Perceived Quality driven VR Streaming

Your N. Here
Your Institution

Second Name
Second Institution

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. 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[], texture complexity[]. 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 as basic rate allocation units, 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, objects in video are difficult to be exactly caught by one or several such equal-size tiles. 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 viewpoint, 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.

- *Tiling video by objects can be completed 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 may be a better solution. Moreover, information of objects is only related to videos, object-based tiling can be completed in advance on server-side.

- *Perceived quality computation and optimization can be completed on client-side.* 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 x% bandwidth compared with state-of-art VR streaming solutions, without decrease of perceived quality.

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)

- 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 [] and human brain signal processing system [].

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 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.

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. []

- *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.

- *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.

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: Current perceived quality measurement model is only for non-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 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

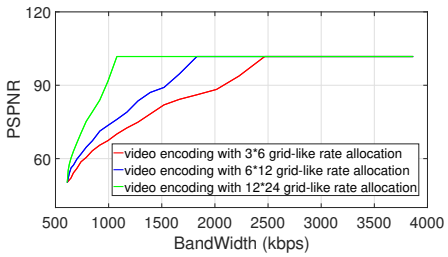


Figure 1: PSPNR-bandwidth tradeoff in video encoding with different granularity of rate allocation.

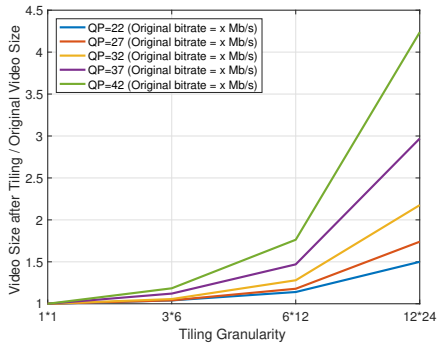


Figure 2: The ratio of video size after tiling and original video size in each tiling granularity and each bitrate level. We notice that fine-grained tiling introduces serious bitrate efficiency problem, especially in low bandwidth situations in which the overall bitrate of each tile is low.

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. 1 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. 2 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 viewpoint. Information of video content is located on server-side while information of user viewpoint 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, we need to clarify if this difference causes significant different rate allocation strategy in VR streaming. Moreover, we need to estimate how much improvement can we get by designing a better tiling scheme.

We set up a trace-driven experiment to evaluate this potential improvement. PSPNR[] 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 four methods:

- *Viewpoint-driven rate allocation.* 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 rate allocation.* Suppose we can freely allocate bitrate to each spatial part of video without video tiling. With consideration of luminance and texture complexity, we do bitrate allocation to maximize user-perceived quality.

- *Perceived quality driven rate allocation. (with 3 VR factors)* Suppose we can freely allocate bitrate to each spatial part of video without video tiling. Besides luminance and texture complexity, we also take consideration of viewpoint moving speed, object Depth-of-Field and light / dark adaptation, we do bitrate allocation to maximize user-perceived quality.

Fig. 3 shows the PSPNR-bandwidth tradeoff of 3 strategies.

Key observations:

- With consideration of content luminance and texture complexity, perceived-quality-driven rate allocation can save 30% bandwidth compared with viewpoint-driven rate allocation 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.

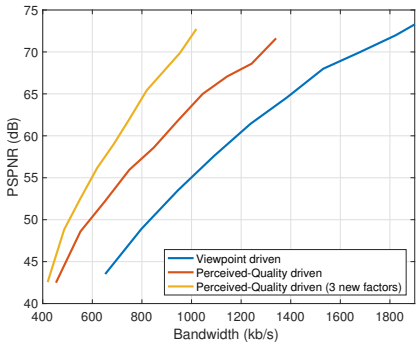


Figure 3: PSPNR-bandwidth tradeoff of (1) Viewpoint-driven rate allocation. (2) Perceived quality driven rate allocation. (3) Perceived quality driven rate allocation. (with 3 VR factors)

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. In this section we show that perceived quality can be optimized 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, visual characteristics of video objects and user viewpoint pattern influence user-perceived quality independently. So we can decouple them into two models to measure perceived quality, 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's viewpoint.

