QuRate: Power-Efficient Mobile Immersive Video Streaming

Nan Jiang  
University of Nebraska-Lincoln  
njiang@huskers.unl.edu

Yao Liu  
SUNY Binghamton  
yaoliu@binghamton.edu

Tian Guo  
Worcester Polytechnic Institute  
tian@wpi.edu

Wenyao Xu  
SUNY Buffalo  
wenyaoxu@buffalo.edu

Viswanathan Swaminathan  
Adobe Research  
vishy@adobe.com

Lisong Xu  
University of Nebraska-Lincoln  
xu@unl.edu

Sheng Wei  
Rutgers University  
sheng.wei@rutgers.edu

ABSTRACT

Smartphones have recently become a popular platform for deploying the computation-intensive virtual reality (VR) applications, such as immersive video streaming (a.k.a., 360-degree video streaming). One specific challenge involving the smartphone-based head mounted display (HMD) is to reduce the potentially huge power consumption caused by the immersive video. To address this challenge, we first conduct an empirical power measurement study on a typical smartphone immersive streaming system, which identifies the major power consumption sources. Then, we develop QuRate, a quality-aware and user-centric frame rate adaptation mechanism to tackle the power consumption issue in immersive video streaming. QuRate optimizes the immersive video power consumption by modeling the correlation between the perceivable video quality and the user behavior. Specifically, QuRate builds on top of the user’s reduced level of concentration on the video frames during view switching and dynamically adjusts the frame rate without impacting the perceivable video quality. We evaluate QuRate with a comprehensive set of experiments involving 5 smartphones, 21 users, and 6 immersive videos employing user head movement traces. Our experimental results demonstrate that QuRate is capable of extending the smartphone battery life up to 1.24X while maintaining the perceivable video quality during immersive video streaming. Also, we conduct an Institutional Review Board (IRB)-approved subjective user study to further validate the minimum video quality impact caused by QuRate.

CCS CONCEPTS

• Information systems → Multimedia streaming;

KEYWORDS

Virtual reality, video streaming, power optimization

ACM Reference Format:

1 INTRODUCTION

With the rapidly increasing computing capability and a huge consumer market, modern commodity smartphones have become a popular platform for the emerging computationally intensive virtual reality (VR) applications [29, 43]. These applications can be seamlessly integrated with the recently released VR head mounted display (HMD) mounts, such as Google Cardboard [15], Google Daydream [16], Samsung Gear VR [38], DODOCase [37], and Archos VR Glasses [2]. Moreover, smartphone-based HMDs have enabled a brand new interface for presenting immersive video (a.k.a., 360-degree video) content in the 360 degree of freedom controlled by a user’s head movements. Such immersive video streaming provides users with an enriched viewing experience as if they were an integral part of the video and enables significantly improved quality of experiences (QoE) as compared to the traditional 3D or high definition 2D videos [24]. However, the improved QoE provided by the immersive video comes with significant costs, such as high bandwidth consumption and performance overhead while streaming the 360-degree video frames [4]. Since the emergence of immersive streaming applications, there have been many research efforts focusing on reducing the bandwidth consumption by employing view-based optimizations [3, 18, 34, 35]. However, the community has not fully investigated the power perspective of immersive video streaming. Power consumption is a critical problem in immersive streaming for two key reasons. First, the smartphone-based HMDs are driven by power-constrained batteries. Second, intensive power consumption can accumulate heat that would significantly impact the viewing experience of HMD users due to the device’s wearable nature. This, in essence, makes power consumption an integral part of the QoE. Although power optimization techniques have been proposed for traditional 2D videos on smartphones [8, 19, 25, 52, 53] and wearable devices [23], these techniques cannot effectively reduce the energy consumption of immersive streaming on smartphone HMDs.
This is mainly due to the unique workload and power profile of immersive streaming, described as follows. First, the volume of video data in immersive streaming is huge (i.e., 6X to 8X of the traditional video [40]), as the entire 360-degree frames must be transmitted and processed. This incurs significantly higher power consumption of network and computation, thus leaving a large room for further optimization even after the traditional power optimization techniques are applied. Second, different from traditional video streaming, immersive streaming is a user-centric video application, as it grants the viewers full control over the view angles via head movements and generates the viewport from the 360-degree frame on the smartphone upon each movement. Consequently, frequent user movements would trigger non-trivial power consumption in sensing, computation and view generation, which is not considered by the traditional power optimization techniques. In summary, a new and customized power management mechanism is essential in achieving power efficiency in immersive streaming.

In this work, we investigate the problem of reducing the power consumption in immersive streaming systems. To address the aforementioned challenges, we first conduct a quantitative power measurement study (discussed in Section 3) of immersive streaming on commodity smartphones. Our measurements indicate that the VR view generation operation consumes significant power and is the topmost power consumption source. Based on this observation, we design a quality-aware frame rate adaptation mechanism to reduce the power consumption. Our key idea is to reduce the frequency at which the VR views are generated, i.e., reducing the frame rate of immersive streaming dynamically. We consider the effect of frame rate reduction on the perceivable video quality by leveraging an objective and quantitative video quality metric called spatio-temporal video quality metric (STVQM) [33]. This metric correlates the perceivable video quality with the frame rate and has been proved to be consistent with the subjective quality metric (the mean opinion score (MOS) [45]). We further leverage one of the unique characteristics in immersive streaming, namely user-initiated view switching, in the power optimization mechanism by following two key design principles. (1) No frame rate reduction during fixed view. The mechanism maintains the original frame rate when viewers are not switching views and only reduces the frame rate during view switching. The rationale behind this principle is that, during a view switching process, the viewer’s attention is typically not at the view being switched but rather the view being switched to, and therefore, the reduced frame rate during switching has limited impact on the perceivable video quality. (2) Quality-aware frame rate selection during view switch. The mechanism selects the optimal frame rate to minimize power consumption under the video quality constraint based on the STVQM metric.

