Perceived Quality driven VR Streaming

Paper # XXX, XXX pages

ABSTRACT

In recent years, Virtual Reality (VR) video streaming becomes popular quickly. How to provide high user-perceived quality with limited bandwidth becomes the biggest problem in VR video streaming. Traditional viewpoint-driven VR streaming makes the assumption that perceived quality is only related to viewpoint-object distance, so it simply allocates high bitrate to objects near user's viewpoint. However, In this paper we prove that user perceived quality is not only related to viewpoint-object distance, it is also related to background luminance, texture complexity, viewpoint moving speed and Depth-of-Field. With consideration of these insights, we can save x% bandwidth while providing the same perceived quality.

To build our PQVRS, we build a model to measure perceived quality in VR display, and design an object-based tiling scheme to cut video into several tiles which can be independently encoded / decoded. Then we present our method to enable client-side perceived quality computation and optimization. We implement a prototype of PQVRS and compare its performance with state-of-art VR streaming schemes. Experimental results show that proposed PQVRS saves x% bandwidth without decrease of perceived quality.

1 INTRODUCTION

In recent years, the world is becoming more virtual than we ever thought it would be. Many video service providers, such as YouTube, roll out Virtual Reality (VR) videos which provide immersive experience to users. While consuming VR videos, users can change their viewpoint, resulting in an interactive experience than consuming traditional videos with a fixed viewing direction. However, VR videos' high demand of resolution and bitrates hinder their wide spread over the Internet. How to provide high user-perceived quality with limited bandwidth becomes the biggest problem in VR video streaming.

Viewpoint-adaptive streaming is regarded as a promising way to solve the problem. It assumes that a object's user-perceived quality depends on the distance from it to user's viewpoint[5]. It allocates high bit-rate for objects near user's viewport, and allocates low bit-rate for objects far from user's viewport.

However, viewpoint-adaptive streaming is a very coarse approximation of user-perceived quality, since user-perceived quality is not only related to object-viewpoint distance. Many prior works state that it is related to some human visual characteristics, such as luminance[7], [20], texture complexity[7]. We model these visual characteristics into our potential improvement evaluation, and result shows that we can save 30% bandwidth without decrease of user perceived quality. Moreover, our user study shows that in VR video display, user-perceived quality is also related to viewpoint moving speed and Depth-of-Field (DoF). When we take consideration of these visual characteristics, we can further save 20% bandwidth without decrease of user perceived quality.

Although these insights about user-perceived quality give us promising potential improvement, optimizing perceived quality in real VR streaming system is challenging in three aspects:

- **Challenge 1: Traditional Human Visual System (HVS) models can not measure perceived quality in VR video display.** Perceived quality can be well-measured in traditional video display by modeling human visual system. However, perceived quality in VR display is very different from traditional video display. So current models can not be applied. We need to build a new model to quantify perceived quality in VR display.
- **Challenge 2: Traditional grid-like video tiling scheme performs poorly in perceived quality optimization.** To allocate different bitrate to different objects, we need to cut the video into several spatial tiles which can be independently encoded / decoded. In traditional grid-like tiling scheme, videos are cut into m*n rectangular tiles of equal size. However, in a coarse-grained tiling, performance of quality allocation is poor. In a fine-grained tiling, serious bitrate efficiency problem occurs in video encoding. Both of them make the performance of perceived quality optimization far from the optimal value.
- **Challenge 3: Information needed for perceived quality computation is disparted on server-side and client-side.** Perceived quality depends on both video content and user behavior, so it can not be precomputed as some traditional quality metrics (such as PSNR, MSE, SSE). Client needs to compute the perceived quality of each bitrate allocation and then make decision in realtime. However, to get information of video content with current DASH, client has to prerequest the information of each pixel from server. This communication overload even exceed the overload of actual video streaming.

In this work, we address technical challenges above, then present the design and implementation of Perceived Quality driven VR Streaming (PQVRS). PQVRS is built on three key insights:

- *Perceived quality model in VR display can be built by simply adding several new factors on traditional perceived quality model.* Although there are several new factors which influence perceived quality in VR display, and the correlation between them may be complex, we find that their influence to perceived quality can be considered independently. So we only need to measure the perceived quality due to each of new insights, and then add them to current perceived quality model.
- *Video tiling strategy can be optimized on server side.* Since traditional grid-like tiling method fail to exactly catch objects in videos, it limits the performance of perceived quality driven rate allocation. Tiling video by objects can solve this problem. Moreover, information of objects is only related to videos, object-based tiling can be completed in advance on server-side.
- *Perceived quality computation can be decoupled.* Although server has no information about user behavior, it can still compute perceived quality with each possible user behavior situations offline. Then server sends the results to client, client combine the computation result and actual user behavior to obtain final perceived quality.

Although accurately computing perceived quality needs the value of each pixel of each frame, it can be well-approximated on client-side using much less information.

Taken together, these insights enable us to engineer a Perceived Quality driven VR Streaming (PQVRS). In PQVRS, we decouple the visual characteristics due to video objects and user viewpoint, and build the VR streaming system which consider visual characteristics of video objects completely on server-side, and considers visual characteristics of user viewpoint pattern completely on client-side.

We implemented a prototype of PQVRS. We ran a pilot study on one content provider with x sessions. Our experiments show that PQVRS can save 45% bandwidth compared with state-of-art VR streaming solutions, without decrease of perceived quality. In real-world adaptive streaming with unstable throughput, we obtain significantly higher user-perceived quality without introducing stalling. (Fig. 1)

Contributions and Roadmap:

- Identifying key visual characteristics which influence user-perceived quality in VR display and estimate potential improvement for it. (§2)
- Identifying key challenges of building a practical perceived-quality-driven VR video streaming system and present our solutions. (§3-6)

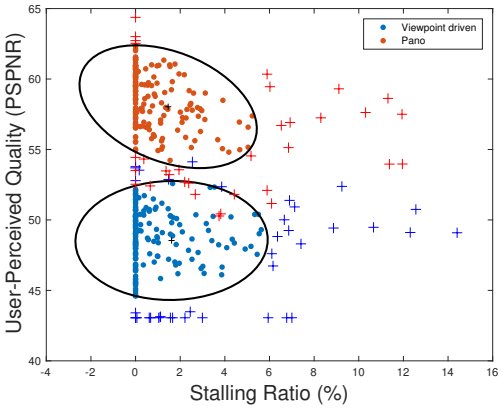


Figure 1: Performance comparison of Pano v.s. traditional viewpoint-driven video streaming under average throughput of 1Mbps, with bandwidth fluctuation. Each plot indicates a video session. We present 75% ellipse confidence area.

- Real-world evaluation that demonstrates substantial performance improvement by PQVRS (§7-8).

2 MOTIVATION

In the field of psychology and biology, it is widely accepted that in a video display, user-perceived quality is very different from original video quality. It is mainly caused by human visual system [12] [2] and human brain signal processing system [25].

In this section we show that current rate allocation strategies in VR streaming is far from optimizing user-perceived quality. If we can rightly allocate bitrate for each part of video to maximize user-perceived quality, we can save 50% bandwidth in the same user-perceived quality, compared with state-of-art methods.

2.1 Current VR streaming solutions

In this section we briefly introduce two widely-used VR streaming solutions: monolithic streaming and viewport-driven streaming.

2.1.1 Monolithic streaming. Monolithic streaming is a widely used scheme in many VR video providers (e.g. IQiyi, Youku, ...). In monolithic streaming, it is assumed that user-perceived quality is rightly equal to video quality. A VR video is considered the same as a traditional non-VR video, so each spatial part of video is allocated to one same bitrate level. Client chooses a proper bitrate for the whole video chunk according to bandwidth throughput, buffer or other factors.

Monolithic streaming's biggest advantage is easy deployment. Since it deliveries VR video as non-VR video, there is

no need to change the video streaming protocol. It can be directly applied on current video streaming system.

The disadvantage of monolithic streaming is obvious. Since in a VR video display, most area of video is out of user's viewport, which are absolutely not viewed by user. Monolithic streaming allocate the same quality level for both in-viewport part and out-viewport part. So it causes serious waste of bandwidth.

2.1.2 Viewpoint-driven streaming. Many user studies have proved that perceived quality of an object in a video is highly related to its distance to user viewpoint. When an object is near to user viewpoint, it should be allocated high bitrate since user is sensitive to its distortion. When an object is far from user viewpoint, it should be allocated low bitrate since distortion is difficult to be noticed.

Viewpoint-driven streaming ([3], [8], [9], [11], [15], [24], [17]) allocates bitrate for each object in a video based on its distance to user viewpoint. Video is usually cut into several spatial tiles, which can be independently encoded / decoded. Then it can decide the bitrate in tile-level granularity.

In addition, viewpoint prediction technology is needed in viewpoint-driven streaming, since we need to predict user's viewpoint in the next few seconds. Fortunately, in recent years, viewpoint prediction have been well studied and many solutions can obtain high prediction accuracy ([3], [18], [17]).

Compared with monolithic streaming, viewpoint-driven streaming can allocate more bitrate for area near the user's viewpoint so it can save a lot of bandwidth while maintaining the same user-perceived quality.

2.2 What influences user-perceived quality

In viewpoint-driven streaming, there is a strong assumption that user-perceived quality is only related to viewpoint-object distance. Although viewpoint-driven streaming saves much bandwidth compared with monolithic streaming, it is still far away from user-perceived quality optimization.

In fact, in a video display, user-perceived quality is also related to many other factors:

- *Perceived quality is highly related to content luminance.* For a very dark content or a very bright content, user is more difficult to notice the content distortion, thus perceived quality is higher. On the other hand, when a content is of medium lightness, user is more sensitive to its distortion. So the same level distortion causes lower perceived quality.
- *Perceived quality is also related to texture complexity.* The same degree distortion is more likely to be noticed in a simple texture than in a complex texture. So perceived quality is higher in complex texture. This phenomenon is called texture masking.