Insight 1: Perceived quality model in VR display can be built by simply adding new factors to 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, [] 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. 4 shows an example. In the video frame, there are two objects (Object A, Object B) and a background. In coarse-grained tiling (Fig. 4(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. 4(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.

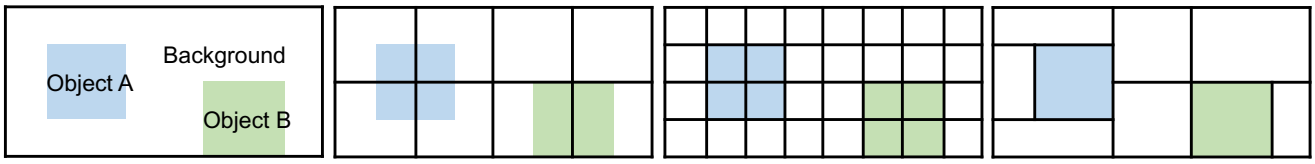
Suppose we can know information about video content, we can cut the video based on objects like Fig. 4(d). In this method, we can obtain similar tiling granularity as Fig. 4(c) but remain similar tile numbers as Fig. 4(b). We think tiling scheme like Fig. 4(d) can outperform traditional grid-like tiling 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 can be computed and optimized on client-side, with little end-to-end information exchanging overhead.

Although accurately computing perceived quality needs the value of each pixel of each frame, it can be well-approximated using much less information. Since values of pixels in the same tile have strong spatial correlation, we can approximate the value of perceived quality in tile level instead of pixel level. (Sec. 6) So we can compute each tile's overall Mean Square Error (MSE) and video-content-related JND factors offline on server-side, and transform them to client. Client use a simple formula to map it to perceived quality of each tile. Experimental results prove the accuracy of our approximation.

3.2 Key ideas to solve the challenges

Based on above insights, we decouple the visual characteristics due to video objects 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:



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

Figure 4: An example of traditional grid-like tiling and object-based tiling.

- *Measuring perceived quality in VR streaming by incrementally adding some new coefficients into traditional non-VR perceived quality model.*
- *Server-side offline video tiling only based on of video content.*
- *Client-side perceived quality computation and optimization.*

4 Perceived quality measurement

In this section we first introduce Peak Signal-to-Perceptible-Noise Ratio (PSPNR) [] which is used as metric for perceived quality measurement. Then, since PSPNR computation needs the Just-Noticeable-Distortion (JND) value of each pixel, we build a JND model for 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) [] is an widely-accepted metric to measure perceived quality. It is defined based on Just-Noticeable-Distortion (JND) theory. 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.

PSPNR is computed in pixel level:

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

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

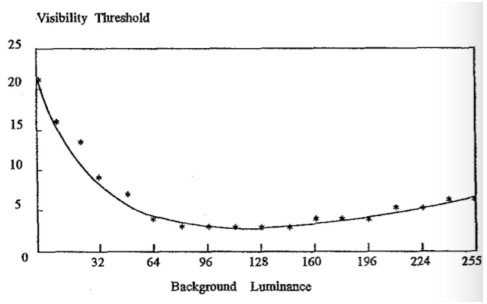


Figure 5: JND due to background luminance.

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

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 topic. We will present our JND model 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 HVS. JND refers to the maximum distortion which cannot be perceived by the human visual system. 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 model has been well-studied in traditional video display since 1995. The most basic and solid research is about the relationship between background luminance and JND. [] proves that objects with moderate background luminance have low JND value, while objects with high or low background luminance have high JND value. (Fig. 5)

In recent years, based on the effect of background luminance, researchers have explored some more factors which