We incorporate the above two principles and implement a new frame rate adaptation mechanism called QuRate for smartphone-based immersive video streaming, which optimizes the power consumption in a quality-aware and user-centric manner. QuRate monitors the user movement pattern at runtime and determines the most power efficient frame rate while maintaining the perceivable video quality. Furthermore, to reduce the runtime performance and power overhead introduced by QuRate itself, we develop an offline/online hybrid execution model. In the offline phase, we build a frame rate library (FRL), which quantifies the correlations among quality, frame rate, and head motion, through power/quality profiling based on historical user data. In the online phase, the library FRL is used to determine the instant frame rate based on the dynamic head movement and the quality constraint. We evaluate the effectiveness of QuRate by using real user head movement data and measure the power consumption of immersive video streaming using five commodity smartphones. Our evaluation results show that QuRate can extend the smartphone battery life by up to 1.24X while achieving satisfactory video quality based on a real user study.

To the best of our knowledge, QuRate is the first power optimization framework for smartphone-based immersive video streaming that considers both user behavior and video content. To summarize, we have made the following contributions.

- We for the first time identify the unique problem of power consumption inefficiency in immersive video streaming based on an empirical power measurement study. The observed inefficiency can be attributed to the unique characteristics of immersive streaming which are not considered by the traditional video power optimization techniques.
- We develop an effective power optimization mechanism called QuRate that addresses the aforementioned power inefficiency problem for immersive streaming. QuRate takes into consideration both the unique user behavior and video content features in immersive streaming to achieve power-efficient frame rate adaptation with minimum video quality impact.
- We evaluate and justify the significant power savings and minimum video quality impact achieved by QuRate. Our comprehensive set of evaluations include empirical evaluations based on empirical user head movement traces from a publicly available dataset, as well as an IRB-approved user study.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Immersive Video Streaming

Virtual reality technology can generate three-dimensional virtual environments emulating the physical world, which provides the users with an immersive experience [7]. It is widely used in many areas, such as gaming [36], healthcare [6], and entertainment videos [17, 35]. In a typical VR setup, the user wears a HMD device that displays the specific view based on head movements, similar to what one would see in the physical world.

Among all the VR applications, immersive video streaming has naturally become a hot spot because of the popularity of video streaming in the consumer entertainment market [17, 35]. For example, there are currently millions of immersive videos available on YouTube, the number of which is rapidly growing on a daily basis [51]. In particular, immersive video is attractive in scenarios like live broadcasts of sports games, in which the viewers can switch their views based on their own preferences, as if they were watching the game in person in the stadium [30]. Figure 1 shows a typical end-to-end workflow of an immersive video streaming system, following the ISO standard for Internet video streaming, namely Dynamic Adaptive Streaming over HTTP (DASH) [42]. The end-to-end system follows a client/server architecture. On the server side, the video packager partitions the source 360-degree video into DASH compliant segments [42], which are deployed on a web server for HTTP streaming. On the client side, the web browser
on the smartphone HMD runs a DASH compliant video player [15] integrated with the WebVR library [31] for VR processing and view generation. The client requests and receives video segments from the server via HTTP following the DASH standard [42].

Although the workflow presented in Figure 1 appears similar to a traditional DASH-based video streaming system [42], the immersive video has a major difference compared to the traditional video in that it involves view switches fully controlled by the user. Despite the additional computation complexity caused by calculating and generating the new views dynamically, the unique view switching behavior in immersive video streaming divides the video streaming session into two separate phases: (1) where the user is not paying attention to the video (i.e., during view switching); and (2) where the user is focused on the video content (i.e., when the view is fixed). In particular, the view switching phase, where the video content is not a critical factor for the user experience, brings in the opportunity for trading off video quality for power savings, which we leverage in the QuRate design (discussed in details in Section 4).

2.2 Performance Optimization for Immersive Streaming

Most prior work in the community has mainly focused on the performance optimization of immersive streaming without targeting the power issues. Similar to traditional HTTP-based video streaming, immersive video streaming can lead to a lot of bandwidth consumption. Consequently, prior work focused on exploring various techniques in reducing network bandwidth consumption. For example, Hosseini et al. [18] and Cutuco et al. [12] proposed dynamic view-aware adaptation techniques to divide each 360-degree frame into multiple tiles and only send the views of the user’s interest to save bandwidth. Bao et al. [3] achieved the same goal by predicting the users’ future movements using machine learning-based methods. Qian et al. [35] developed a view prediction mechanism to save the bandwidth of immersive video streaming over cellular network. On the other hand, many works have been conducted building an edge-based VR system to reduce the latency. Shi et al. [39] reduced the latency without the requirement of pre-rendering or viewpoint prediction by building an edge-based system. Li et al. [22] proposed a solution called MUVR to maximize efficiency when dealing with multiple edge users. Other than bandwidth and latency optimization, Liu et al. [24] aimed to optimize resource utilization efficiency and QoE.

2.3 Power Efficient Mobile Video Streaming

With the popularity of streaming video content on power constrained mobile devices in the past decade, there have been many research efforts on exploring power efficient streaming mechanisms for mobile videos (i.e., traditional 2D videos). The state-of-the-art research can be categorized into two directions. One line of research focuses on power measurement study or optimization for generic mobile applications. For example, Carroll et al. [8] measured and analyzed the power of each component on a smartphone for general non-video streaming cases. Zhang et al. [53] designed and released a tool that can estimate the power consumption of smartphone components for any generic applications using built-in battery voltage sensors and knowledge of battery discharge behavior. In addition to smartphones, LiKamWa et al. [25] conducted a full-fledged power profiling on a wearable device (i.e., Google Glass).

The other line of research focuses specifically on power efficient mobile video streaming. For example, Zhang et al. [52] evaluated power consumption of traditional video streaming on mobile devices with 4G/LTE. Wei et al. [47, 49] developed HTTP/2 server push-based mechanisms that reduce the power cost for traditional video streaming. Liu et al. [25, 48] reduced the screen power consumption by leveraging GPU to maintain the luminance during traditional video streaming. Recently, Jiang et al. [21] presented a power breakdown analysis on smartphone HMDs for immersive video streaming. Yan et al. [50] proposed an energy efficient VR system that dynamically scales the brightness of the VR display according to the user’s eye movement. To the best of our knowledge, our work is the first to optimize the power consumption of immersive video streaming by considering both user behavior and video content.

