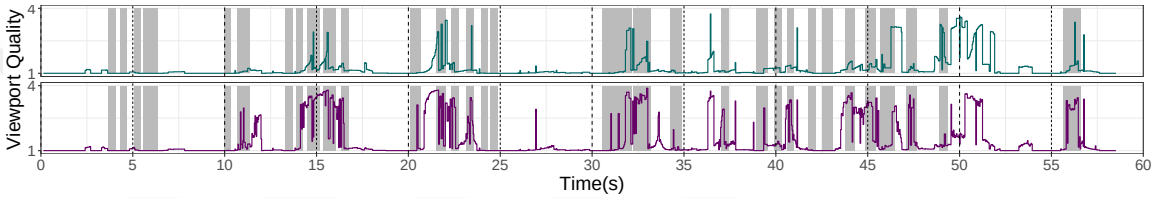


Figure 7: Average viewport quality (top) and MTHQD (bottom) results for 60 Mbps using HTTP/1.1.

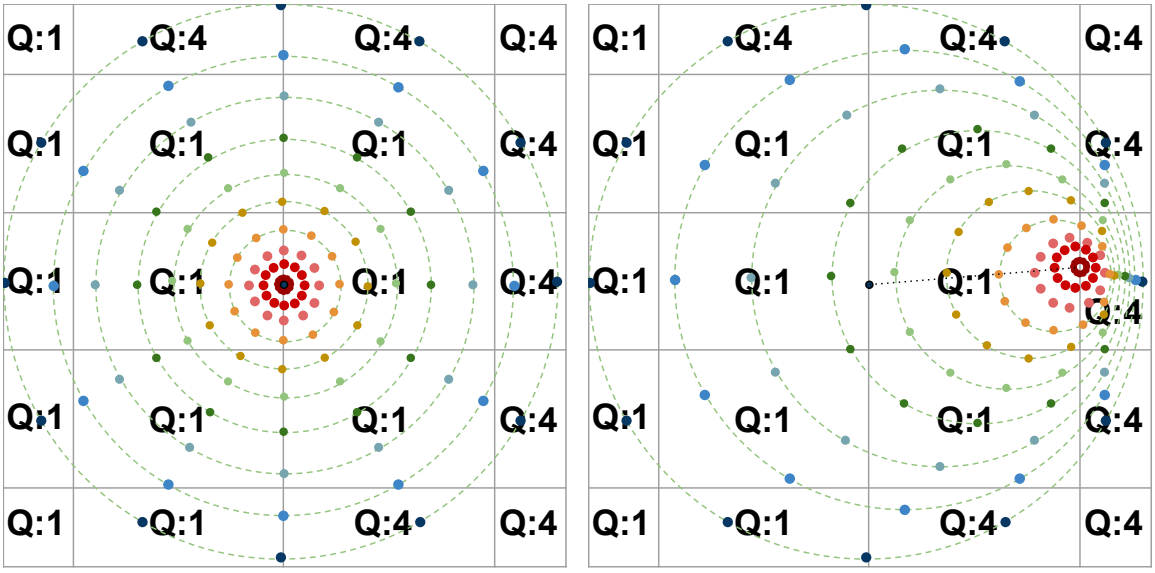


(a) Task 1.



(b) Task 3.