Moreover, in VR video display, perceived quality is related to some new factors, which are not considered in traditional video display:

- *Viewpoint moving speed can influence perceived quality.* One of the most highlighted feature of VR video is that users can freely move their viewpoints. According to our data analysis, more than x% time, user's viewpoint is moving faster than y deg/s. When user moves his / her viewpoint, perceived quality is significantly improved, since user is unable to detect the distortion. [22]
- *Depth of Field (DoF) can influence perceived quality.* In VR display, different objects have different DoF and it is simulated by binocular parallax. Objects with small DoF (which are near to user) have greater parallax, and greater parallax leads to difficulty of binocular fusion, thus human's ability of detecting distortion is weaker. [4], [10]
- *User's light / dark adaptation can influence perceived quality.* When user wears a HMD in VR display, environmental brightness perceived by eyes is totally depended on luminance of video content itself. So when the scene changes dramatically from dark to light or from light to dark, user's ability of detecting distortion will be weaker for a period of time. [14]

2.3 Challenges

In order to achieve perceived quality driven VR streaming, the core challenge is: perceived quality is related to both video content and user behavior, how to take together both-sides information to maximize the perceived quality.

For specifically, there are three aspects:

Challenge 1: Traditional Human Visual System (HVS) models can not measure perceived quality in VR video display.

Perceived quality can be well-measured in traditional video display by building mathematical model from luminance, texture complexity and viewpoint-object distance to perceived quality. However, as we have mentioned, besides these existing factors, perceived quality in VR display is related to several new factors which are never modeled before. Their mathematical relationship to perceived quality, and how these new factors combined with existing factors together influence perceived quality are unknown problems.

Challenge 2: Traditional grid-like video tiling scheme performs poorly in perceived quality optimization.

To optimize perceived quality, we need to independently allocate bitrate of each spatial part of the video. Grid-like tiling scheme is a widely used method to solve the problem. Video is cut into several rectangular tiles with equal size

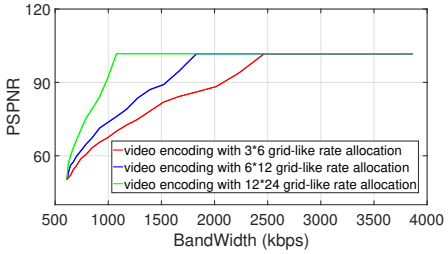


Figure 2: PSPNR-bandwidth tradeoff in video encoding with different granularity of rate allocation.

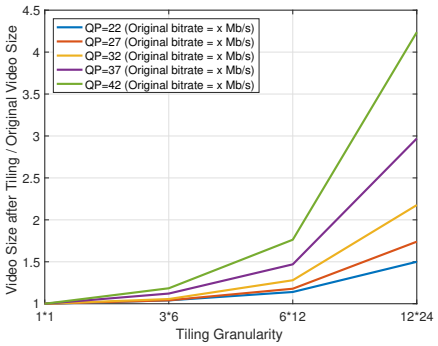


Figure 3: The ratio of video size after tiling and original video size in each tiling granularity and each bitrate level. We notice that fine-grained tiling introduces serious bitrate efficiency problem, especially in low bandwidth situations in which the overall bitrate of each tile is low.

which can be independently encoded / decoded. So we can allocate different bitrate to different tiles.

However, traditional grid-like tiling performs poorly in perceived quality optimization in two aspects:

(1) Coarse-grained tiling causes coarse-grained rate allocation, and perceived quality obtained by coarse-grained rate allocation is far from that obtained by fine-grained rate allocation. We encode the video with different rate allocation granularity. We apply PSPNR to measure perceived quality and Fig. 2 shows the PSPNR-bandwidth tradeoff. Rate allocation with 3*6 / 6*12 granularity obtains -x% / -x% PSPNR compared with 12*24 granularity, this is a significant performance gap in perceived quality optimization.

(2) Fine-grained tiling introduces serious bitrate efficiency problem in video encoding. In practice, each tile has to be encoded independently instead of encoded together. When we cut the video into tiles, the total video size is increased. Fig. 3 shows the video size of different tiling granularity compared with original video size. We find that cutting a video into 12*24 tiles introduces 50% to 330% additional video size compared to original video. This significantly lower its practical performance in perceived quality optimization, especially in low bandwidth situations where the bitrate efficiency problem is very serious.

Challenge 3: Information needed for perceived quality computation is disparted on server-side and client-side.

To optimize perceived quality, client needs to compute the perceived quality of each bitrate allocation and then make decision. However, different from video quality which only related to video content itself, perceived quality depends on both video content and user behavior. Information of video content is located on server-side while information of user behavior is located on client-side. To get information of video content with current DASH, client has to pre-request in information of each pixel from server. This communication overload even exceed the overload of actual video streaming.

2.4 Potential improvement

As we have stated that perceived quality is not only related to object-viewpoint distance, it is also related to many factors, we need to clarify how much potential improvement can we get by taking above insights.

We set up a trace-driven experiment to evaluate this potential improvement. PSPNR [7] is applied as a metric to measure perceived quality. Real traces are collected from over 800 VR displays of 48 users. In our experiment, each video has 5 different bitrate level. We compare the performance of 3 methods:

- *Viewpoint-driven VR streaming.* Video is cut into 6*12 spatial rectangular tiles. Tiles on user's viewpoint is allocated the high bitrate. For other tiles, bitrate is allocated linearly decreasing with its distance to user's viewpoint.
- *Perceived quality driven VR streaming.* Suppose we can freely allocate bitrate to each spatial part of video without video tiling. With consideration of user viewpoint, content luminance and texture complexity, we do adaptive streaming to maximize user-perceived quality.
- *Perceived quality driven VR streaming. (with 3 VR factors)* Suppose we can freely allocate bitrate to each spatial part of video without video tiling. Besides user viewpoint, content luminance and texture complexity, we also take consideration of viewpoint moving speed, content Depth-of-Field and light / dark adaptation, we do adaptive streaming to maximize user-perceived quality.

Fig. 4 shows the PSPNR-bandwidth tradeoff of 3 strategies.
Key observations:

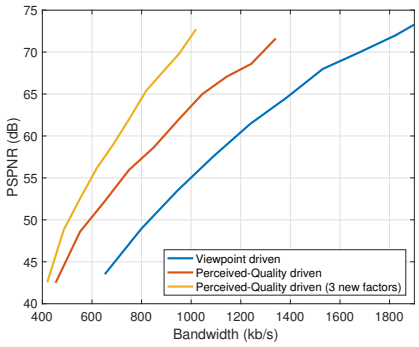


Figure 4: PSPNR-bandwidth tradeoff of (1) Viewpoint-driven VR streaming. (2) Perceived quality driven VR streaming. (3) Perceived quality driven VR streaming. (with 3 VR factors)

- With consideration of content luminance and texture complexity, perceived-quality-driven VR streaming can save 30% bandwidth compared with viewpoint-driven VR streaming providing the same PSPNR. Moreover, when we take consideration of viewpoint moving speed, object Depth-of-Field and light / dark adaptation, we can further save 20% bandwidth.

3 KEY INSIGHTS AND IDEAS

As stated in previous section, the biggest challenge of optimizing user-perceived quality is that perceived quality is related to both video content and user behavior. To solve the challenge, we show that perceived quality optimization can be decoupled: we can optimize video-content-related factors on server-side and optimize user-behavior-related factors on client-side.

3.1 Insights

In this section, we show that in VR streaming, each visual factor influences user-perceived quality independently. So we can decouple them into several single factors to measure perceived quality respectively, then we design an server-side offline tiling scheme only with consideration of video content, and design a client-side bitrate allocation method only with consideration of user behavior.

Insight 1: Perceived quality model in VR display can be built by simply adding several new factors on traditional perceived quality model.

In traditional video display, there are many factors which influence user-perceived quality. Prior works have studied how single factors influence user-perceived quality. Moreover, [5] proves that, in perceived quality computation, several factors (luminance, texture complexity, etc.) can be considered independently. As a result, we can independently measure the influence of each factor as a coefficient, and multiply them together to obtain the final user-perceived quality.

However, in VR display, there are some new factors which are never considered in traditional video display. So how these new factors, together with factors in traditional video display, influence the perceived quality in VR streaming, is an unknown problem. We intend to assume that they can also be computed independently, and then multiplied as coefficients together with traditional factors to obtain final user-perceived quality.

To prove our guess, we set up a user study to evaluate how perceived quality is influenced by multiple factors. (Sec. 4) Our result shows that perceived quality model in VR display can be decoupled. We only need to simply add new factors to traditional perceived quality model.

Insight 2: Video tiling strategy can be optimized on server side.

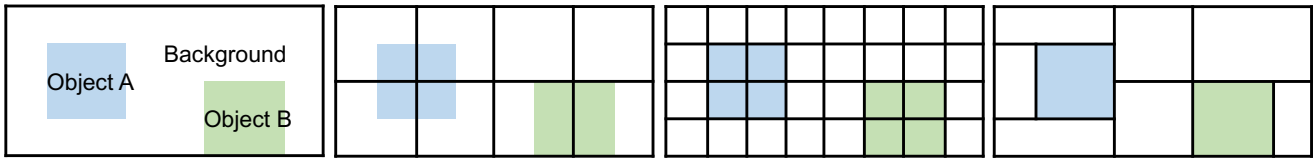
In traditional grid-like video tiling scheme, video is cut into m*n rectangular tiles with equal size.

This tiling strategy has an obvious drawback: at most times the boundary between objects are not exactly on the boundary between two adjacent tiles. Fig. 5 shows an example. In the video frame, there are two objects (Object A, Object B) and a background. In coarse-grained tiling (Fig. 5(b)), the object can not be exactly caught by one or several tiles. It leads to bitrate allocation of low performance. In fine-grained tiling (Fig. 5(c)), although the object can be caught by several tiles, it cut the video into too many tiles and many cutting lines are unnecessary. It leads to serious bandwidth efficiency problem as described in Sec. 2.3.

Suppose we can know information about video content, we can cut the video based on objects like Fig. 5(d). In this method, we can obtain similar tiling granularity as Fig. 5(c) but remain similar tile numbers as Fig. 5(b). We think tiling scheme like Fig. 5(d) can outperform traditional grid-like tiling scheme. Moreover, since information of objects is only related to video content, it is totally independent from user. So object-based tiling can be completed offline on server-side.