3 POWER MEASUREMENT STUDY FOR IMMERSIVE STREAMING

In this section, we present a quantitative power measurement study of immersive video streaming on multiple smartphone HMDs. The key goal is to characterize immersive streaming-specific power usage patterns and to shed light on designing the corresponding power optimization strategies.

3.1 Immersive Streaming System Setup

We set up a reference immersive streaming system following the workflow depicted in Figure 1. Table 1 summarizes the detailed system setup information for each component. In order to conduct a comprehensive evaluation incorporating the software and hardware variations of the smartphones under test, we adopt the first four smartphones in Table 2 as the test platform for power evaluation. The selected smartphones all have removable batteries, making it feasible to directly use external power monitor [27]. In addition, these smartphones cover a wide variety of different software and hardware settings including CPU, GPU, chipset, battery capacity, OS version, etc., which may impact the power efficiency during immersive streaming. The fifth phone is used in the stress test for evaluation, as presented in Section 5.5.

For the power measurement, we connect the Monsoon power monitor [27] to the smartphone HMD as the power supply, as shown in Figure 2. Then, we playback the immersive video on the smartphone and collect the power samples generated by the monitor at the interval of 20 ms using the PowerTool software [26]. To eliminate the potential power noise from irrelevant components,
we configure each smartphone as follows during the entire measurement study: mute the smartphone, turn on the airplane mode with only WiFi enabled, turn on the power saving mode, and set the smartphone to the lowest brightness.

3.2 Power Breakdown: A Closer Look
Since users’ head movements play an important role in the power consumption of the immersive video, we adopt the methodology in [21] and [50] to measure the power consumption with real user traces. In particular, we adopt a publicly available user head movement dataset [11] in our power measurement study, which were collected from real world users. We adopt 3 representative immersive videos (i.e., Videos 1 - 3) from the dataset, as described in Table 4, which cover varying frequencies and speeds of motions. Also, we choose 3 arbitrary users (i.e., Users 1, 2, and 3 in [11]) who have watched Videos 1-3 and use their actual head movement data to evaluate the power consumption of these three videos.

Furthermore, we consider five power components, namely basic, screen, video decoding, networking, and VR view generation, as the major power consuming sources on the smartphone while playing the immersive videos. To uncover the power consumption of each component, we conduct differential power measurements and analysis with 5 test cases as shown in Figure 3, with each case collecting 1-minute of power samples from the power monitor. In Case 1 (Screen off) and Case 2 (Screen on), we turn the screen of the smartphone off and on without playing the immersive video yet, and the power difference between these two cases represents the power consumption of the screen itself. In Case 3 (Local playback), we store and play the immersive video locally on the phone without streaming over the network and, therefore, the difference between Case 3 and Case 2 represents the video decoding power. Next, Case 4 (Streaming without VR view generation) is similar to Case 3 except that the video content is now streamed from the remote server and, therefore, their difference represents the power consumed by the network communication. Finally, in Case 5 (Full immersive streaming) we conduct a full-fledged immersive video streaming, and the increased power consumption from Case 4 is due to the VR view generation.

Table 3 and Figure 4 illustrate the power breakdown results from the differential power evaluation. We observe that the 4 smartphones exhibit similar power breakdown results despite the hardware and software variations. Among all the power components, VR view generation consumes the most power (between 40.2% to 44.5%), and the network consumes the second most (between 27.7% to 28.9%), which matches our expectations given the size of the videos and that the VR views must be frequently generated on the smartphone. We also notice that the screen takes the third place in the chart (between 14.1% to 15.9%), which is corresponding to the high resolution and large size of the screens.

The power breakdown results provide us with several key insights towards the potential power optimization strategies. First, we observe that the immersive video consumes a significant amount of power on all four smartphones, which outweighs the power impact of device variations in terms of software and hardware configurations. Therefore, in this specific scenario, a content-based optimization approach specific to the immersive video would be desirable to significantly improve the power efficiency. Second, the power evaluation results reveal that the VR view generation consumes the most power compared to other power consumption sources evaluated. This suggests that when designing power optimization techniques, we should prioritize the power reduction of the VR view generation process. All the above insights combined lead to our proposed quality-aware and user-centric power optimization framework, namely QuRate, to reduce the VR view generation power.
Table 2: Five smartphones adopted in the immersive video power experiments, which involve a large variety of software and hardware configurations to uncover the specific power profile for immersive videos.

<table>
<thead>
<tr>
<th>Phone</th>
<th>LG V20</th>
<th>Samsung S7</th>
<th>Moto G5</th>
<th>LG G4</th>
<th>Google Pixel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android Version</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
<td>6.0</td>
<td>9.0</td>
</tr>
<tr>
<td>Battery</td>
<td>Removable</td>
<td>Embedded</td>
<td>Removable</td>
<td>Removable</td>
<td>Embedded</td>
</tr>
<tr>
<td>Battery Capacity</td>
<td>3200mAh</td>
<td>3000mAh</td>
<td>2800mAh</td>
<td>3000mAh</td>
<td>2770mAh</td>
</tr>
<tr>
<td>RAM</td>
<td>4 GB</td>
<td>4 GB</td>
<td>2/3 GB</td>
<td>3 GB</td>
<td>4 GB</td>
</tr>
<tr>
<td>CPU (Hz)</td>
<td>2.15 G &amp; 1.6 G</td>
<td>2.15 G &amp; 1.6 G</td>
<td>1.4 G</td>
<td>1.4 G &amp; 1.8 G</td>
<td>2.15 G &amp; 1.6 G</td>
</tr>
<tr>
<td>GPU</td>
<td>Adreno 530</td>
<td>Adreno 530</td>
<td>Adreno 505</td>
<td>Adreno 418</td>
<td>Adreno 530</td>
</tr>
<tr>
<td>Resolution</td>
<td>1440 x 2560</td>
<td>1440 x 2560</td>
<td>1080 x 1920</td>
<td>1440 x 2560</td>
<td>1080 x 1920</td>
</tr>
<tr>
<td>Chipset</td>
<td>MSM8996-820</td>
<td>MSM8996-820</td>
<td>MSM8937-430</td>
<td>MSM8992-808</td>
<td>MSM8996-821</td>
</tr>
</tbody>
</table>

Table 3: Quantitative power breakdown values from differential power evaluation on four phones (mW).