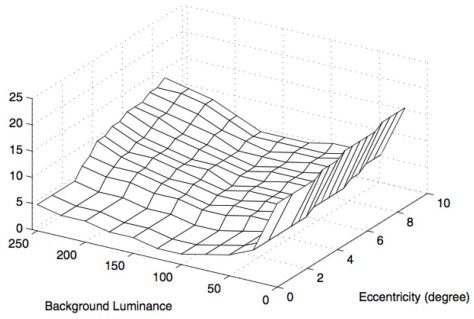


Figure 6: JND due to background luminance & viewpoint-object distance.

are also related to JND, such as texture complexity, temporal fluctuation, viewpoint-object distance and many other factors. Although the relationship between multiple JND factors may be complex, for simplicity, we can decouple them into several single factors. For example, [] set up a user study to test the JND value with the combined effect of background luminance and viewpoint-object distance. The result proves that viewpoint-object distance can be considered independently from background luminance (Fig. 6). It can be regarded as a coefficient which can be directly multiplied on JND value computed by background luminance. Based on this insight, JND computation for multiple factors is decoupled into several single factors. This significantly simplify the JND model .

Moreover, 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. 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 [].

To answer these 2 questions, we conduct the user study using real Head Mounted Device (HMD) and explore how these new factors influence JND in VR display.

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

X subjects participated in the experiments, including X males and X females. 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 []. In the experiment, a small square area, 32 x 32 pixels, is located in the center of a flat field of constant grey level. For each possible grey level of the flat field (background), 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.

To explore how viewpoint moving speed, depth-of-field, light / dark adaptation influence JND, we set up 3 experiments:

1. **JND v.s. background luminance & viewpoint moving speed** Set the background grey level to 30, 80, 130, 180, 230, then 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. Observer's viewpoint is allowed to chase the moving object. Testing the visibility threshold for these situations.

2. **JND v.s. background luminance & depth of field** Set the background grey level to 30, 80, 130, 180, 230, then set the depth-of-field of the 32*32 square area to 2m, 10m, 50m. Testing the visibility threshold for these situations.

3. **JND v.s. background luminance & light / dark adaptation** Before our test, we show subjects a pure background with different gray level (30, 80, 130, 180, 230) for 20 seconds. In our test, we set background gray level to 30, 80, 130, 180, 230. Testing the visibility threshold for these situations.

4.3 Results

4.3.1 JND v.s. background 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 x% viewpoints are moving faster than y deg/s. (Fig. 7) 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. [ ].

We first analyze tracking condition. Fig. 8 shows $JND(l,v)$, the combined effect of viewpoint moving speed and background luminance to JND. We notice that influence of DoF to JND is very similar in different background luminance, so this combined effect can be decoupled into 2 parts:

$$JND(l,v) = f_{lum}(l) \times f_{track}(v) \tag{4}$$

where $f_{lum}(l)$ represents the visibility threshold with only consideration of background luminance, and $f_{track}(v)$ is a

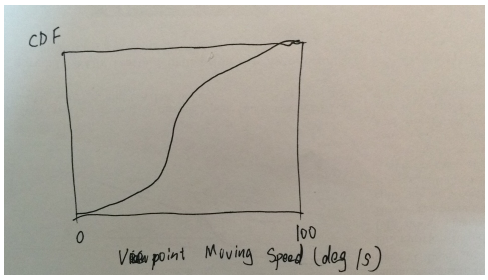


Figure 7: CDF gram of viewpoint moving speed.

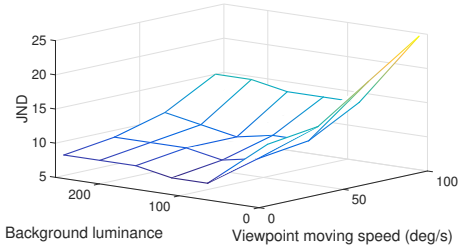


Figure 8: JND due to background luminance and viewpoint moving speed.

coefficient which represents influence of viewpoint moving speed on visibility threshold.

$f_{lum}(l)$ has been presented as Fig. 5. According to curve fitting, we can obtain $f_{track}(v) = \frac{JND(l,v)}{f_{lum}(l)}$ as Fig. 9.