Insight 3: Perceived quality computation can be decoupled.

Although accurately computing perceived quality needs the information of both video content and actual user behavior, we can precompute perceived quality of given video content with each possible user behavior situations on server-side, without any information of actual user behavior, and transform the computation result to client. Client only needs to choose the situation which match actual user behavior most, then uses given perceived quality value to approximate the actual perceived quality. Experimental results prove that we can obtain very good approximation accuracy with very



(a) A video frame. (b) Grid-like tiling in coarse granularity. (c) Grid-like tiling in fine granularity. (d) Object-based tiling.

Figure 5: An example of traditional grid-like tiling and object-based tiling.

little end-to-end exchanging overhead of computation result transforming.

3.2 Key ideas to solve the challenges

Based on above insights, we decouple the visual characteristics due to video content and user viewpoint, and design a perceived quality driven VR streaming system which consider visual characteristics of video objects completely on server-side, and considers visual characteristics of user behavior completely on client-side. Specifically, we build our system based on following ideas:

- *Measuring perceived quality in VR streaming by incrementally adding some new coefficients into traditional non-VR perceived quality model.* (§4)
- *Server-side offline video tiling only based on of video content.* (§5)
- *Server-side perceived quality precomputation and client-side approximation & optimization.* (§6)

4 PERCEIVED QUALITY MEASUREMENT

In this section we first introduce Peak Signal-to-Perceptible-Noise Ratio (PSPNR) [7] which is used as metric for perceived quality measurement. Then, since PSPNR computation needs the Just-Noticeable-Distortion (JND) [6] value of each pixel, we present our JND definition based on human visual system in VR display.

4.1 Perceived quality measuring by Just-Noticeable Distortion (JND) model

Since user-perceived quality is a subjective thing, how to measure it becomes a problem. For example, it is very hard for a user to say the perceived quality of video A is 2 times better than that of video B, or it is just 1.7 times better.

Peak Signal-to-Perceptible-Noise Ratio (PSPNR) [7] is an widely-accepted metric to measure perceived quality. It is defined based on Just-Noticeable-Distortion (JND) theory [6]. Given the original video frame and a compressed video frame, user can notice their difference only if the difference between them is above a visibility threshold. When the different is under this threshold, it will not detected by user. This threshold is called Just-Noticeable-Distortion.

Based on JND theory, PSPNR is defined by accumulating error of each pixel which can be detected by user:

$$PSPNR = 20 \times \log_{10} \frac{255}{\sqrt{PMSE}} \tag{1}$$

$$PMSE = E\{[|p(x,y) - \hat{p}(x,y)| - JND(x,y)]^2 \times \Delta(x,y)\} \tag{2}$$

$$\Delta(x,y) = \begin{cases} 1, & |p(x,y) - \hat{p}(x,y)| > JND(x,y) \\ 0, & |p(x,y) - \hat{p}(x,y)| \le JND(x,y) \end{cases} \tag{3}$$

where $p(x,y)$ and $\hat{p}(x,y)$ are value of pixel (x,y) in original video frame and compressed video frame. $JND(x,y)$ is the visibility threshold of pixel (x,y).

Compared with PSNR (which is widely used in evaluating video / image quality), the core difference of PSPNR is introducing a term $JND(x,y)$ for each pixel (x,y). So how to compute JND for each pixel (x,y) is an important issue. We will present our JND computation in the following section.

4.2 Building Just-Noticeable-Distortion Model for VR display

The just-noticeable distortion (JND) provides cues for measuring the visibility of the Human Vision System (HVS). JND refers to the maximum distortion which cannot be perceived by human. It describes the perceptual redundancy of the picture by providing the visibility threshold. The JND model generally exploits the visibility of the minimally perceptible distortion.

JND has been well-studied in traditional video display since 1995. The most basic and solid research is about the relationship between content luminance and JND. [20] and [7] prove that content with moderate content luminance have low JND value, while content with very high or very low content luminance have high JND value. (Fig. 6)

In recent years, based on the insight of content luminance - JND relationship, researchers have explored some more

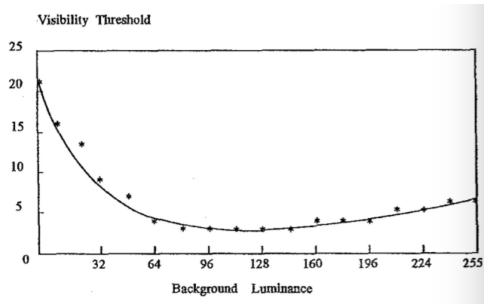


Figure 6: JND due to content luminance.

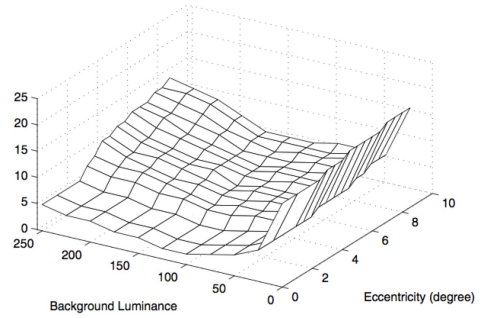


Figure 7: JND due to content luminance & viewpoint-object distance.

factors which are also related to JND, such as texture complexity, viewpoint-object distance ([7], [5]) and some other factors. Although the relationship between multiple JND factors may be complex, for simplicity, we can decouple them into several single factors. For example, [5] set up a user study to test the JND value with the combined effect of content luminance and viewpoint-object distance. The result proves that viewpoint-object distance can be considered independently from content luminance (Fig. 7). It can be regarded as a coefficient which can be directly multiplied on JND value computed by content luminance. Based on this insight, JND computation for multiple factors is decoupled into several single factors. This significantly simplify the JND computation.

In VR display, user experience is very different because 3 new factors should be taken into consideration: (1) viewpoint moving decreases visual acuity, (2) low Depth-of-Field decreases visual acuity and (3) light / dark adaptation decreases visual acuity. So previous JND model for traditional video display can not be directly applied on VR video display. Moreover, it is unknown that (1) how these new factors influence JND with combined effect of old factors, and (2) if they can also be decoupled into several single factors, like viewpoint-object distance [5].

Strictly answering these 2 questions needs to test JND value for each possible combinations of all factors (including all possible luminance, texture, viewpoint-object distance, viewpoint moving speed, Depth-of-Field and light / dark adaptation, which make up a 6-dimension feature space), which is not practical for a real-world user study. For simplicity, we imitate the user study in [5], which makes only 2 factors variable and other 4 factors fixed, then we can know how is the combined effect of the 2 factors to JND, and if they can be decoupled to 2 single factors which independently influence JND.

So we present 3 user studies: (1) JND v.s. luminance & viewpoint moving speed. (2) JND v.s. luminance & depth of field. (3) JND v.s. luminance & light / dark adaptation. We evaluate the effect of 3 VR-only factors to JND, each with combined effect of luminance, since luminance is the most well-studied JND factor.

We conduct the user study using real Head Mounted Device (HMD). Parameters of proposed HMD are listed as Table 1:

Table 1: HMD Parameters

Equipment	Oculus GO
CPU	Xiaolong 821 customized drive Edition
Memory	3GB
Screen Resolution	2560 × 1440
Refresh Rate	72Hz
Fixed pupil distance	63.5mm

20 subjects participated in the experiments. All of them were in their twenties. The subjects obtained extensive practice during the experiments.

In this paper, we imitate the human visibility experiments in [7]. In the experiment, a small square area, 32 x 32 pixels, is located in the center of a flat field of constant grey level (luminance). For each possible grey level (luminance) of the square area, the noises of fixed amplitude are randomly added to or subtracted from the pixels within the square area. Through varying the amplitude of the noise, the visibility threshold for each grey level is determined when the square region contaminated by the noises is just noticeable.

Experimental settings of our 3 user studies are:

1. **JND v.s. luminance & viewpoint moving speed** Set a background with constant grey level and set the 32 * 32 square with 5 different grey levels: 30, 80, 130, 180 and 230. The square is doing uniform rectilinear motion on the background. Noises are randomly added to or subtracted from it. We test the visibility threshold in 2 conditions: (1) Observer's viewpoint is gazed on a fixed point and the square moves across the fixed point. (set the moving speed of 32*32 square

area to 2 deg /s, 4 deg / s, 6 deg / s, 8 deg / s, 10 deg / s) (2) Observer's viewpoint is allowed to tracking the moving object. (set the moving speed of 32*32 square area to 5 deg /s, 10 deg / s, 20 deg / s, 40 deg / s, 80 deg / s)

2. **JND v.s. luminance & depth of field** Set a background with constant grey level and set the 32 * 32 square with 5 different grey levels: 30, 80, 130, 180 and 230. Then set the depth-of-field of the square to 1m, 1.5m, 3m, 50m. Testing the visibility threshold for these situations.

3. **JND v.s. luminance & light / dark adaptation** Before our test, we show subjects a pure background with gray level L_A (L_A = 30, 80, 130, 180, 230) for 20 seconds. Then we change the background gray level to L_B (L_B = 30, 80, 130, 180, 230) and show subjects the 32*32 square. Testing the visibility threshold for these situations.

As described above, our user study contains multiple independent repetitive experiments. But we have stated that perceived quality is related to human visual system and human brain (§2), and human brain can be trained during the experiment. So long-time continuous experiment may make subjective more and more skillful, which leads to elimination of JND difference in different situation, and decreasing value of JND during the process of experiment. To eliminate the impair of this, we (1) dispart the experiment into 3 days, with only 5 minutes for each day. (2) random the test order of each single test.