<table>
<thead>
<tr>
<th>Phone</th>
<th>LG V20</th>
<th>Samsung S7</th>
<th>Moto G5</th>
<th>LG G4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>50.2</td>
<td>141.2</td>
<td>36.4</td>
<td>54.1</td>
</tr>
<tr>
<td>Screen</td>
<td>557.6</td>
<td>577.3</td>
<td>464.6</td>
<td>418.5</td>
</tr>
<tr>
<td>Video Decoding</td>
<td>501.3</td>
<td>769.4</td>
<td>513.7</td>
<td>577.3</td>
</tr>
<tr>
<td>Network</td>
<td>984.1</td>
<td>1443.3</td>
<td>1052.4</td>
<td>1045.6</td>
</tr>
<tr>
<td>VR View Generation</td>
<td>1409.3</td>
<td>2068.7</td>
<td>1578.6</td>
<td>1683.3</td>
</tr>
</tbody>
</table>

Figure 4: Power breakdown results of the smartphone HMDs during immersive streaming.

4 OUR PROPOSED POWER OPTIMIZATION APPROACH: QU RATE

The goal of QuRate is to reduce the power consumption of immersive video streaming on smartphones. According to the VR power measurements conducted in Section 3, a significant amount of energy is consumed by the smartphone HMD for real-time view generation in response to user’s head movements. More importantly, such computations are currently conducted for every 360-degree frame by default. Based on this observation, our key insight is to reduce the frame rate (i.e., the number of times per second that VR views are generated) to achieve power savings. However, the challenge is to maintain the user perceivable video quality while reducing the frame rate, which we aim to address in the design of QuRate.

4.1 QuRate System Architecture

Figure 5 shows the system architecture of the proposed power optimization mechanism QuRate, as part of the immersive streaming system. The unshaded blocks represent the original immersive streaming system involving hardware, OS, and application layers. The shaded blocks illustrate the architecture and workflow of QuRate, which is a cross-layer system component tightly integrated with the original system. QuRate consists of four components (blue blocks) that regulate the frame rates by interposing in between the VR framework and the HMD hardware. More concretely, first, the Motion Detector obtains the device orientation information from the hardware sensor and determines whether the smartphone HMD is in motion as well as its current speed. Then, the Frame Rate Controller obtains the motion information from the motion detector and executes our frame rate selection algorithm, which selects the most power efficient frame rate under an application or user specific quality requirement. In particular, QuRate obtains the mapping between frame rate and video quality from a frame rate library that was generated offline based on historical power profiling using real user head movement data. Last, the Frame Rate Controller uses the application-level VR framework to generate the VR view based on the currently selected frame rate.

4.2 Frame Rate Adaptation Method

4.2.1 Feasibility. The feasibility of power optimization via frame rate reduction is mostly constrained by the fact that a reduced frame rate may lead to downgraded video quality and thus compromise the user experience. We explore such feasibility from the field of biology, where researchers have discovered the limitations of human...
eyes in observing moving object. Normally, when the velocity of an object is larger than 20 degree/second, the gain (i.e., the ratio between eye velocity and object velocity) can no longer maintain in the range of 0.9 to 1.0, which is required for the human vision system to observe the object clearly [14]. In this case, corrective saccades, a compensation mechanism that head and eye-ball movements, is needed to realign the target. However, according to [44], the possibility of error in corrective saccades is 29% - 79% depending on the environment, which means corrective saccades is highly unreliable and the eyes would still have blurred vision while viewing a fast moving object.

Based on the above evidence and the scientific discovery from the biology field, reducing the frame rate of VR video in a reasonable range and while the user’s view is fast switching would pose insignificant impact to the user experience, because the view is already blurred to begin with. This key observation serves as the basis of our frame rate reduction method for power optimization, which we present in details in the next subsections and further justify using subjective user studies in Section 5.7.

4.2.2 Practical Frame Rate Adaptation. For a premium viewing experience, the frame rate of immersive video is typically 60 FPS. Since a large amount of computation must be conducted at the rendering of each video frame (e.g., read the viewer’s orientation, locate the field of view within the 360-degree frame, and generate the left and right views for the viewer’s eyes), it leaves large room for power savings by reducing the frame rate (i.e., the frequency that the VR view is generated). However, since a reduced frame rate may significantly impact the video quality, we only conduct such reduction while the user is switching views. Our intuitions are two-fold. First, the video scene during fast view switching will be low quality to begin with based on the discussions in Section 4.2.1; Second, the video quality during view switching is non-critical to the user experience, as it is an indication that user is interested in the new view. Taking a 360-degree soccer video as an example, the user would focus on a fixed view, such as two players grabbing the soccer ball from each other. Then, when the ball is passed through a wide range, the user’s attention will switch and track the ball until it reaches another fixed view. During the switching, i.e., while both the user’s orientation and the ball are in motion, the quality of the video and thus the frame rate is much less critical to the user’s experience, which can be reduced without compromising the QoE.

Based on this observation, in QuRate, we maintain the original frame rate while the view is fixed (i.e., motion speed below a noise threshold) and only reduce the frame rate when the user switches from the current view to a new view. The frame rate reduction mechanism is shown in Algorithm 1, which employs the Motion Detector to determine whether the frame rate should be reduced.

### 4.3 Quality-Aware Offline Training and Online Frame Rate Selection

Despite its obvious effectiveness in power savings, it is well known that frame rate reduction would degrade the quality of the video if not well controlled. Therefore, we must quantitatively evaluate the quality loss due to frame rate reduction and develop a systematic approach to minimize it. As the first step in achieving this goal, we adopt an objective video quality metric, namely spatio-temporal quality metric (STVQM) [33] to evaluate the quality of the immersive video under frame rate control, which considers the interactions between spatial and temporal quality perceptions:

\[
STVQM = SVQM \cdot \frac{1 + a \cdot TI^b}{1 + a \cdot TI^b \cdot (30/FR)},
\]

where \(a\) and \(b\) are constants determined by a least-square non-linear fitting using the subjective data, which leads to \(a = 0.028\), \(b = 0.764\); FR refers to frame rate; and SVQM (spatial video quality); TI (temporal information) and SI (spatial information) are calculated as [46]:

\[
SVQM = 100 - e^{-(PSNR+\omega_S \cdot SI+\omega_T \cdot TI-\mu)/s},
\]

\[
TI = \max \{\text{std}_\text{space}[M_n(i,j)]\},
\]

\[
SI = \max \{\text{std}_\text{space}[\text{Sobel}(F_n)]\},
\]