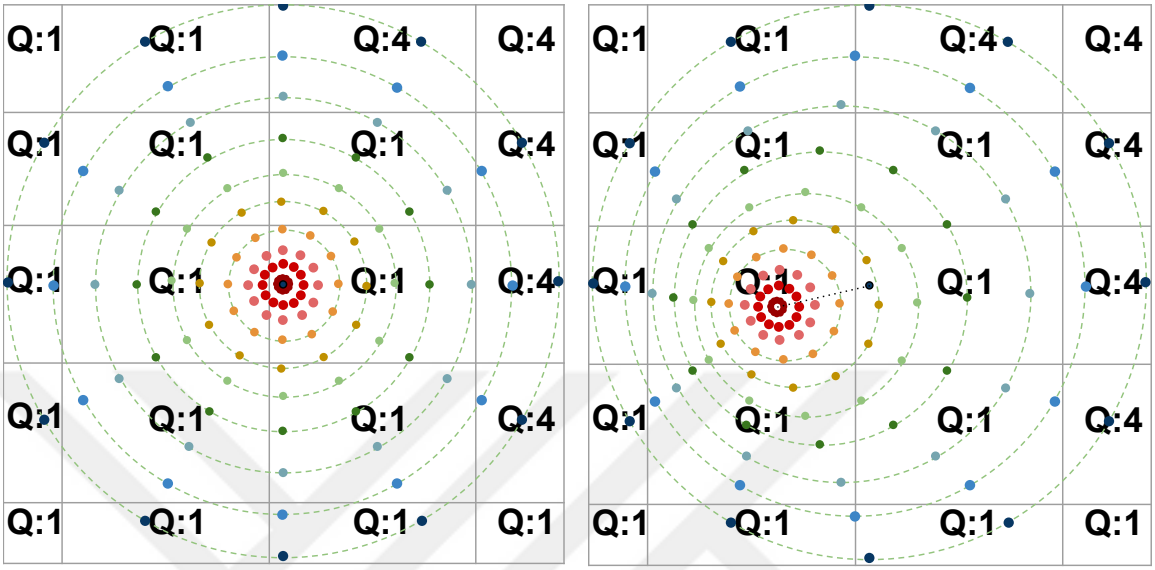
Figure 8: Comparison of WA, CT, PG and Algorithm 4 for a user viewing the *Time-lapse* sequence for Task 1 (a) and Task 3 (b), respectively. A lower viewport quality value indicates better quality.



(a) PG.

(b) Algorithm 4.

Figure 9: The instantaneous viewport at 10.8th seconds into the playback overlaid with the circles used by PG and Algorithm 4.



(a) PG. (b) Algorithm 4.

Figure 10: The instantaneous viewport at 48.9th seconds into the playback overlaid with the circles used by PG and Algorithm 4.

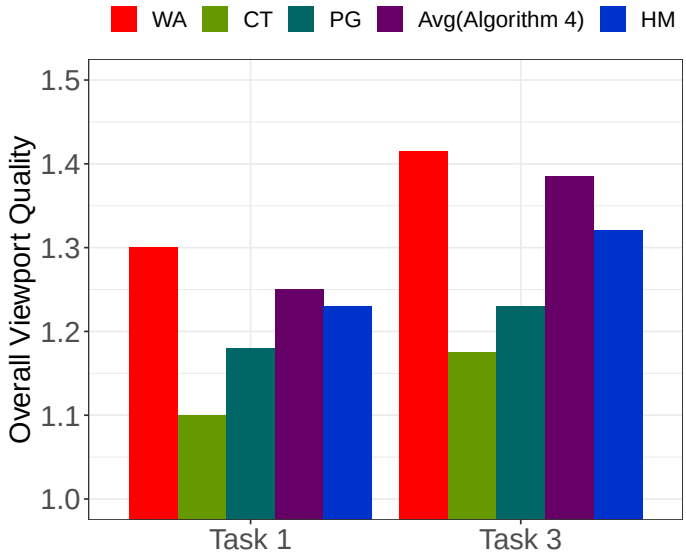


Figure 11: Overall viewport quality averaged across all the runs for Task 1 and Task 3.

REFERENCES

[1] M. M. Hannuksela, Y.-K. Wang, and A. Hourunranta, "An overview of the OMAF standard for 360° video," in *DCC*, 2019.

[2] "ISO/IEC 23009-1:2017 information technology — dynamic adaptive streaming over HTTP (DASH) — part 1: Media presentation description and segment formats." [Online] Available: `https://www.iso.org/standard/79329.html`. Accessed on Apr. 1, 2022.