4.3 Results

4.3.1 JND v.s. luminance & viewpoint moving speed. One of the most highlighted feature of VR video is that users can freely move their viewpoints. According to our data analysis, more than 30% viewpoints are moving faster than 20 deg/s. (Fig. 8) When human viewpoint is moving, visual acuity is decreased in 2 different conditions: (1) if user is tracking an object, visual acuity decreases smoothly with viewpoint moving speed increases. (2) If user is not tracking an object, visual acuity decreases dramatically with viewpoint moving speed increases. [22]

We first analyze tracking condition. Fig. 9 shows $JND^{track}_{lum\&speed}(l, v)$, the combined effect of viewpoint moving speed v and content luminance l to JND. We notice that effect of v to $JND^{track}_{lum\&speed}$ is very similar in different l, so this combined effect can be decoupled into 2 parts:

$$JND^{track}_{lum\&speed}(l, v) = f_{lum}(l) \times f_{track}(v) \quad (4)$$

where $f_{lum}(l)$ represents the visibility threshold with only consideration of content luminance, and $f_{track}(v)$ is a coefficient which represents influence of viewpoint moving speed on visibility threshold.

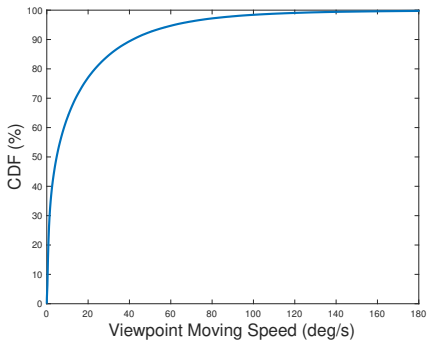


Figure 8: CDF gram of viewpoint moving speed.

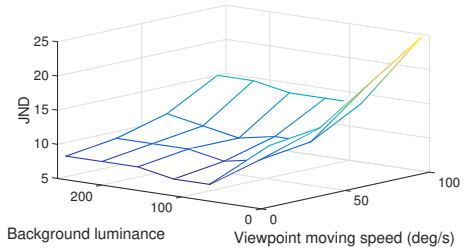


Figure 9: JND due to content luminance and viewpoint moving speed.

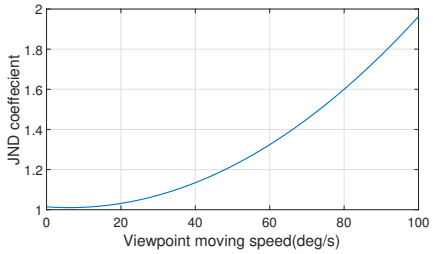


Figure 10: $f_{track}(v)$, the JND coefficient of viewpoint moving speed.

$f_{lum}(l)$ has been presented as Fig. 6. According to curve fitting, we can obtain $f_{track}(v) = \frac{JND^{track}_{lum\&speed}(l,v)}{f_{lum}(l)}$ as Fig. 10.

With similar method, we obtain result of no-tracking condition, $f_{notrack}(v)$, as Fig. 11.

In the real-world, viewpoint moving speed is not given, it needs to be predicted by client-side. Since human action has great randomness, viewpoint moving speed may fluctuate dramatically. It is very difficult to accurately predict user's viewpoint moving speed even in the near future (1s - 5s). Fortunately, since viewpoint moving speed has strong temporal locality, we can easily predict a lower bound of

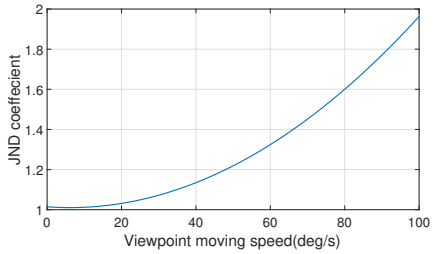


Figure 11: $f_{notrack}(v)$, the JND coefficient of viewpoint moving speed.

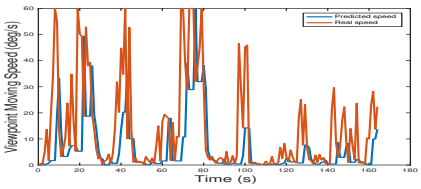


Figure 12: An example of real viewpoint moving pattern and predicted viewpoint moving pattern.

viewpoint moving speed in the near future. Fig. 12 shows an example of real-world viewpoint moving speed and our predicted viewpoint moving speed by a very simple method. In most of the time, predicted speed does not exceed real speed. So we will not locate the video quality lower than it should be. §8 proves that this speed prediction is already enough to cover most of the gain.

4.3.2 JND v.s. luminance & depth of field. Depth-of-field (DoF) refers to the 3D distance from the object to human eyes. In real world, human detect object's DoF by parallax effect (Fig. 13). When a human is focusing on an object, the object's location is different in his left eye and right eye. Then human brain combines 2 different 2D images from 2 eyes and build a 3D image with Depth-of-Field. Plenty of works [] state that DoF influences visual acuity in 2 aspects: (1) Human visibility is weak when focusing on object with small DoF, since combining 2 highly different image from left / right eye is hard and slow. (2) Human visual acuity is weak when focus distance is different from the object's DoF (eg. there are object A and B in a same place but with different DoF, when human is focusing on A, his visual acuity for B is weak).

In non-VR video displays, the screen is bioptic, where only a single display is presented that is viewed by both eyes, so all contents have the same DoF (in this situation the DoF is exactly the distance from eyes to screen) However, in most VR video displays, the screen is stereoscopic, where the illusion of depth is created by delivering images rendered from different angles to each eye. So left eye and right eye

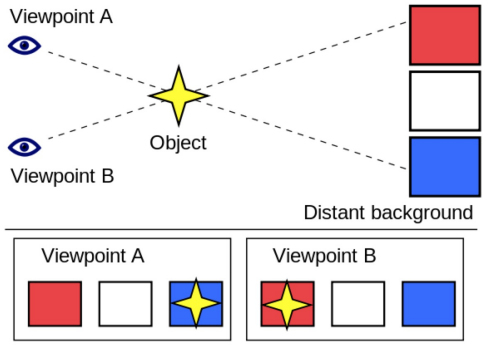


Figure 13: An example of parallax effect.

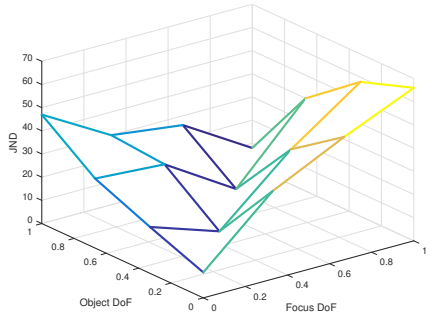


Figure 14: JND due to focus Depth-of-Field and object Depth-of-Field.

receives different image. As a result, in VR display, contents have different DoF.

Fig. 14 shows $JND(l, D_o, D_f)$, the JND value for constant luminance = 127, with object's DoF D_o and user's focus distance D_f. According to this result, we can verify 2 conclusion above: JND is greater when object has small DoF or object has DoF different from DoF user is focusing on.

To prove the influence of DoF can be considered independently as a coefficient, Fig. 15 shows $JND(l, D_o)$, the combined effect of object DoF D_o and content luminance l to JND, with $D_f = D_o$. We notice that influence of D_o to JND is very similar in different content luminance. Similarly, we also check whether D_f can be considered independent from luminance l in the same method, and the answer is yes. So the mapping from l, D_o and D_f to JND can be decoupled into 2 parts:

$$JND_{lum\&DoF}(l, D_o, D_f) = f_{lum}(l) \times f_{DoF}(D_o, D_f) \quad (5)$$

where $f(_{lum}(l))$ represents the visibility threshold with only consideration of content luminance, and $f_{DoF}(D_o, D_f)$

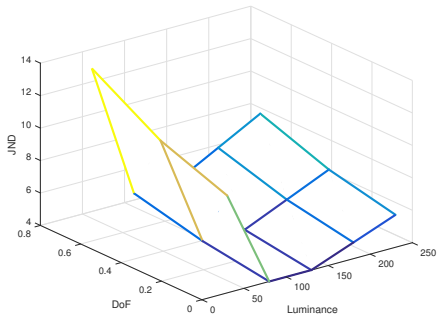


Figure 15: JND due to content luminance and object Depth-of-Field.

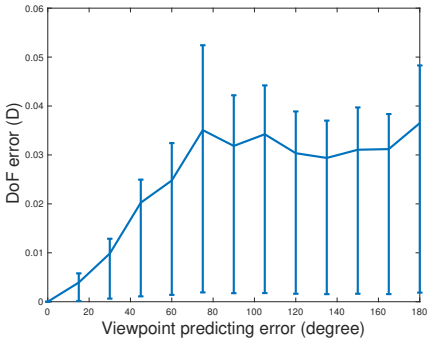


Figure 17: Viewpoint prediction error v.s. DoF error of D_f. We present the average error, along with the upper bound and the lower bound of 75% cases (around the average case).

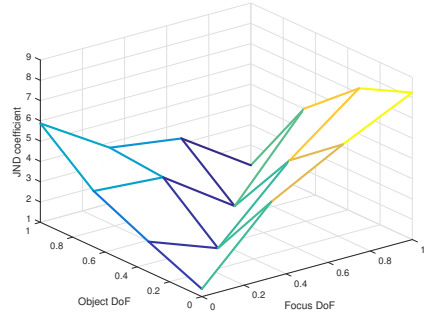


Figure 16: $f_{DoF}(D_o, D_f)$, the JND coefficient of object / focus Depth-of-Field.

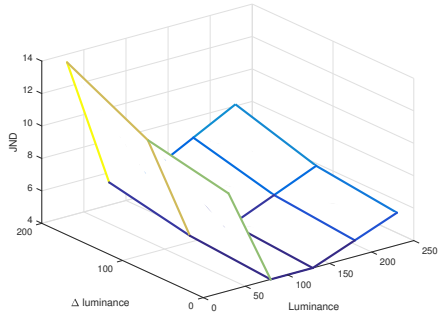


Figure 18: JND due to content luminance and light / dark adaptation.

is a coefficient which represents DoF's influence on visibility threshold.

$f_{lum}(l)$ has been presented as Fig. 6. According to curve fitting, we can obtain $f_{DoF}(D_o, D_f) = \frac{JND_{lum\&DoF}(l, D_o, D_f)}{f_{lum}(l)}$ as Fig. 16.