In Equation (4), \(\text{std}_\text{space}\) stands for the standard deviation of the pixels in one video frame. \(\text{Sobel}(F_n)\) refers to the pixels in the video frame at time point \(n\) after being filtered with a sobel filter [41]. \(M_n(i,j)\) in Equation (3) refers to the pixel differences between the frames in the user’s view of time points \(n\) and \(n-1\) at position \((i,j)\). In addition, \(PSNR\) in Equation (2) refers to the peak signal to noise ratio, which is a commonly used video quality metric [20]. All other constants are chosen by a least-square non-linear fitting algorithm as described in [33], where \(\omega_S = 0.0356\), \(\omega_T = 0.236\), \(\mu = 36.9\), and \(s = 2.59\).

The reason why we choose this metric is that it takes into account both the motion in the video and the frame rate being applied. The former (i.e., motion) matches well with the motion feature of the immersive video, which includes both the motion in the original video and that caused by user-initiated view switches. The latter (i.e., frame rate) matches well with the proposed approach based on frame rate control. Furthermore, according to [33], the STVQM metric has been clearly justified by the mean opinion scores from well organized subjective experiments.

Based on the STVQM metric and representative user head movement data (e.g., from [11]), we can calculate the quality-aware and power-efficient frame rate by rewriting Equation (1) as follows:

**ALGORITHM 1**: Frame Rate reduction during view switching.

1. Let \(flag\) be the indicator of view switching, i.e., 1 refers to view switching and -1 refers to view fixed.
2. Let \(S\) be the switching speed threshold.
3. Let \(Switching\_Speed\) be the current speed of view switching, calculated by \(VRPose()\).
4. **Function render()**
5. if \(Switching\_Speed \leq S\) then
   6. \(flag \leftarrow -1;\)
   7. else
   8. \(flag \leftarrow 1;\)
   9. end if
10. if \(flag = 1\) then
11. Reduce render frequency;
12. end if
13. ViewPoint \(\leftarrow\) NewViewPoint;
14. end
ALGORITHM 2: Quality-aware frame rate selection.

1. Input Motion Speed (v);
2. Output Frame Rate (FR);
3. Set the minimum acceptable STVQM from offline FRL as Q;
4. Function calcFrameRate(V)
5. Pose ← VDisplay.getStyle(); // WebVR API [28]
6. V ← Pose.linearVelocity(); // WebVR API
7. FR ← FRL(V, Q); // Equation (6)
8. end

\[ FR = \frac{30 \cdot a \cdot T1^b \cdot STVQM}{SVQM \cdot (1 + a \cdot T1^b) - STVQM}. \]

Following Equation (5), we can calculate the frame rate at the system runtime based on the quality requirement of the target video. However, we note that such an online frame rate calculation is infeasible due to the complexity of Equation (5), which requires the computations of \( T1, SI, \) and \( SVQM \) every time the video or user motion varies at runtime. According to [33] and [46], such computations involve pixel-level processing of one or multiple video frames, which by itself incurs non-trivial performance and power overhead and may offset the power saving goal of QuRate.

To address the challenge of the direct online mechanism, we develop an offline frame rate library, as presented in Figure 5, to facilitate power-efficient frame rate reduction at runtime. This library can be built using a dataset of user head movement data while watching immersive videos. In particular, for each user \( u_i \), we record each video \( v_j \), where \( 1 \leq i \leq I, 1 \leq j \leq J, \) and \( I \) and \( J \) represent the number of users and videos in the dataset, respectively, we conduct the following three steps to build the frame rate library:

- **Step 1**, assign user \( u_i \)'s movement data to an automatic view switching algorithm and play/record the VR video \( v_j \) with user \( u_i \)'s movement;
- **Step 2**, calculate the \( T1 \) and \( SI \) values of the recorded video following Equations (3) and (4), as well as the \( SVQM \) value following Equation (2); and
- **Step 3**, employ Equation (1) to calculate the STVQM value for video \( v_j \) at user \( u_i \)'s view switching speed and all possible frame rates \( FR \) (e.g., 10, 20, ..., 60).

We repeat the above three steps for all the user-video pairs and obtain the following lookup table:

\[ FR = FRL(v, Q), \]

where \( FRL \) represents the frame rate library, which is not a closed form equation but presented as a lookup table obtained from the user/video dataset; \( v \) is the user motion speed available in \( FRL \) that is closest to the instant motion speed of the target user; and \( Q \) is the objective video quality that the user aims to maintain.

The generated \( FRL \) enables us to determine the power efficient frame rate for a new user. In particular, the parameters \( Q \) and \( v \) are corresponding to the quality-aware and user-centric design principles in QuRate, respectively.

Based on the offline frame rate library in Equation (6), we develop the online algorithm for frame rate adaptation, as shown in Algorithm 2. The algorithm selects the best frame rate based on the current user’s view switching speed, which is determined by QuRate through the sensors on the smartphone HMD.

### 4.4 Estimating Power Consumption

During our experiments, we have noticed that manual power evaluation is a tedious process for each user-video pair. For example, for a one-minute video, we must spend at least one minute for the video playback and roughly another minute for preparing the test and collecting the results. In addition, the measurement noise is very common due to the complexity of the smartphone [8]. Other than that, the power measurement requires re-structuring the interconnection of the battery component, which increases the uncertainty. The experiment also needs to be paused frequently to cool down the system and avoid the inaccuracy caused by the generated heat.

To overcome these challenges, we develop an analytical power model for the immersive video streaming system. This power model is based on the power measurement samples we have obtained and can be used to analyze the power consumption with the QuRate scheme. In this way, we can estimate the power consumption after only measuring the power once in the default case. This is helpful in tuning the power optimization framework (e.g., adjusting the threshold values).