[3] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "HEVC-compliant tile-based streaming of panoramic video for virtual reality applications," in *ACM Multimedia*, 2016.

[4] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *IEEE ICC*, 2017.

[5] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based HEVC video for head mounted displays," in *IEEE ISM*, 2016.

[6] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *IEEE ISM*, 2016.

[7] I. D. Curcio, H. Toukomaa, and D. Naik, "Bandwidth reduction of omnidirectional viewport-dependent video streaming via subjective quality assessment," in *AltMM*, 2017.

[8] D. Naik, I. D. D. Curcio, and H. Toukomaa, "Optimized viewport dependent streaming of stereoscopic omnidirectional video," in *Packet Video Workshop*, 2018.

[9] J. Son, D. Jang, and E.-S. Ryu, "Implementing motion-constrained tile and viewport extraction for VR streaming," in *ACM NOSSDAV*, 2018.

[10] J. Son, D. Jang, and E.-S. Ryu, "Implementing 360 video tiled streaming system," in *ACM MMSys*, 2018.

[11] L. Xie, X. Zhang, and Z. Guo, "CLS: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming," in *ACM Multimedia*, 2018.

[12] 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 *ACM Multimedia*, 2017.

[13] J. Fu, X. Chen, Z. Zhang, S. Wu, and Z. Chen, "360SRL: A sequential reinforcement learning approach for ABR tile-based 360 video streaming," in *IEEE ICME*, 2019.

[14] X. Jiang, Y.-H. Chiang, Y. Zhao, and Y. Ji, "Plato: Learning-based adaptive streaming of 360-degree videos," in *IEEE LCN*, 2018.

[15] A. T. Nasrabadi, A. Samiei, and R. Prakash, "Viewport prediction for 360° videos: A clustering approach," in *ACM NOSSDAV*, 2020.

[16] D. Monakhov, I. D. Curcio, and S. Mate, "On data wastage in viewport-dependent streaming," in *IEEE MMSP*, 2019.

[17] I. D. D. Curcio, H. Toukomaa, and D. Naik, "360-degree video streaming and its subjective quality," *SMPTE Motion Imaging Journal*, vol. 127, no. 7, pp. 28–38, 2018.

[18] S. L. Yasakethu, C. T. Hewage, W. A. Fernando, and A. M. Kondoz, "Quality analysis for 3d video using 2d video quality models," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1969–1976, 2008.

[19] R. Schatz, A. Zabrovskiy, and C. Timmerer, "Tile-based streaming of 8k omnidirectional video: subjective and objective QoE evaluation," in *QoMEX*, pp. 1–6, 2019.

[20] M. Yu, H. Lakshman, and B. Girod, "A framework to evaluate omnidirectional video coding schemes," in *ISMAR*, pp. 31–36, 2015.

[21] Y. Sun, A. Lu, and L. Yu, "Weighted-to-spherically-uniform quality evaluation for omnidirectional video," *IEEE Signal Processing Letters*, vol. 24, no. 9, pp. 1408–1412, 2017.

[22] 3GPP Technical Specification 26.118 v.16.0.2, "Virtual reality (VR) profiles for streaming applications (release 16)." [Online] Available: `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3325`. Accessed on Apr. 1, 2022.

[23] J. van der Hooft, M. Torres Vega, S. Petrangeli, T. Wauters, and F. De Turck, "Quality assessment for adaptive virtual reality video streaming: A probabilistic approach on the user's gaze," in *22nd Conf. Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 19–24, 2019.

[24] H. T. T. Tran, T. H. Hoang, P. N. Minh, N. P. Ngoc, and T. C. Thang, "A perception-based quality metric for omnidirectional images," in *ICCE-Asia*, pp. 151–152, 2019.

[25] R. G. de A. Azevedo, N. Birkbeck, I. Janatra, B. Adsumilli, and P. Frossard, "A viewport-driven multi-metric fusion approach for 360-degree video quality assessment," in *IEEE ICME*, 2020.