Be similar to viewpoint moving speed in the last subsection, since there is unenviable error in viewpoint prediction, DoF of user focus D_f may be also inaccurate (while D_o is always accurate because it is only video content's property). Fortunately, we analyze the spatial locality of DoF in 50 videos, Fig. 17 shows that when the viewpoint predicting error is less than 30 deg, the average error of DoF is below 0.01, and in 87.5% cases, the error is below 0.013. This viewpoint predicting accuracy can be obtained by current linear regression method.

4.3.3 JND v.s. luminance & light / dark adaptation. In human vision system, in order to transit from day to night vision they must undergo a dark adaptation period in which each eye adjusts from a high luminescence setting to a low luminescence setting. ([13], [16]) Similarly, when human eyes transit from night to day vision they also undergo a light adaptation period. It is well-known that during the process of light / dark adaptation, human visual acuity decreases.

In non-VR video displays, the environment illumination totally depends on the real environment (e.g. under the sunlight, or in a classroom with electric lamp, or in a dark room). However, VR video displays are very different. When user wears HMD, the environment illumination totally depends on video content itself. So when the illumination of video content changes dramatically, eyes need a period of time to adapt the new illumination.

Fig. 18 shows $JND_{lum\&ada}(l, \Delta e)$, the combined effect of light / dark adaptation and content luminance to JND. l represents content luminance, Δe represents the variation of environment luminance before the test and during the test.

Be similar to above 2 results, this combined effect can also be decoupled into 2 parts:

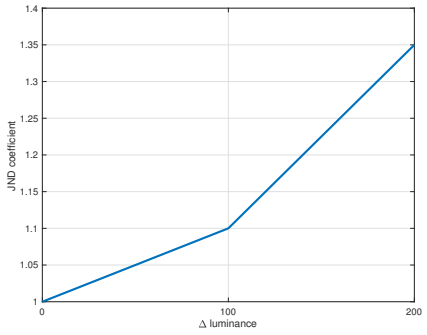


Figure 19: $f_{adapt}(\Delta e)$, the JND coefficient of light / dark adaptation.

$$JND_{lum\&ada}(l, \Delta e) = f_{lum}(l) \times f_{adapt}(\Delta e) \quad (6)$$

where $f(_{lum}(l))$ represents the visibility threshold with only consideration of content luminance, and $f_{adapt}(\Delta e)$ is a coefficient which represents DoF's influence on visibility threshold.

And we obtain $f_{adapt}(\Delta e) = \frac{JND_{lum\&ada}(l,\Delta e)}{f_{lum}(l)}$ as Fig. 19.

4.3.4 Put it together. To get the final JND value, we need to put traditional JND factors together with above three new VR-only JND factors.

Table 2 lists the symbol and definition used for JND computation.

Table 2: symbol and definition used for JND computation

symbol	Definition	Source
$f_{lum}(l)$	Visibility threshold due to content luminance l.	[7], [20]
$f_{text}(t)$	Visibility threshold due to texture complexity t.	[7]
$f_{dist}(d)$	The coefficient of viewpoint-object distance t.	[5]
$f_{track}(v)$, $f_{notrack}(v)$	The coefficient of viewpoint moving speed v.	This paper
$f_{DoF}(D)$	The coefficient of Depth-of-Field D.	This paper
$f_{adapt}(\Delta e)$	The coefficient of light / dark adaptation Δe.	This paper

For the first 3 JND factors which have been well-studied in traditional video display, measurement of $f_{lum}(l)$ can be found in [20], measurement of $f_{text}(t)$ can be found in [7] and measurement of $f_{text}(t)$ can be found in [5]. We leave the detail of their computation out of this paper.

It is widely accepted [7] that in non-VR video display, JND_{nonVR} can be computed as follow:

$$JND_{nonVR} = \max\{f_{lum}(l), f_{text}(t)\} \times f_{dist}(d) \quad (7)$$

Since we have proved that in VR display, 3 new VR-only JND factors, viewpoint moving speed, DoF, light / dark adaptation can be regarded as coefficient on non-VR JND value, JND_{VR} can be computed as:

$$JND_{VR} = \max\{f_{lum}(l), f_{text}(t)\} \times f_{dist}(d) \times f_{track/notrack}(v) \times f_{DoF}(D \quad (8)$$

5 OBJECT-BASED TILING SCHEME

In this section we first introduce our object detection method. Based on this object detection, we describe our object-based tiling scheme and make comparison with state-of-art grid-like tiling schemes.

5.1 Object Detection based on Quality-Bitrate Efficiency

In the field of computer vision, there are many object detection algorithms which can cut a video into different content objects. However, in our video tiling scheme, the purpose of object detection is little different. Actually we do not care if two adjacent tiles contain exactly the same object semantically. The most important thing is the probability of them to be allocated the same bitrate level by client. Even though two adjacent objects are different, if they have similar properties (like luminance, contrast, Depth of Field), we can also contain them in one tile because there is high probability for user to allocate them the same bitrate level.

To meet our purpose, for any rectangular tile which can be independently encoded, we define its **Quality-Bitrate Efficiency (QBE)** as follow:

$$QBE = \frac{PSPNR_{highest} - PSPNR_{lowest}}{B_{highest} - B_{lowest}} \quad (9)$$

where $PSPNR_{highest}$ / $PSPNR_{lowest}$ denotes the PSPNR value of this tile's highest / lowest bitrate version, and $B_{highest}$ / B_{lowest} denotes the bitrate of this tile's highest / lowest bitrate version. In order to eliminate the influence for PSPNR by different user viewpoint positions, we only consider luminance, texture complexity and Depth-of-Field in QBE computation. These informations can be obtained completely on server-side.

In a video frame, different objects have different QBE. QBE is highly related to properties of content objects. According to PSPNR computation (Section 4), objects with very dark / light object luminance, complex texture or high depth of field, usually has higher QBE.

In perceived quality optimization, it is obvious that objects with high QBE is more likely to be allocated high bitrate (because it can improve more PSPNR in the same bandwidth cost), and objects with low QBE is more likely to be allocated low bitrate. So when two adjacent objects have similar QBE, they have high probability to be allocated the same bitrate level.

5.2 Tiling video by objects

Based on above insights, we aim to cut the video into T rectangular tiles of unequal size, such that content items within the same tile have similar QBE.

In this paper, tiling video by objects consists of 3 steps:

Step 1: Partitioning the original video into 12*24 rectangular basic units of equal size.

Basic unit is the smallest unit of proposed object-based tiling. In the tiling process, each tile must be composed by one or several entire basic units which form a rectangular shape.

Step 2: Computing QBE of each basic unit.

We get QBE of each basic unit according to (9). After that, suppose a tile t consists of N_t basic units $u_1, u_2, \dots, u_{N_t}$, we can compute its QBE Variance ($QBEV_t$) as follow:

$$QBEV_t = \frac{\sum_{1 \le i \le N_t} (QBE_{u_i} - E(QBE_{u_i}))^2}{N_t} \tag{10}$$

where $E(QBEu_i)$ is the average QBE of basic units in tile t:

$$E(QBE_{u_i}) = \frac{\sum_{1 \le i \le N_t} QBE_{u_i}}{N_t} \tag{11}$$

A tile with high QBEV means visual properties (e.g. luminance, texture complexity) of objects in this tile have very different properties. A tile with low QBEV means objects in this tile are similar. So a good tiling scheme should keep each tile's QBEV value as low as possible.

Step 3: Merging all basic units into T rectangular tiles, such that their weighted average QBEV is minimal.

Suppose V is the whole video frame. R_i is the ith rectangular tile and S_i is its area.This optimization problem can be formalized as following:

$$\begin{aligned} &\min \sum_{i=1}^{T} QBEV_i S_i \\ &\text{s.t.} \bigcup_{i=1}^{N} R_i = V \\ &R_i \bigcap R_j = \emptyset \qquad \forall 1 \le i, j \le T \end{aligned} \tag{12}$$

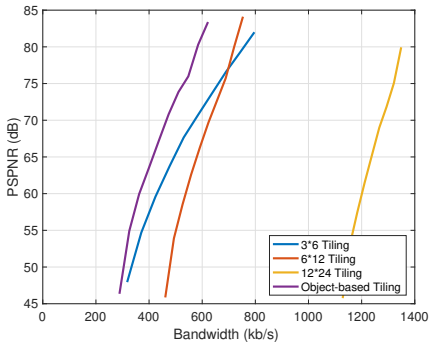


Figure 20: The PSPNR-bandwidth tradeoff of proposed Object-based Tiling scheme and traditional grid-like tiling scheme (3*6, 6*12 and 12*24).

However, this optimization problem is a NP-hard problem, so we can not get the optimal solution in polynomial time. In practice, we apply a dynamic programming algorithm to get the suboptimal solution for this problem. In our implementation, we set $T = 72$ since it performs well in most situations.

5.3 Comparison of object-based tiling and grid-like tiling

We evaluate our tiling scheme and make comparison with traditional grid-like tiling schemes.

Fig. 20 shows the PSPNR-bandwidth tradeoff of 3*6 grid tiling, 6*12 grid tiling, 12*24 grid tiling and proposed object-based tiling.

For traditional gird-like tiling schemes, the performance of different tiling granularity depends on bandwidth. In low bandwidth, most part of video is allocated the lowest bitrate level. So there is no need to cut the video into many tiles. 3*6 tiling performs well because of its high bitrate efficiency. However, in high bandwidth, a coarse tiling granularity causes suboptimal bitrate allocation, so it is beaten by 6*12 tiling. Unfortunately, 12*24 tiling performs poorly in all bandwidth because of its serious bitrate efficiency problem.

Proposed object-based tiling scheme beats traditional grid-like tiling scheme of all bandwidth. In high bandwidth situation, it saves 20% bandwidth compared with 6*12 grid-like tiling. Although 3*6 tiling's low bandwidth performance is near to proposed object-based tiling, its high bandwidth performance is far away from object-based tiling.

6 CLIENT-SIDE PSPNR COMPUTATION AND OPTIMIZATION

In client-side PSPNR optimization, client first need to compute PSPNR value for each rate allocation. However, PSPNR

computation needs information of both video content and user behavior. In practice, client is unaware of video content before actually receiving video content. How to decouple PSPNR computation process to let client-side get PSPNR value without knowing information of video content is a big challenge.