With similar method, we obtain result of no-tracking condition.

4.3.2 JND v.s. background luminance & depth of field

Depth of field (DoF) refers to the distance from the object to human eyes. Human detect object's DoF by eye focusing.

Eye focusing consists of two steps: (1) vergence and (2) accommodation. (Fig. 11). When human looks at an object, the eyes must rotate around a horizontal axis so that the projection of the object is in the centre of the retina in both eyes. To look at an object closer by, the eyes rotate towards each other (convergence), while for an object farther away

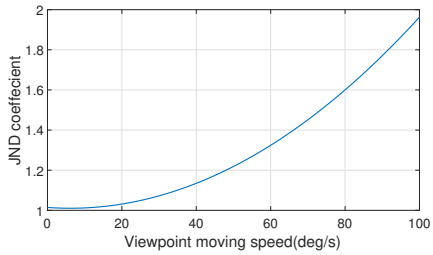


Figure 9: $f_{track}(v)$, the JND coefficient of viewpoint moving speed.

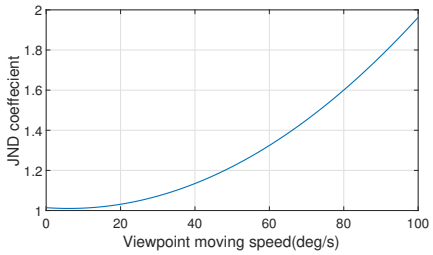


Figure 10: $f_{notrack}(v)$, the JND coefficient of viewpoint moving speed.

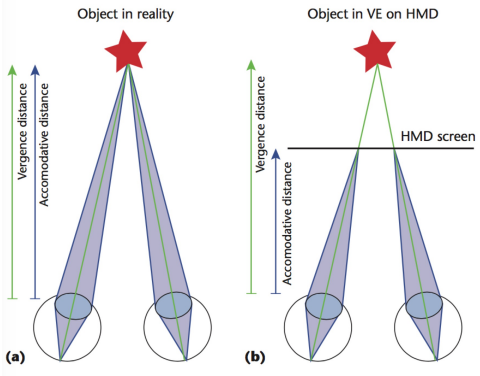


Figure 11: Vergence-accommodation conflict.

they rotate away from each other (divergence). This process is called vergence. Accommodation is the process by which the eye changes optical power to maintain a clear image or focus on an object as its distance varies.

In non-VR video displays, the screen is bioptic, where only a single display is presented that is viewed by both eyes, so contents have the same DoF (in this situation the DoF is exactly the distance from eyes to screen), vergence distance and accommodation distance are consistence. 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. In this situation, accommodation distance is still fixed (from eyes to screen) but vergence distance is different for different contents according to their DoF, thus causes vergence-accommodation conflict. Vergence-accommodation conflict can cause decrease of visual acuity []. As a result, in VR display, objects with high DoF has high JND because its vergence-accommodation conflict is serious.

Fig. 12 shows $JND(l,D)$, the combined effect of DoF and background luminance to JND. We notice that influence of DoF to JND is very similar in different background luminance, so this combined effect can be decoupled into 2 parts:

$$JND(l,D) = f_{lum}(l) \times f_{DoF}(D) \tag{5}$$

where $f(_{lum}(l))$ represents the visibility threshold with

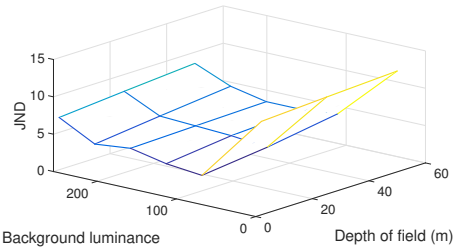


Figure 12: JND due to background luminance and Depth-of-Field.