Theoretically, when the frame rate is adjusted to a constant value, the average power consumption during the playback can be estimated using the following equation:

\[ P_{Est.} = (1 - \alpha) \cdot P_{Def.} + \alpha \cdot P_{Def.} \cdot \frac{FR}{FR_{Def.}}, \]

where \( P_{Est.} \) refers to the estimated power consumption with the frame rate control; \( \alpha \) refers to the percentage of power consumed by view generation over the total power consumption; \( P_{Def.} \) is the actual power consumption with the default frame rate \( FR_{Def.} \); and \( FR \) is the constant value that the frame rate is adjusted to.

We further expand Equation (7) to consider the case that the frame rate is varying during the playback (i.e., after adopting the QuRate scheme), as shown below:

\[ P_{Est.} = (1 - \alpha) \cdot P_{Def.} + \alpha \cdot P_{Def.} \cdot \sum_{i=1}^{n} (\eta_i \cdot \frac{FR_i}{FR_{Def.}}), \]

where \( n \) is the number of different frame rates, and \( \eta_i \) is the frequency of each frame rate \( FR_i \) that appears during the video playback. In this way, we can estimate the power consumption after only measuring the power once in the default case. This is helpful in tuning the power optimization framework (e.g., adjusting the threshold values). In Section 5.4, we evaluate the accuracy of our predictive power model for immersive video streaming under varying frame rates.

### 5 Evaluation

We evaluate QuRate with the goal of understanding its efficiency in power savings and the potential impact, if any, on the perceivable quality of the video. In particular, we first measure and compare the power consumption in the cases with and without QuRate using empirical head movement data. Then, we evaluate and justify the power analytically model by comparing the modeled power results with the empirical measurements. Also, we conduct battery stress...
test to further verify the power evaluation results in empirical user settings. Last but not least, we carry out IRB-approved subjective QoE evaluations with human subjects involved, which proves the minimum impact QuRate poses on the perceivable video quality.

5.1 Experimental Setup

We adopt the same system setup (i.e., the power monitor and five smartphones) as in Section 3 for our evaluation of QuRate. Also, based on the test videos described in Table 4 obtained from the publicly available head movement dataset [11], we select 21 out of 59 users who have watched the same set of 6 videos (referred to as Videos 1 to 6 hereafter based on Table 4). We calculate the switching speeds of the 21 users based on the timestamps and orientation coordinates provided by the dataset, as shown in Equation (9), where \( S_i \) represents the switching speed of the orientation vector \( O_{k} \) from time \( t_{i-1} \) to \( t_i \).

\[
S_i = \frac{\arccos(\frac{\bar{O}_{k} \cdot \bar{O}_{k+1}}{||\bar{O}_{k}|| ||\bar{O}_{k+1}||})}{t_i - t_{i-1}}.
\]  

For each video, we rank the 21 users based on the average speed of each user watching all the 6 videos. In order to study the impact of the user’s view switching speed, we select 4 representative users for each video to construct the offline frame rate library (e.g., for Video 1, we select User 8 ranked 19th, User 3 ranked 14th, User 7 ranked 8th, and User 6 ranked 5th), as shown in Table 5. In this process, our selection criterion is to cover high, medium, and low ranked user groups.

5.2 Offline Frame Rate Library Creation

We build the offline frame rate library by calculating the STVQM values for all the 6 videos following Equation (1), as shown in Table 4, where the STVQM score refers to the quality of the video itself (i.e., without applying the users’ movement). Then, we use the STVQM scores to categorize the motions of the 6 videos into slow, medium, and fast based on the understanding from [33], where a slower motion video obtains a lower STVQM score.

Next, we apply the head movements of selected users from Table 5 to each video and calculate all the parameters (e.g., TI and SI) using Equations (1) to (2) with a TI and SI calculator [46] and a screen recorder [32] as described in Section 4.3. Finally, we plot 4 curves representing the frame rate library (i.e., Equation (6)) for each video to indicate the relationship between the video quality and the frame rate under different view switching speeds, as shown in Figure 6. Each curve in Figure 6 represents one user and thus indicates the behavior of one switching speed for the video. We observe that for each video, a faster switching speed requires lower frame rate at the same STVQM. This matches with our intuition that a fast switching view indicates the user’s lack of interest in the current view, which allows us to reduce the frame rate while still maintaining the premium video quality.

For each video in Figure 6, we choose the video quality of users with the lowest switching speed at 60 FPS as the target video quality (e.g., we select the STVQM objective as 48 for Video 1). After applying the 4 users’ switching speeds to Figure 6, we build the frame rate library to facilitate the online frame rate selection for an arbitrary new user, as shown in Table 4. We consider any switching speed slower than the slowest speed in Table 5 as a fixed view, for which we apply the highest frame rate (i.e., 60 FPS). Based on our statistical analysis of the 21 users, the percentages of fixed views in the 6 test videos are 36%, 33%, 37%, 37%, 32%, and 35%, which indicate large (more than 60%) room for power reduction.

5.3 Online Quality-aware Frame Rate Selection

5.3.1 Evaluation Method. We choose 10 users that are not involved in Table 5 for each video (i.e., Users 10 - 19) as the test user set to evaluate the effectiveness of QuRate at the online stage. For these 10 users, we first calculate their average switching speeds, e.g., the solid curve in Figure 7 shows the view switching speed of User 10 watching Video 1. Then, based on the frame rate library, we assign a frame rate to each second of the video, as presented by the dashed curve in Figure 7. For example, at the 30th second, if the switching speed of User 10 watching Video 1 is faster than the fast switching speed in the frame rate library, we choose the frame rate as 20 FPS.

5.3.2 Feasibility Evaluation. We conduct a feasibility evaluation to validate our hypothesis that users typically spend non-trivial amount of time in view switching and thus enable the opportunity for applying QuRate for power savings. Figure 8 summarizes the frequencies of view switches that are beyond the pre-defined threshold speed for frame rate reduction (i.e., considered as a view switch by QuRate), which are based on the public dataset [11]. We observe that the average frequency of view switching for all the 60 user/video combinations is 22.8%, with the highest of 68.1%, which indicates potential opportunities for power savings via QuRate.

Furthermore, the switching frequencies demonstrate noticeable dependencies on individual users, which justifies the necessity of the user-centric principle adopted by QuRate.