In this section we first introduce our client-side PSPNR computation, and then describe our strategy of bitrate allocation to optimize PSPNR.

To optimize PSPNR in client-side, we first present an approximation of PSPNR computation, then we describe our algorithm of client-side PSPNR optimization.

6.1 Client-side PSPNR computation

First we decouple JND into Content JND (CJND) and Behavior JND (BJND). We define a pixel (x, y)'s Content JND (CJND) as:

$$CJND(x,y) = \max\{f_{lum}(l(x,y)), f_{text}(t(x,y))\} \times f_{DoF}(D(x,y)) \tag{13}$$

CJND is only related to video content. It can be pre-computed on server-side.

We define Behavior JND (BJND) as:

$$BJND(x,y) = f_{dist}(d(x,y)) \times f_{speed}(v) \times f_{adapt}(\Delta e) \tag{14}$$

BJND is only related to client behavior, it is totally independent from video content so it can be obtain on client-side.

Based on (13) and (14), equation (1) can be rewritten as:

$$PSPNR = 20 \times \log_{10} \frac{255}{\sqrt{PMSE}} \tag{15}$$

$$PMSE = E\{[|p(x,y) - \hat{p}(x,y)| - CJND(x,y) \times BJND(x,y)]^2 \times \Delta(x,y)\} \tag{16}$$

Given a tile, although the CJND value of different pixel may vary greatly, we notice that BJND value of different pixels are not so different. This is because:

(1) All pixels in one tile share the same $f_{speed}(v)$ and $f_{adapt}(\Delta e)$.

(2) Although strictly, $f_{dist}(d(x,y))$ is different for different pixel (x, y), pixels in the same tile is usually spatial adjacent. Their distance to user viewpoint is roughly the same, so they usually have similar $f_{dist}(d(x,y))$ value.

Based on above insight, we use only one BJND value in one tile, instead of one BJND value for each pixel. We suppose (x_{mid}, y_{mid}) is the pixel in the center of the tile, then we can approximate (16) as:

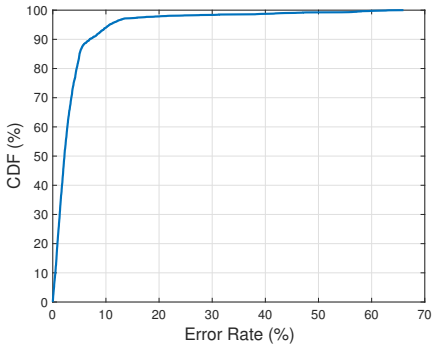


Figure 21: CDF of PSPNR approximation accuracy.

$$PMSE \approx E\{[|p(x,y) - \hat{p}(x,y)| - CJND(x,y) \times BJND(x_{mid}, y_{mid})]^2 \times \Delta($$ (17)

Now, in (17), server has all information (including $p(x, y)$, $\hat{p}(x, y)$, $CJND(x, y)$, $\Delta(x, y)$) with only exception of one single value: $BJND(x_{mid}, y_{mid})$. Moreover, according to definition of PSPNR, PSPNR is strictly monotonically increasing with $BJND(x_{mid}, y_{mid})$. So in practice, server can presuppose N different BJNDs offline to compute N corresponding PSPNRs by (17), then transforms these PSPNR values to client-side. Client-side computes the actual $BJND(x_{mid}, y_{mid})$ value and choose the nearest presupposed BJND, then obtain the corresponding PSPNR value.

In real-world implementation, in order to further improve PSPNR approximation accuracy, when the actual $BJND(x_{mid}, y_{mid})$ is in the gap between 2 adjacent presupposed BJNDs, we simply compute the actual PSPNR by linear regression. Fig. 21 shows the approximation accuracy. In 88% situations, error rate of proposed method is less than 5%. In realtime VR video streaming, PSPNRs of presupposed BJNDs are packaged in MPD file which can be downloaded by client, their bandwidth cost is negligible.

6.2 PSPNR optimization

Client-side PSPNR optimization consists of 2 levels:

Level 1: In constraint bandwidth consumption and buffer size, allocate the bitrate of each temporal segment.

Level 2: For each temporal segment, after its bitrate is allocated, distribute the bitrate to each spatial tile.

6.2.1 *Level 1: Allocating bitrate of each temporal segment.*

Suppose a VR video consists of K temporal segments: s_1 , s_2, ... , s_K. For segment s_k, when it is allocated R_k bandwidth, it can obtain $q_k(R_k)$ PSPNR. Our problem is to allocate bitrate $R_1, R_2, \ldots, R_K$ for each temporal segment.

When the client is allocating bitrate for video segment s_k, suppose the buffer status is B_k and the bandwidth throughput

prediction is C_k. Moreover, the previous video segment s_{k-1} has been allocated the bitrate R_{i-1}, thus its perceived quality is $q_{k-1}(R_{k-1})$. In rate adaptation logic, making decision of R_k should take consideration of these values ([21], [23]) to avoid rebuffing and quality fluctuation.

Then the rate allocation problem can be formalized as:

$$R_k = f(q_{k-1}(R_{k-1}), B_k, C_k) \tag{18}$$

Fortunately, BOLA (Near-Optimal Bitrate Adaptation for Online Videos) [21] is a well-known method to solve the problem completely on client-side, and it has been well built in Dashjs. Given the value of R_{i-1}, B_i, C_i, the algorithm output an appropriate R_i. So BOLA can be directly applied on the Level 1 rate allocation. We leave the detail of BOLA design out of this paper since there are plenty of works about BOLA design in rate adaptation.

6.2.2 Level 2: Allocating bitrate of each spatial tiles. In above Level 1 rate allocation, two questions are remained: (1) Given rate allocation R_k for temporal segment s_k, how to allocate the bitrate to each tile in segment s_k. (2) How to build $q_k(R_k)$, the mapping from bitrate to PSPNR.

Actually, the second question can be merged into the first question: For a temporal segment s_k, when the bitrate of each tile is determined according to R_k, its PSPNR $q_k(R_k)$ is determined. So the problem of Level 2 rate allocation is how to distribute R_k bitrate to each tile to optimize $q_k(R_k)$.

According to (15), PSPNR is monotonous decreasing with PMSE. Maximizing PSPNR is equivalent to minimizing PMSE.

Suppose we separate the video segment s_k into N_k spatial tiles, numbered 1, 2, ..., N_k. They are encoded into M quality levels with different bitrate. We define $r_{i,j}$ as bitrate of chunk i with level j. We define the optimal quality level for i-th tile as l_i and define PMSE value of tile i with jth bitrate as $p_{i,j}$.

Therefore, at each adaptation step, the client needs to solve the following optimization problem to minimizing PMSE:

$$\begin{aligned} \min_{l: l_i \in [1,M]} \quad & \sum_{i=1}^{N} p_{i,l_i} \\ \text{s.t.} \quad & \sum_{i=1}^{N} r_{i,l_i} \le R_k W. \end{aligned} \tag{19}$$

This optimization problem can be solved as a Multiple-Choice Knapsack problem. A brute force search which exhaustively evaluates all combinations guarantees an optimal solution. However, the computational complexity is $O(N^M)$. To reduce the computation time, we use a dynamic programming method (algorithm 1) to approximate the problem where the computational complexity is $O(R_k MN)$.

Algorithm 1 PSPNR driven Rate Allocation Algorithm.

Require: Throughput bound, R_k; Number of tiles, N_k; Number of rates, M; Rate set, $\{r_{i,j}\}$; PMSE set, $\{p_{i,j}\}$;
Ensure: Allocation rate levels set, $\{l_i\}$;

```
Initialize knapsack revenue table K ∈ R^{(N_k+1)×R_k} by 0;
Construct the prefix table P ∈ Z^{(N+1)×R_k};
for i from 1 to N_k do
    for b ∈ [0, B] do
        K_{i,b} = min_{j∈[1,M]}{K_{i-1,b-r_{i-1,j}} + p_{i-1,j}};
        P_{i,b} = arg min_{j∈[1,M]}{P_{i-1,b-r_{i-1,j}} + p_{i-1,j}};
    end for
end for
Find R̂_k = arg min_{b∈[0,R_k]}{K_{N_k,b}};
for i from N_k to 1 do
    l_i = P_{i,R̂_k};
    R̂_k = R̂_k − r_{i,l_i};
end for
return {l_i}
```

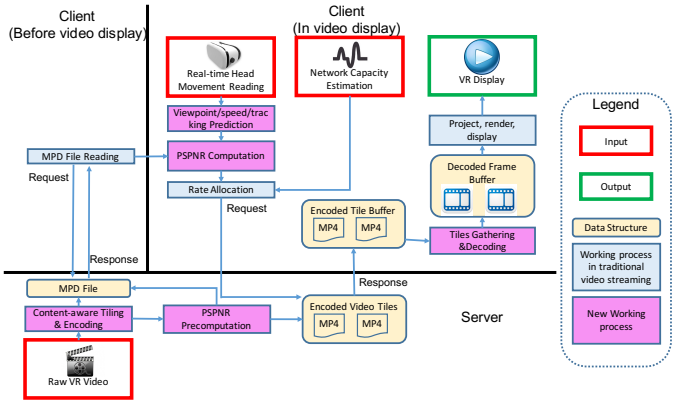


Figure 22: Workflow of PQVRS.

7 IMPLEMENTATION

PQVRS is open source (10K lines of code across C++, Matlab, Java, and python) and can be accessed at [1]. Implementation of PQVRS is shown in Fig. 22. Compared with traditional video streaming system, several modules are newly added or modified, which are highlighted by pink color.

7.1 Server-Side

Content-Aware Tiling / Encoding: Based on our tiling algorithm described in Sec. 5, we cut each video segment into $N = 72$ rectangular tiles with different size. Each tile is encoded independently. The information of tiling result is packeted into MPD file so that client is aware of how each segment is cut. In addition, with consideration of client-side decoding overhead, we encode the lowest quality version

of whole video without tiling, called "Base Layer". For each video segment, we first transfer the Base Layer to client, and then transfer tiles only in user's viewport with corresponding quality. This can significantly reduce the number of tiles decoded on client-side (from 72 to around 12 per video segment), thus make PQVRS applicable on real-world devices.