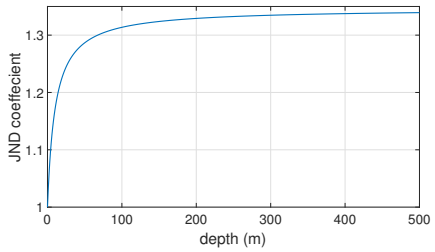


Figure 13: $f_{DoF}(D)$, the JND coefficient of Depth-of-Field.

only consideration of background luminance, and $f_{DoF}(D)$ is a coefficient which represents DoF's influence on visibility threshold.

$f_{lum}(l)$ has been presented as Fig. 5. According to curve fitting, we can obtain $f_{DoF}(D) = \frac{JND(l,D)}{f_{lum}(l)}$ as Fig. 13.

4.3.3 JND v.s. background luminance & light / dark adaptation

In human vision system, in order to transition 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.[] This adaptation period is different for both rod and cone cells and results from the regeneration of photopigments to restore retinal sensitivity. Similarly, when human eyes adjust from low luminescence setting to high luminescence setting,

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 changes dramatically, eyes need a period of time to adapt the new illumination.

Some prior works point out during the process of light/dark adaptation, human visual acuity decreases.

Fig. 14 shows $JND(l,u)$, the combined effect of light / dark adaptation and background luminance to JND. Be similar to above 2 results, this combined effect can also be de-

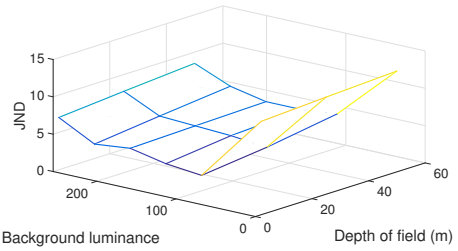


Figure 14: JND due to background luminance and Depth-of-Field.

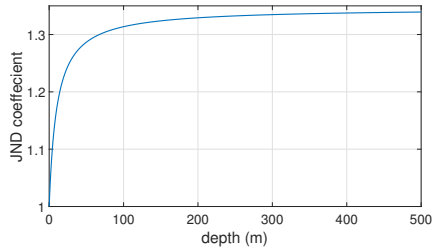


Figure 15: $f_{adapt}(u)$, the JND coefficient of light / dark adaptation.

coupled into 2 parts:

$$JND(l,u) = f_{lum}(l) \times f_{adapt}(u) \tag{6}$$

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

And we obtain $f_{adapt}(D) = \frac{JND(l,u)}{f_{lum}(l)}$ as Fig. 15.

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.

Then JND can be computed as:

$$JND = \max\{f_{lum}(l), f_{text}(t)\} \times f_{dist}(d) \times f_{track/notrack}(v) \times f_{DoF}(D) \times f_{a}$$
(7)

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.

Table 2: symbol and definition used for JND computation

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

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 or they contain different objects. 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:

$$QBE = \frac{PSPNR_{highest} - PSPNR_{lowest}}{B_{highest} - B_{lowest}} \tag{8}$$

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 PSPNR 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 (8). After that, suppose a tile t consists of N_t basic units u_1, u_2, ... , u_{N_t} , we can compute its QBE Variance ($QBEV_t$) as follow:

$$QBEV_t = \frac{\sum_{1 \le i \le N_t} (QBE_{u_i} - E(QBE_{u_i}))^2}{N_t} \tag{9}$$

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

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

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:

$$\begin{aligned} &\min \sum_{i=1}^{T} QBEV_i S_i \\ &\text{s.t.} \bigcup_{i=1}^{N} R_i = V \\ &R_i \bigcap R_j = \emptyset \qquad \forall 1 \le i, j \le T \end{aligned} \tag{11}$$

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.

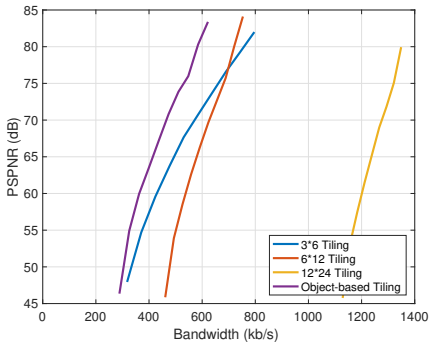


Figure 16: The PSPNR-bandwidth tradeoff of proposed Object-based Tiling scheme and traditional grid-like tiling scheme (3*6, 6*12 and 12*24).