5.3.3 Power Evaluation and Comparison. In order to evaluate the performance of QuRate, we apply each user’s head movement data to Algorithms 1 and 2. Then, we measure the power consumption and video quality of each user watching the videos with two other cases for comparison: (1) no frame rate reduction (i.e., the Default case); and (2) no QuRate for quality control (i.e., the Naive case). Figure 9 summarizes the average power consumption of 10 users (i.e., Users 10 - 19) watching each video in the three cases on the LG V20 phone, where Naive means reducing the frame rate to the lowest value (i.e., at 10 FPS) without considering the quality impact. Figure 10 presents the runtime video quality (i.e., the STVQM value) of each case with User 10 watching the 6 videos. The standard deviations of the curves are 15.37 - 15.85 (Default), 3.97 - 3.90 (QuRate), and 4.20 - 4.30 (Naive). Furthermore, we repeat the experiments with Users 10 - 14 on Samsung S7, Moto G5, and LG G5, the results of which are shown in Figure 11. We observe that the Naive case saves the most power (27.57% to 43.89%) in our evaluations. However, it also results in the lowest video quality as shown in Figure 10. Also, the default frame rate achieves the highest video quality most of the time. Yet, it is highly unstable (i.e., the standard deviation can be up to 15.85) and consumes the highest power. After applying the QuRate scheme, the power consumption is reduced by a considerable amount (5.62% to 32.74%) with relatively consistent video qualities, as compared to the Default case. In addition, we notice that by using QuRate, the power consumption distribution is much larger than the other two approaches. We believe this is because QuRate is user motion related.
Table 4: Summary of the six test videos from dataset [11] in terms of STVQM score, video properties (i.e., motion, video length, and resolution), and sample representations of the derived frame rate library (i.e., STVQM objective, and frame rate selection at slow, medium, and fast switching speeds).

<table>
<thead>
<tr>
<th>Video</th>
<th>Name</th>
<th>STVQM Score</th>
<th>Motion</th>
<th>Video Length</th>
<th>Resolution</th>
<th>STVQM Objective</th>
<th>Slow</th>
<th>Medium</th>
<th>Fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rhinos</td>
<td>48.78</td>
<td>slow</td>
<td>1:41</td>
<td></td>
<td>48</td>
<td>60</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Paris</td>
<td>66.94</td>
<td>Medium</td>
<td>4:04</td>
<td>3840 × 2048</td>
<td>92</td>
<td>60</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Roller Coaster</td>
<td>83.03</td>
<td>Fast</td>
<td>3:26</td>
<td></td>
<td>118</td>
<td>60</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>Diving</td>
<td>41.55</td>
<td>Slow</td>
<td>6:52</td>
<td></td>
<td>38</td>
<td>60</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>Timelapse</td>
<td>96.44</td>
<td>Fast</td>
<td>1:31</td>
<td></td>
<td>80</td>
<td>60</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Venice</td>
<td>31.49</td>
<td>Slow</td>
<td>2:55</td>
<td></td>
<td>48</td>
<td>60</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 6: Offline frame rate library generation based on 6 videos and 4 users.

Table 5: Selected users’ motion speeds (degree/second).

<table>
<thead>
<tr>
<th>Video</th>
<th>User</th>
<th>Speed</th>
<th>Rank</th>
<th>Video</th>
<th>User</th>
<th>Speed</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>12.2</td>
<td>19</td>
<td>2</td>
<td>4</td>
<td>31.9</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>28.4</td>
<td>14</td>
<td>7</td>
<td>7</td>
<td>87.2</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>80.3</td>
<td>5</td>
<td>20</td>
<td>122.9</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>17.2</td>
<td>17</td>
<td>8</td>
<td>39.7</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>25.3</td>
<td>12</td>
<td>4</td>
<td>44.6</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>81.8</td>
<td>4</td>
<td>5</td>
<td>68.5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>98.9</td>
<td>3</td>
<td>7</td>
<td>110.5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>17.2</td>
<td>20</td>
<td>8</td>
<td>27.7</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>81.8</td>
<td>16</td>
<td>2</td>
<td>49.1</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>83.7</td>
<td>8</td>
<td>6</td>
<td>71.1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>118.0</td>
<td>2</td>
<td>1</td>
<td>158.0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7: View switching speed vs. Frame rate selected by QuRate for User 10 watching Video 1 in dataset [11].

and different users would incur different power consumption and thus the wide distribution.

Frame Rate in Case Naive. We also conduct an experiment analyzing different frame rate in Case Naive with Users 10 - 19 watching Video 1 using LG V20, as shown in Figure 12. We measure the power consumption of the system while setting the frame rate to 10 FPS, 20 FPS, and 30 FPS (i.e., Naive_10, Naive_20, and Naive_30).

Apparently, larger frame rate results in larger power consumption. However, we also observe that for some users (e.g., Users 15, 16, and 19), the power consumption of QuRate is less than Naive_30. This finding matches with the percentage of fast movements in Figure 8, as we assign a lower frame rate to fast view switching. Therefore, if the user is always switching the view at a high speed, the total power cost might be lower than some of the Naive cases.
We adopt the battery stress test as a means of cross validating the values calculated by the model. We observe that the two curves in Figure 9.25% (\(|T_{QuRate} - T_{Default}|/T_{Default}\)) refer to the improvement brought by QuRate (i.e., \(T_{QuRate} - T_{Default}\)). The results indicate that QuRate effectively extended the battery life by 68.5% to 124.3% compared to the Default case during immersive video streaming.

**Discussions.** We observe that the above battery life extension from reducing the power consumption of VR View Generation exceeds the proportion that component takes in the whole power profile (i.e., around 40%). We believe that it is caused by additional power savings from other system components while adopting the QuRate method. For instance, [9] reports that the video content and frame rate would significantly impact the power consumption of the screen display, which is not counted into the VR View Generation category in our power profiles.

### 5.4 Accuracy of Power Modeling

In order to verify the power analytical model proposed in Section 4.4, we employ the data from the Default and Naive cases in Figure 9, together with Equation (8), where \(a = 40.2\%\), and \(FR_{Def}\) and \(FR\) are 60 and 10, respectively. Figure 13 shows the results on 4 smartphones comparing actual power measurements and the power values calculated by the model. We observe that the two curves in each figure are very close to each other, and the statistical analysis shows that the average discrepancy between the two curves is only 9.25\% \((P_{Actual} - P_{Calculation})/P_{Actual}\), which is acceptable for the requirement of a power model in tuning the power optimization framework.