PSPNR Precomputation: Based on equation (x), server tries to compute PSPNR of each tile for 5 different BJNDs, then packet the result into MPD file. Compared with video content, size of this additional information is negligible.

7.2 Client-Side

Viewpoint / Speed / Tracking prediction: In our system, viewpoint prediction is achieved by linear regression because of its efficiency and robustness. According to result of viewpoint prediction, we can estimate user's viewpoint moving speed by differentiating adjacent user viewpoints. Moreover, since we have analyzed object traces of video on server-side by YOLO [19] and deliver the brief description to client-side, client-side can match the predictive viewpoint traces and object traces in order to predict if user is going to tracking an object.

PSPNR Computation: On client-side, based on our viewpoint / speed / tracking prediction, the value of BJND can be obtained by equation (x). Then client read MPD file to get the PSPNR results with different BJNDs, and choose the one with the nearest BJND.

Tiles Gathering & Decoding: For each video segment, client first decode the Base Layer. For tiles in user's viewport, we set up a multi-process decoder based on ffmpeg, and set up a buffer to cache the decoded frames of tiles to be played in the future. Specifically, the decoding scheduler dynamically selects a received tile and sends it to an idle decoder. The decoded frames are not necessarily consumed right away; instead they can be stored in the buffer residing in the video memory. When a cached frame is needed during the playback, its corresponding tiles are fed into GPU, then then mapping them from 2D to 3D for rendering on Head-Mounted Device.

8 PERFORMANCE EVALUATION

In this section, we show that:

- In the network with fixed bandwidth, compared with state-of-the-art VR streaming system, PQVRS saves 45% bandwidth on average while providing the same PSPNR, or provides 15-20 higher PSPNR in the same bandwidth.
- In the network with unstable bandwidth, PQVRS improves 10 higher PSPNR and decreases 50% stalling.

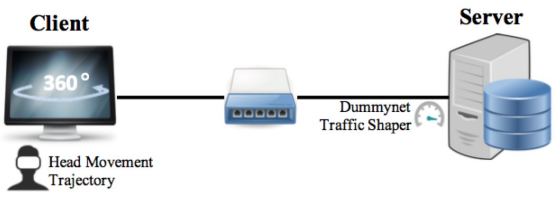


Figure 23: Network topology.

- In real-world perceived quality rating by users, PQVRS can obtain notable higher user rating.
- PQVRS introduce less CPU workload than state-of-art VR streaming system.

8.1 Evaluation Setup

To evaluate the performance, we implement PQVRS, a real-world VR streaming system which adaptively choose the quality of each part of video and display on HMD. Fig. 23 shows the network topology in the experiment, which consists of a client and a server.

In the experiments, we choose 50 videos with different types and lengths. We set the duration of one video segment as 1 second. To generate different quality videos, we use quantization parameter (QP) ranging from 22 to 42 in steps of five leading to five different bitrate versions.

In our experiment, 5 solutions are chosen to make comparison:

- *Flare [17]:* A state-of-the-art viewpoint-driven VR streaming. Video is cut into 4*6 spatial rectangular tiles. Tiles on user's viewpoint is allocated the high bitrate. For other tiles, bitrate is allocated linearly decreasing with its distance to user's viewpoint.
- *PQVRS (old JND model):* Video is cut into 6*12 spatial rectangular tiles. With consideration of user viewpoint, content luminance and texture complexity, we do adaptive streaming to maximize user-perceived quality.
- *PQVRS+ (new JND model):* Video is cut into 6*12 spatial rectangular tiles. Besides user viewpoint, content luminance and texture complexity, we also take consideration of viewpoint moving speed, content Depth-of-Field and light / dark adaptation, and do adaptive streaming to maximize user-perceived quality.
- *PQVRS++ (new JND model + new tiling):* Our final solution in this paper. Based on our new JND model, our new tiling method is also applied to further improve the performance.
- *Theoretical performance:* Upper bound of perceived quality-driven VR streaming, assuming that client-side always predict viewpoint, viewpoint moving speed correctly. Video can be encoded in real-time, which

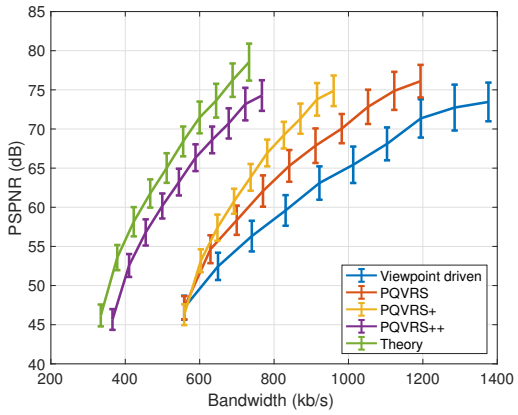


Figure 24: PSPNR-bandwidth tradeoff of 5 methods: Viewpoint-driven, PQVRS, PQVRS+, PQVRS++, theoretical performance.

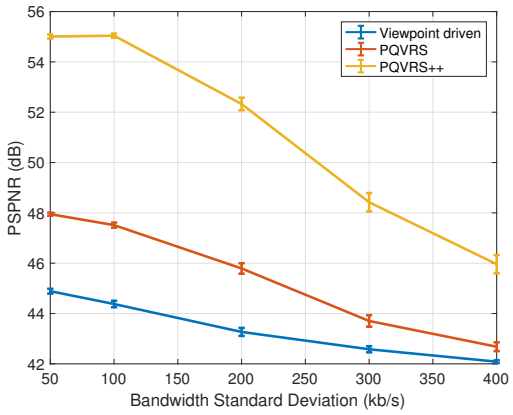


Figure 25: PSPNR of 5 methods: Viewpoint-driven, PQVRS, PQVRS+, PQVRS++, theoretical performance.

allocates different quality in different part without cutting video to tiles.

8.2 Performance under network with different bandwidth stability

In the real world, VR streaming may happen in all kinds of networks, such as Ethernet, Wifi and LTE. Although they share the same application layer protocol, the stability of bandwidth of these networks are very different (due to different packet loss rate, jitter in transport layer). A good VR streaming protocol should be robust on network with different bandwidth stability. So in this section, we evaluate the performance of Flare and PQVRS on both fixed bandwidth and real-world bandwidth with different bandwidth stability.

8.2.1 Performance comparison under fixed bandwidth. When network is under a fixed bandwidth, video streaming is stalling-free since the requested video segment will always be retrieved back to client on time. User perceived quality is almost only related to PSPNR. So we test the PSPNR-bandwidth tradeoff for above methods under fixed bandwidth. Fig. 24 shows the result. Compared with Flare, PQVRS++ saves 45% bandwidth on average while providing the same PSPNR, or provides 15-20 higher PSPNR in the same bandwidth.

8.2.2 Performance comparison under real-world bandwidth. Actually, in real-world video streaming, bandwidth is unstable which may change significantly in a short time. When the bandwidth change dramatically from a high level to a low level, it not only decrease the value of PSPNR, but also causes stalling in video display, which is another important metric of perceived quality.

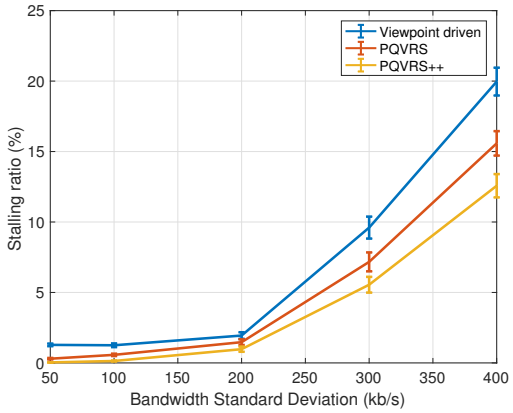


Figure 26: Stalling of 5 methods: Viewpoint-driven, PQVRS, PQVRS+, PQVRS++, theoretical performance.

We test the performance of Flare and proposed PQVRS under the network with 800 kb/s bandwidth on average, but with different levels of bandwidth fluctuation.

Fig. 25 presents average PSPNR of 3 methods, and Fig. 26 presents average time of stalling of 3 methods.Result show that PQVRS++ improves 10 higher PSPNR and decreases 50% stalling.

8.3 Performance robustness under random user behavior

In VR streaming, any system which takes advantage of viewpoint prediction or other user behavior prediction, will definitely suffer from the risk of prediction error due to randomness of user behavior. Although in most cases, user behavior (viewpoint, speed) is predictable, however, when user behavior is unpredictable, a robust streaming system should also

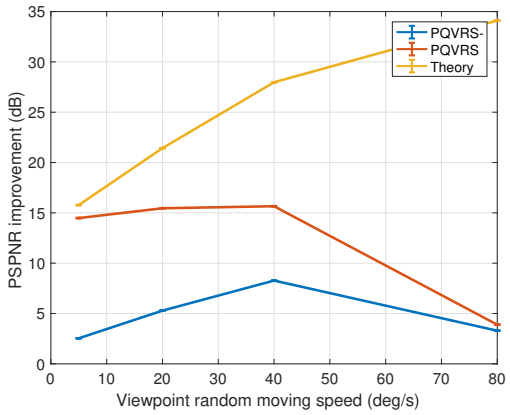


Figure 27: PSPNR improvement of PQVRS, PQVRS+, PQVRS++ and theoretical cases on random viewpoint traces, compared with viewpoint-driven VR streaming.

present an acceptable performance, which will not give user a seriously bad perceived quality.

We generate a database of randomly moving user viewpoint, with different average moving speed. In our generated database, user's viewpoint is moving with a random speed to a random direction in each second, so it is completely unpredictable. Fig. 27 shows our improvement compared with Viewpoint-driven method on these viewpoint traces.

In this random, unpredictable user behavior, when viewpoint moving speed is below 20 deg/s, PQVRS++ can still reach a good improvement. Of course, in the worst case, when the speed get large enough, the predicted viewport will be totally wrong. PQVRS++ has nearly no improvement compared with viewpoint driven VR streaming since they both allocate high video quality on absolutely wrong place. But Fig. 27 shows that in any case, PQVRS++ does not perform poorly than it.