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. 16 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, icoarse 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

According to (1), PSPNR computation needs information of each pixel in each frame. However, in practice, client is unaware of these informations. Moreover, transferring these information to client is impractical because of huge bandwidth consumption.

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

According to (1), in PSPNR computation, PMSE needs the information of each pixel in each frame. Since PMSE is related to both video content and user behavior, it can not be pre-computed on server-side.

However, computing PMSE in realtime causes huge bandwidth consumption. For a tile t, suppose it has F_t frames, with each frame P_t pixels, then computing its PMSE needs $O(F_tP_t)$ communication complexity. This amount of information is at the same level of video content itself, which is unaffordable in practice.

To solve the problem, we present our approximation of PMSE which can compute PMSE of a tile in O(1) client-side computation complexity and communication complexity.

Suppose t is a rectangular tile in a temporal segment which contains frame f_1 to frame f_2. Its top-left pixel is (x_1,y_1) and bottom-right pixel is (x_2,y_2). The Mean Square Error (MSE) of this tile can be computed on server-side:

$$\begin{aligned} MSE_t = E(|p(x,y,f)-\hat{p}(x,y,f)|^2) \\ x_1 \le x \le x_2, y_1 \le y \le y_2, f_1 \le f \le f_2 \end{aligned} \tag{12}$$

where $p(x,y,f)$ denotes the value of pixel (x,y) in frame f of compressed video, $\hat{p}(x,y,f)$ denotes the value of pixel (x,y) in frame f of original video.

Then we decouple JND into Static JND (SJND) and Dynamic JND (DJND). We define this tile's Static JND (SJND) as:

$$\begin{aligned} SJND_t = E[\max\{f_{lum}(l(x,y,f)), f_{text}(t(x,y,f))\} \\ \times f_{DoF}(D(x,y,f))] \\ x_1 \le x \le x_2, y_1 \le y \le y_2, f_1 \le f \le f_2 \end{aligned} \tag{13}$$

where $l(x,y,f)$, $t(x,y,f)$, and $d(x,y,f)$ denotes the background luminance, texture complexity and Depth-of-Field of pixel (x,y) in frame f.

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

We define Dynamic JND (DJND) as:

$$\begin{aligned} DJND_t = E[f_{dist}(d(x,y)) \times f_{speed}(v) \times f_{adapt}(u)] \\ x_1 \le x \le x_2, y_1 \le y \le y_2 \end{aligned} \tag{14}$$

where $d(x,y)$ denotes the distance from pixel (x,y) to user's viewpoint, v denotes the moving speed of user viewpoint.

DJND is only related to client, it is totally independent from video content. Although a brute force computation algorithm still needs $O(P_t)$ computation complexity, we can get its value by integration which needs only O(1) computation complexity:

$$DJND_t = [\iint_{(x_1,y_1)}^{(x_2,y_2)} f_{dist}(d(x,y))dxdy] \times f_{speed}(v) \times f_{adapt}(u)$$
$$= F_{dist}(d(x,y)) \mid_{(x_1,y_1)}^{(x_2,y_2)} \times f_{speed}(v) \quad (15)$$

where $F_{dist}(d)$ is original function of $f_{dist}(d)$.

Then we use MSE, SJND and DJND to fit PMSE:

$$\begin{aligned} PMSE_t &= f(MSE_t, SJND_t, DJND_t) \\ &= MSE_t \times f_1(SJND_t) \times f_2(DJND_t) \end{aligned} \quad (16)$$

In practical, we use discretization to map real number space of SJND and DJND to finite intervals, and then apply our large dataset to get the average value of f_1 and f_2 in each interval.