[26] Y. Rai, P. Le Callet, and P. Guillotel, "Which saliency weighting for omni directional image quality assessment?," in *QoMEX*, 2017.

[27] Z. Hu, A. Bulling, S. Li, and G. Wang, "EHTask: Recognizing user tasks from eye and head movements in immersive virtual reality," *IEEE Trans. Visualization and Computer Graphics*, pp. 1–1, 2021.

[28] I. D. D. Curcio and S. Ahsan, "Viewport margins for 360-degree immersive video," in *IEEE MMSP*, 2020.

[29] M. N. Akcay, B. Kara, S. Ahsan, A. C. Begen, I. Curcio, and E. Aksu, "Head-motion-aware viewport margins for improving user experience in immersive video," in *ACM Multimedia Asia*, 2021.

[30] S. Ahsan, A. Hourunranta, I. D. D. Curcio, and E. Aksu, "FriSBE: Adaptive bit rate streaming of immersive tiled video," in *Packet Video Workshop*, 2020.

[31] R. Fielding and J. Reschke, "Hypertext transfer protocol (HTTP/1.1): Message syntax and routing." [Online] Available: `https://datatracker.ietf.org/doc/html/rfc7230`, 2014. Accessed on Apr. 1, 2022.

[32] M. Belshe, R. Peon, and M. Thomson, "Hypertext transfer protocol version 2 (HTTP/2)." [Online] Available: `https://datatracker.ietf.org/doc/html/rfc7540`, 2015. Accessed on Apr. 1, 2022.

[33] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *IEEE INFOCOM*, 2016.

[34] M. Hostetter, D. A. Kranz, C. Seed, C. Terman, and S. Ward, "Curl: a gentle slope language for the web.," *World Wide Web Journal*, vol. 2, no. 2, pp. 121–134, 1997.

[35] Nokia Technologies, "Omnidirectional MediA Format (OMAF)." [Online] Available: `https://github.com/nokiatech/omaf`. Accessed on Apr. 1, 2022.

[36] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *ACM MMSys*, 2017.

[37] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: Open-source HEVC/H.265 encoder," in *ACM Multimedia*, 2016.

[38] M. N. Akcay, B. Kara, S. Ahsan, A. C. Begen, I. Curcio, and E. Aksu, "Rate-adaptive streaming of 360-degree videos with head-motion-aware viewport margins," in *IEEE MIPR*, 2022.

VITA

Burak Kara graduated from Muğla Science High School in June 2015. He received the degree of Bachelor of Science in Computer Science from Özyeğin University in June 2020. His senior design project, HTTP Adaptive Streaming over Multiple Network Interfaces, was published in ACM MMSys'20. He has been pursuing a Master of Science degree in Computer Science since October 2020 at Özyeğin University.

In his M.Sc. study, he worked on 360-degree video and low latency streaming on a joint project between Nokia and Özyeğin University.

PUBLICATIONS

- Mehmet N. Akcay, Burak Kara, Saba Ahsan, Ali C. Begen, Igor Curcio, and Emre Aksu. 2021. "Head-motion-aware viewport margins for improving user experience in immersive video." In ACM Multimedia Asia (MMAsia '21). Article 43, 1–5. https://doi.org/10.1145/3469877.3490573

- M. N. Akcay, B. Kara, S. Ahsan, A. C. Begen, I. Curcio, and E. Aksu, "Rate-adaptive streaming of 360-degree videos with head-motion-aware viewport margins." In IEEE Conf. Multimedia Information Processing and Retrieval (MIPR '22). https://doi.org/10.1109/MIPR54900.2022.00056

- B. Kara, M. N. Akcay, A. C. Begen, S. Ahsan, I. Curcio, and E. Aksu, "Motion awareness in quality assessment of 360-degree videos," in IEEE MultiMedia, 2022. (Submitted, Major Revision)