8.4 Real-world user rating

Based on our real-world VR streaming system, we compare above methods by real-world user rating.

20 subjects participated in the experiments. For each subject, we shown him (her) a same VR video which is streaming by Viewpoint-driven, PQVRS and PQVRS++ in the same network condition. The order of display by 3 methods is random. After 3 displays, the object needs to give a rating of perceived quality for them. The rules of rating is shown in Fig. 28.

Fig. 29 presents the result of user rating. Proposed PQVRS++ obtains score 4.2 on average, while PQVRS gets 2.7 and Flare gets only 1.4.

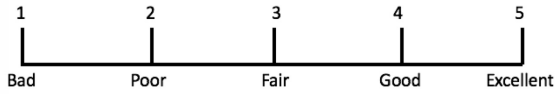


Figure 28: User rating rules.

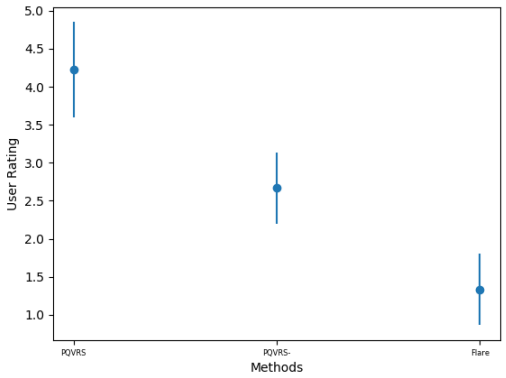


Figure 29: User rating results.

8.5 Analysis of VR streaming workload

We implement a prototype of Flare, PQVRS, and test their workload on Dell Precision 7920, to make comparison with proposed PQVRS++.

First we test client-side CPU workload. Although Dell Precision 7920 is a work station with 40 CPUs, now we only make 1 of them available, in order to simulate the performance on normal VR devices. Fig. 30 shows the comparison of client-side CPU occupancy rate between Flare, PQVRS and PQVRS++, when streaming a 2880*1440 VR video for 1 minute duration. We notice that 85% CPU workload comes from video decoding & rendering, and content downloading, perceived quality computation and quality allocation in PQVRS introduce negligible CPU workload. In total, PQVRS++ saves 5% CPU workload compared with Flare. This marginal benefit is mainly because we decreases the number of tiles need to be decoded by clients per segment, which makes decoder a bit faster. (as described in §7)

Server-side workload comparison is done with all 40 CPU available. Result is shown in Fig. 31. Since all modules on server-side are offline works, we test the CPU processing time for each single 1-minute VR video, rather than CPU occupancy rate. Results show that the dominant part of server-side processing time is video encoding. The processing time of PSPNR computation is negligible. So PQVRS++ doesn't introduce considerable workload for video server.

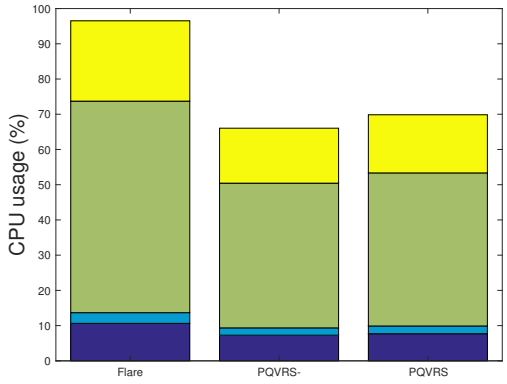


Figure 30: Client-side CPU workload comparison between Flare, PQVRS and PQVRS++.

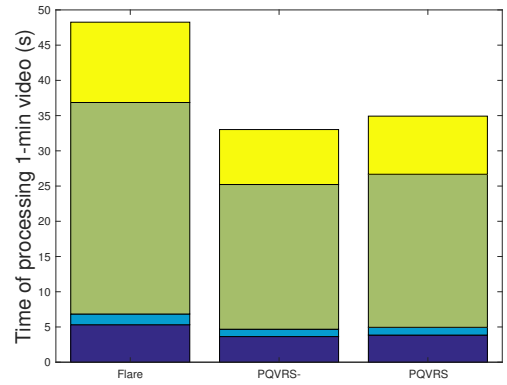


Figure 31: Server-side CPU workload comparison between Flare, PQVRS and PQVRS++.

9 RELATED WORKS

User Perceived Quality in Video Display:

In traditional non-VR video streaming, prior work has shown correlations between various visual characteristics and user perceived quality (e.g., users are sensitive to luminance, texture complexity, viewpoint-object distance), and built mathematical models (e.g., [5], [7], [25]). Our work focuses on exploring the new factors which influence user perceived quality in VR display which are never considered in traditional non-VR display, and then build an adaptive streaming system based on our insights.

VR Adaptive Streaming:

Monolithic streaming is the naive VR adaptive streaming method by streaming the entire 360-degree scene in constant quality without exploiting and optimizing the quality for the user's viewport, only choose different quality for different segments.

Since user's near-future viewpoint can be predicted in high accuracy, in order to save bandwidth, client chooses high quality for content in user's viewport and choose low quality for content out of user's viewport. There are many viewpoint-driven VR adaptive streaming works ([3], [8], [9], [11], [15], [24], [17]). Main difference between these adaptive streaming systems is that they choose different viewpoint prediction strategies, different rate allocation strategies, different tiling granularities and different multi-tiles decoding logics. But there are 2 common points: (1) quality allocation for each spatial part of the video is only based on the distance between content and viewpoint. (2) video is cut into tiles of equal size. Our proposed PQVRS has essential difference on these 2 points.

10 CONCLUSION

In this paper we find that in VR video display, user perceived quality is very different from video quality itself because it is related to many human visual characteristics, and optimizing perceived quality can bring out 50% bandwidth saving.

To solve the challenges of perceived quality measurement, bandwidth efficiency problem in video tiling, and client-side perceived quality real-time computation, we build PQVRS, a real-world VR streaming system. In PQVRS, we decouple perceived quality optimization to client-side and server-side, to build a perceived quality measurement model, a server-side video tiling algorithm and a client-side perceived quality real-time computation algorithm.

Experimental results show that PQVRS saves 45% bandwidth while providing the same perceived quality, compared with state-of-the-art VR streaming systems.

REFERENCES

[1] [n. d.]. ([n. d.]). https://github.com/ws826402696/VR_VideoStreaming_Project

[2] S.M. Anstis. 1974. A chart demonstrating variations in acuity with retinal position. *Vision Research* 14, 7 (1974), 589 – 592.

[3] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. 2017. Shooting a Moving Target: Motion-Prediction-Based Transmission for 360-Degree Videos. In *IEEE International Conference on Big Data*.

[4] K Carnegie and T Rhee. 2015. Reducing Visual Discomfort with HMDs Using Dynamic Depth of Field. *IEEE Computer Graphics and Applications* 35, 5 (2015), 34–41.

[5] Zhenzhong Chen and Christine Guillemot. 2009. Perception-oriented video coding based on foveated JND model. In *Picture Coding Symposium*. 1–4.

[6] Chun Hsien Chou and Chi Wei Chen. 1996. A perceptually optimized 3-D subband codec for video communication over wireless channels. *IEEE Transactions on Circuits and Systems for Video Technology* 6, 2 (1996), 143–156.

[7] Chun Hsien Chou and Yun Chin Li. 1995. Perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile.

IEEE Trans on Circuits and Systems for Video Technology 5, 6 (1995), 467–476.

[8] Vamsidhar Reddy Gaddam, Michael Riegler, Ragnhild Eg, Pal Halvorsen, and Carsten Griwodz. 2016. Tiling in Interactive Panoramic Video: Approaches and Evaluation. *IEEE Transactions on Multimedia* 18, 9 (2016), 1819–1831.

[9] Mario Graf, Christian Timmerer, and Christopher Mueller. 2017. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation. In *ACM on Multimedia Systems Conference*. 261–271.

[10] David M. Hoffman, Ahna R. Girshick, Kurt Akeley, and Martin S. Banks. 2008. VergenceâĂŞaccommodation conflicts hinder visual performance and cause visual fatigue. *Journal of Vision* 8, 3 (2008), 33.

[11] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer! (2016), 107–110.

[12] Gordon E. Legge and John M. Foley. 1980. Contrast masking in human vision. *J. Opt. Soc. Am.* 70, 12 (Dec 1980), 1458–1471.

[13] Richard A. Normann and Frank S. Werblin. 1974. Control of Retinal Sensitivity. *The Journal of General Physiology* 63, 1 (1974), 37–61.

[14] R A Normann and F S Werblin. 1974. Control of retinal sensitivity. I. Light and dark adaptation of vertebrate rods and cones. *Journal of General Physiology* 63, 1 (1974), 37–61.

[15] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *Proceedings of the 25th ACM International Conference on Multimedia (MM '17)*. ACM, New York, NY, USA, 306–314.

[16] E N Pugh. 1975. Rushton's paradox: rod dark adaptation after flash photolysis. *The Journal of Physiology* 248, 2 (1975), 413–431.

[17] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. ACM, New York, NY, USA, 99–114.

[18] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *The Workshop on All Things Cellular: Operations*. 1–6.

[19] J. Redmon and A. Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 00. 6517–6525.

[20] R. J. Safranek and J. D. Johnston. 1989. A perceptually tuned sub-band image coder with image dependent quantization and post-quantization data compression. In *International Conference on Acoustics, Speech, and Signal Processing*. 1945–1948 vol.3.

[21] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9.

[22] Joyce H. D. M Westerink and Kees Teunissen. 1995. Perceived sharpness in complex moving images. *Displays* 16, 2 (1995), 89–97.

[23] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *ACM Conference on Special Interest Group on Data Communication*. 325–338.

[24] Alireza Zare, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications. 601–605.

[25] Y. Zhao, L. Yu, Z. Chen, and C. Zhu. 2011. Video Quality Assessment Based on Measuring Perceptual Noise From Spatial and Temporal Perspectives. *IEEE Transactions on Circuits and Systems for Video Technology* 21, 12 (Dec 2011), 1890–1902.