Fig. X shows the fitting accuracy. In x% situations, error bound of proposed method is less than x%. In realtime VR video streaming, MSE and SJND of each tile are packaged in MPD file which can be downloaded by client, their bandwidth cost is negligible. We can obtain accurate Client-side PSPNR computation only based on them.

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, ... , 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 [] to avoid rebuffing and quality fluctuation.

Then the rate allocation problem can be formalized as:

$$R_k = f(q_{k-1}(R_{k-1}), B_k, C_k) \quad (17)$$

Fortunately, BOLA (Near-Optimal Bitrate Adaptation for Online Videos) [] 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.

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, $\{l_i\}$;
1: Initialize knapsack revenue table $\mathscr{K} \in \mathbb{R}^{(N_k+1)\times R_k}$ by 0;
2: Construct the prefix table $\mathscr{P} \in \mathbb{Z}^{(N+1)\times R_k}$;
3: **for** i from 1 to N_k **do**
4: **for** $b \in [0, B]$ **do**
5: $\mathscr{K}_{i,b} = \min_{j\in[1,M]}\{\mathscr{K}_{i-1,b-r_{i-1,j}} + p_{i-1,j}\}$;
6: $\mathscr{P}_{i,b} = \arg\min_{j\in[1,M]}\{\mathscr{P}_{i-1,b-r_{i-1,j}} + p_{i-1,j}\}$;
7: **end for**
8: **end for**
9: Find $\hat{R}_k = \arg\min_{b\in[0,R_k]}\{\mathscr{K}_{N_k,b}\}$;
10: **for** i from N_k to 1 **do**
11: $l_i = \mathscr{P}_{i,\hat{R}_k}$;
12: $\hat{R}_k = \hat{R}_k - r_{i,l_i}$;
13: **end for**
14: **return** $\{l_i\}$

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.

According to (1), 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. Then we use M constant QP values to encode them into M quality levels. So Each chunk has M different rate, 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 QP as $p_{i,j}$.

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

$$\begin{aligned} \min_{l:l_i\in[1,M]} \quad & \sum_{i=1}^{N} p_{i,l_i} \\ \text{s.t.} \quad & \sum_{i=1}^{N} r_{i,l_i} \le R_k W. \end{aligned} \quad (18)$$

This optimization problem can be solved as a Multiple-Choice Knapsack problem. A brute force search which exhaustively evaluates all combinations guarantees an optimal

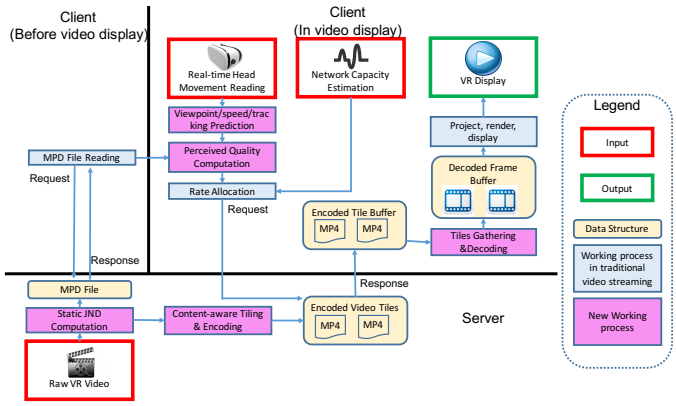


Figure 17: Workflow of PQVRS.

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($R_k MN$).

7 Implementation

PQVRS is open source (10K lines of code across C++, Matlab, Java, and python) and can be accessed at [x]. Now we describe our implementation of PQVRS.

7.1 Implementation of PQVRS workflow

Implementation of PQVRS is shown in Fig. 17. Compared with traditional video streaming system, several modules are newly added or modified, which are highlighted by pink color.

7.1.1 Server-Side

Static JND Computation:

Content-Award Tiling / Encoding:

7.1.2 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 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.

Perceived Quality Computation:

Tiles Gathering & Decoding:

7.2 Challenges in an operational setting