### 5.5 Battery Stress Test

**Stress Test Methodology.** While the measurement results from the power monitor provides us with a high-resolution power evaluation, the effectiveness of the evaluation heavily depends on that of the power monitor and many settings on the smartphone under test. To eliminate the impact of potential power measurement noise, we adopt the battery stress test as a means of cross validating the power evaluation results in an empirical user setting. In a nutshell, the stress test emulates the actual user’s viewing behavior on the smartphone by repeatedly and continuously playing back the test immersive video. During this process, we periodically sample the statistics of the remaining battery capacity from the OS kernel log, which serves as the most straightforward and empirical power metric that a regular end user would perceive. The test continues until the battery completely drains, at which point we measure the total video playback time and use it as the practical indicator for power efficiency.

**Stress Test Results.** In our implementation of the stress test, we first adjust the phone settings to eliminate the noises as described in Section 3.1. Then, we started the test video playback on the smartphone after it is fully charged. Once the battery drains out, we extract the battery data information during the whole course of the test leveraging the BatteryStats framework on Android. Figure 14 summarizes the results of the stress test. Given the fact that the stress test does not require sophisticated power monitor connection, it enables us to employ a newer smartphone model (Google Pixel with Android 9.0). The numbers next to the curves indicate how long the battery lasts (in minutes) and the percentage refers to the improvement brought by QuRate (i.e., \(T_{QuRate} - T_{Default}\)). The results indicate that QuRate effectively extended the battery life by 68.5% to 124.3% compared to the Default case during immersive video streaming.

**Discussions.** We observe that as the resolution of the video drops, the power consumption would also be reduced. This is because lower resolutions typically lead to smaller video sizes, which requires lower power during the transmission. We also note that the power saving of QuRate increases from 28.97\% to 31.92\% as the resolution decreases. This is consistent with our expectation as for low resolution cases, the percentage of power cost in view generation is larger. Consequently, the same proportion of power saving in view generation would lead to larger total savings. For instance, the percentages of view generation in high and low video resolutions are 40\% and 60\%, respectively. Assuming QuRate saves 50\% power in the view generation, there will be 20\% savings in the high resolution video, and 30\% savings in the low resolution case.

### 5.6 Impact of Video Resolutions

We further evaluate the power saving of QuRate under various video resolutions, which serves the basis of analyzing QuRate under the DASH streaming scenario. By applying a high-resolution input video to the DASH packager, i.e., Bitmovin [5], we obtain five videos with different resolutions, namely 1920×1080, 1280×682, 854×454, 640×340, and 426×226. We then apply the user data of Users 10 – 19 watching Video 1 on LG V20 with the five videos and measure the power consumption. Figure 15 shows the average power results in each case. The numbers on the short bars indicate the percentage of power saving of QuRate.

We note that as the resolution of the video drops, the power consumption decreases. This is because lower resolutions typically lead to smaller video sizes, which requires lower power during the transmission. We also note that the power saving of QuRate increases from 28.97\% to 31.92\% as the resolution decreases. This is consistent with our expectation as for low resolution cases, the percentage of power cost in view generation is larger. Consequently, the same proportion of power saving in view generation would lead to larger total savings. For instance, the percentages of view generation in high and low video resolutions are 40\% and 60\%, respectively. Assuming QuRate saves 50\% power in the view generation, there will be 20\% savings in the high resolution video, and 30\% savings in the low resolution case.
5.7 Subjective QoE Evaluation

In order to fully evaluate the impact of frame rate reduction on the perceivable QoE, we conduct an IRB-approved subjective experiment. We recruited 14 participants (6 male and 8 female with average age of 25-year-old) from different academic fields across the campus. 9 out of the 14 participants have had past experiences watching immersive videos. Each participant is asked to watch 6 groups of videos, each of which contains two versions of the same immersive video with and without QuRate applied (i.e., Case QuRate and Case Default). After watching each group, the participants were asked to fill up a questionnaire as shown in Table 6 to specify which of the two videos in each group has higher quality. The participants can choose A, B, or “there is no difference”. The feedback from the subjective experiment indicates that 100% (i.e., 14 out of 14) of the participants did not observe any difference in the qualities of all the videos when QuRate is applied. This result meets with our expectation that users would pay less attention while switching views with VR videos, and the reduced frame rate did not significantly reduce the perceivable video quality.

Table 6: IRB-approved user study questionnaire for subjective QoE evaluation of QuRate.

<table>
<thead>
<tr>
<th>Video</th>
<th>A</th>
<th>B</th>
<th>No Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group 6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 CONCLUSION

We investigated the power optimization of immersive video streaming on smartphones. Based on the unique power characteristics of the immersive video streaming system, we developed a quality-aware and user-centric frame rate control mechanism, namely QuRate, which optimizes the power consumption while considering the perceivable video quality and user head movements. QuRate only reduces the frame rate when the user view is switching and assigns optimal frame rates dynamically to maintain a premium and stable video quality. Our experimental results based on an empirical user movement dataset show 5.62% to 32.74% power savings compared to the baseline approach. Also, the battery stress test on five different phones indicates that QuRate can extend the battery life by 68.5% to 124.3%. The results of subjective user study indicated that 100% users did not observe any quality degradation with QuRate applied.
Figure 12: Power consumption of different frame rates in Case Naive (10 FPS, 20 FPS, and 30 FPS) compared to Case Default and Case QuRate.

Figure 13: Comparison of power measurement and power model for QuRate from Figure 9.

Figure 14: Battery stress test results.

Figure 15: Power comparison of Cases Default and QuRate in five different video resolutions (data collected from average value of Users 10 - 19 watching Video 1 using LG V20).

ACKNOWLEDGEMENT
We appreciate the constructive review comments from the reviewers. This work was supported in part by the National Science Foundation under awards CNS-1912593, CNS-1616087, CNS-1618931, and CNS-1755659.

REFERENCES


